Universidade Federal da Bahia
Escola Politécnica
Colegiado do Curso de Eng. Elétrica

Jessé de Oliveira Santana Alves

Robust Trajectory Control of an Unmanned Aerial Vehicle

Supervisor: Prof. Dr. Humberto Xavier de Araújo
Co-supervisor: Prof. Dr. Carlos Eduardo Viana Nunes

Salvador-Ba – Brasil
07 de julho de 2022

Jessé de Oliveira Santana Alves

Robust Trajectory Control of an Unmanned Aerial Vehicle

Project presented to the Undergraduate Course in Electrical Engineering at the Federal University of Bahia as part of the requirements for obtaining the degree of Electrical Engineer.

Supervisor: Prof. Dr. Humberto Xavier de Araújo

Co-supervisor: Prof. Dr. Carlos Eduardo Viana Nunes

Salvador-Ba – Brasil

07 de julho de 2022

Jessé de Oliveira Santana Alves

Robust Trajectory Control of an Unmanned Aerial Vehicle

Project presented to the Undergraduate Course in Electrical Engineering at the Federal University of Bahia as part of the requirements for obtaining the degree of Electrical Engineer.

Trabalho aprovado. Salvador-Ba – Brasil, 07 de julho de 2022:

Supervisor: Prof. Dr. Humberto Xavier de Araújo

Coorientador: Prof. Dr. Carlos Eduardo Viana Nunes

Prof. Dr. Bernardo Ordoñez

Prof. Dr. Tito Luís Maia Santos

Prof. Dr. Tiago Trindade Ribeiro

Salvador-Ba – Brasil
07 de julho de 2022

This work is dedicated to my mother, Angelica de Oliveira Santana Alves and my father, José Bonifácio Silva Alves.

Agradecimentos

I thank my God, who is the architect, designer, guide, executor and restorer of my career and my life.

To my parents, who were the biggest and longest investors in education in my life. Being protagonists in the formation of my character, in guiding the “path I should walk” and in building the value that teaching has in my life.

To my dear wife, who has shared this story with me since high school, being my support in all difficult moments and being my companion and contributor in all achievements.

To my advisor, who was one of the main professors to reveal the universe of control engineering to me, sharing his vast knowledge and encouraging me to learn more.

To my co-supervisor, who was present from the fundamentals of signals and control until the completion of my course, becoming not only a participant in my training, but also a friend.

To my internship supervisor Luciano Coelho, who introduced me to the job market, being exceptional in his dedication to instructing, monitoring and motivating the completion of the proposed activities. Working with a tutor in the process of innovating and going beyond my capabilities.

To my extraordinary and surviving Federal University of Bahia, responsible for building a technically and politically qualified professional, grounded in theory and self-taught. Being trained by her was certainly a great privilege.

"Have not I commanded thee?
Be strong and of a good courage;
be not afraid, neither be thou dismayed:
for the LORD thy God is with thee
whithersoever thou goest.".
(Joshua 1:9)

Resumo

Nos últimos anos, o controle automático de trajetória para um Veículo Aéreo Não Tripulável (VANT) quadrirrotor tem aumentado a cada dia. Neste contexto, é necessário compreender a dinâmica deste robô, representá-lo por um modelo matemático preciso e projetar um bom controlador, capaz de garantir estabilidade e desempenho para a realização de tarefas variadas. Diante dessa demanda, o presente projeto estuda o VANT do tipo quadrirrotor AR.Drone 2.0, abordando brevemente a história de seu desenvolvimento, analisando os conceitos básicos de sua operação, desenvolvendo a modelagem do sistema e projetando dois controladores: o controlador LQR com alocação de polos e o controlador LQR Robusto com alocação de polos. Por fim, este trabalho analisa e discute a robustez das diferentes técnicas. O desenvolvimento dessas técnicas de controle é realizado utilizando-se desigualdades matriciais lineares (LMIs). Os algoritmos são programados com o *LMI toolbox* do MATLAB e as simulações são feitas com o auxílio da ferramenta SIMULINK.

Palavras-chave: VANT; Quadrirrotor; LQR; Robustez; LMI.

Abstract

Currently, automatic trajectory control for quad-rotor Unmanned Aerial Vehicle (UAV) has increased every day. In this context, it is necessary to understand the dynamics of this robot, represent it by a precise mathematical model and design a good controller, capable of guaranteeing stability and performance to perform various tasks. Faced with this demand, the present project studies the AR.Drone 2.0 UAV type, briefly approaching the history of its development, analyzing the basic concepts of its operation, developing the modeling of the system and designing two controllers: the LQR controller with pole allocation and the LQR Robust controller with pole allocation. Finally, this work analyzes and discusses the robustness of the different techniques. The development of these control techniques is carried out using linear matrix inequalities (LMIs). The algorithms are programmed with MATLAB's LMI toolbox and simulations are performed with the help of the SIMULINK tool.

Keywords: UAV; Quadrotor; LQR; Robustness; LMI.

Lista de ilustrações

Lista de tabelas

Lista de símbolos

VANT	Veículo Aéreo Não Tripulável
$< w >$	Referencial da terra
$< b >$	Referencial do VANT
x	Deslocamento linear do VANT no eixo x para o referencial $< w >$
y	Deslocamento linear do VANT no eixo y para o referencial $< w >$
z	Deslocamento linear do VANT no eixo z para o referencial $< w >$
θ	Ângulo de arfagem do VANT no eixo y para o referencial $< w >$
ϕ	Ângulo de rolagem do VANT no eixo x para o referencial $< w >$
ψ	Ângulo de guinada do VANT no eixo z para o referencial $< w >$
$\mathcal{R}$	Matriz de rotação
u_θ	Sinal de controle do subsistema (θ,x)
u_ϕ	Sinal de controle do subsistema (ϕ,y)
$u_{\dot{z}}$	Sinal de controle do subsistema (z)
$u_{\dot{\psi}}$	Sinal de controle do subsistema (ψ)
$\Delta v_{1,2,3,4}$	Variação de tensão aplicada em cada motor do VANT
$\omega_{1,2,3,4}$	Velocidades angular de cada rotor do VANT
$f_{1,2,3,4}$	Força de propulsão de cada hélice
F	Força total de propulsão da aeronave
$\boldsymbol{f}$	Vetor de força que atua sobre o veículo
$\boldsymbol{\tau}$	Vetor de torque que atua sobre o veículo
ξ	Vetor de deslocamento linear
η	Vetor de posição angular
K_θ	Ganho do subsistema (θ,x)

K_ϕ	Ganho do subsistema (ϕ,y)
$K_{\dot{z}}$	Ganho do subsistema (z)
$K_{\dot{\psi}}$	Ganho do subsistema (ψ)
ω_θ	Frequência natural do subsistema (θ,x)
ω_ϕ	Frequência natural do subsistema (ϕ,y)
ζ_θ	Coeficiente de amortecimento do subsistema (θ,x)
ζ_ϕ	Coeficiente de amortecimento do subsistema (ϕ,y)
$\tau_{\dot{z}}$	Constante de tempo do subsistema (z)
$\tau_{\dot{\psi}}$	Constante de tempo do subsistema (ψ)
$\theta_{máx}$	Valor máximo do ângulo de arfagem
$\phi_{máx}$	Valor máximo do ângulo de rolagem
$\dot{z}_{máx}$	Valor máximo da velocidade de subida
$\dot{\psi}_{máx}$	Valor máximo da velocidade de guinada
C_x	Coeficiente de arrasto translacional do eixo x
C_y	Coeficiente de arrasto translacional do eixo y
$\succ$	Definida positiva
$\succeq$	Semi-definida positiva
$\prec$	Definida negativa
$\approx$	Aproximadamente
$\in$	Pertence

Sumário

1 INTRODUCTION

UAV (unmanned aerial vehicle), also known as *drone*, is a self-explanatory term to define aircraft that do not require pilots on board to navigate (GARCIA P.C.; LOZANO, 2006). There is also a specific term for small UAVs, called micro aerial vehicles (MVA) (NONAMI *et al.*, 2010). These small aerial robots are the ones that have been most used in civil applications around the world, whether for filming, topography, food delivery in *delivery* services, use in the healthcare sector, among others. Currently, several models of UAVs have been developed for different civil applications, however, historically, investments for the development of this device came through military initiatives. The first UAV was manufactured in 1916 by *Americans Lawrence and Sperry*, shown in Figure 1, where a gyroscope was used to stabilize the aircraft. This is one of the first aircraft orientation and position controls (NONAMI *et al.*, 2010). However, this UAV did not have enough technology to be used in the first and second world wars. It was only at the end of the 1950s - encouraged by the Vietnam War - that advances became increasingly greater and UAVs were proving to be advantageous in military terms, as the absence of a pilot on board allows for greater maneuverability and lower weight. , lower cost, more flight resistance and no direct risk to the pilot's health (GARCIA P.C.; LOZANO, 2006). Therefore, at the end of the Vietnam War, the United States and Israel began increasing research into building smaller and cheaper aircraft, and one of the products of this research was the use of *drones* in the Gulf War, in 1991 (NONAMI *et al.*, 2010).

Fig. 1 – First UAV, created in 1916.

Source: Adapted from (NONAMI *et al.*, 2010).

Since then, large-scale investments have become present in several countries. The most famous military aircraft in this category is called *Predator*, shown in Figure 2. However, in the 1990s, research was not restricted to military use. At NASA, the APATS (Aircraft for Environmental Research and Sensor Technology) project is an example of the application of this aircraft for civil use. In this project, with the aim of carrying out environmental measurements, there was a major technological advance related to the altitude reached by the *drone*, reaching up to 30,000 meters, the flight time and the type of engines and sensors (NONAMI *et al.*, 2010) .

Fig. 2 – *Predator*: the most famous UAV for military use.

Source: Adapted from (NONAMI *et al.*, 2010).

Over the past three decades, the applications and types of aerial robots have grown exponentially. This worldwide growth has brought innovations in the size of UAVs, as well as in the type of flight technology, type of trajectory and stabilization controls and *software* navigation. Currently, with this great diversity of military and civil aircraft, several classifications can be made, for example: regarding average weight, risk of impact on the ground, operating altitude, autonomy, types of platforms, among others (VALAVANIS; VACHTSEVANOS, 2015). In this study, only the classification regarding the type of platform will be explained. According to (NONAMI *et al.*, 2010), there are four types:

1. Fixed-wing UAVs, which are aircraft with wings and propulsion by propellers or turbines. These require a runway for takeoff and landing or a type of catapult that launches them into the air. Generally, they are more resistant and have longer flight time (Figure 3);

2. Rotary-wing UAVs or helicopter-type UAVs. These aircraft are also classified in aerodynamic studies as Vertical Takeoff and Landing (DPV) (GARCIA P.C.; LOZANO, 2006), that is, they do not require an extensive area for takeoff and landing, being attractive for civil use due to this advantage. . This *drone* model nowadays

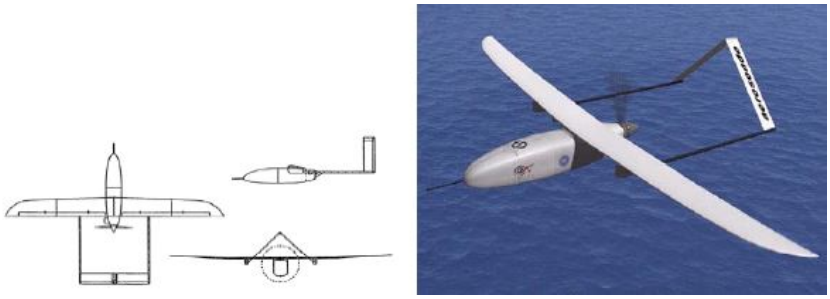


Fig. 3 – Fixed-wing UAV.

Source: (GARCIA P.C.; LOZANO, 2006).

has different types of configurations, as well as number of propellers, as can be seen in Figure 4.

(a) UAV Helicopter

(b) Quadcopter type UAV

(c) Hexacopter UAV

(d) Octocopter UAV

Fig. 4 – Rotary Wing UAVs.

Source: Adapted from (NONAMI *et al.*, 2010; PISKORSKI NICOLAS BRULEZ, 2012; DRONES, 2021; CHINA, 2021).

3. Balloon or airship-type UAVs: they are generally very large and fly at low speeds (Figure 5).

Fig. 5 – Airship Type.

Source: Adapted from (NONAMI *et al.*, 2010).

4. UAVs with oscillating wings: these are generally very small and have aerodynamics that imitate the movement of insect wings (Figure 6).

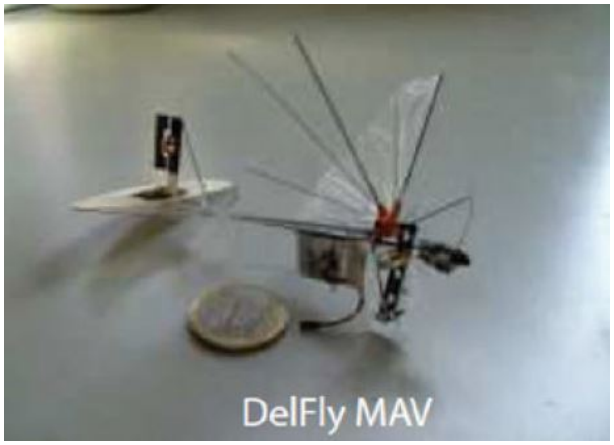


Fig. 6 – Oscillating Wings Type.

Source: Adapted from (NONAMI *et al.*, 2010).

Currently, UAVs are applied, in the public and private sector, in inspections of land, pipelines, public services and buildings, search and rescue, environmental monitoring, agriculture and forestry, firefighting, aerial mapping and meteorology, research in university laboratories, among others (NONAMI *et al.*, 2010). Furthermore, the market for the commercialization of *drones* for the purposes of filming and photography, entertainment and other applications for the population, that is, for individuals, has grown a lot in recent decades, resulting in cheaper and resizing of aircraft to operate in the middle of cities, closed places and even indoors. As, in these cases, space for takeoff and landing,

the size of the aircraft and its cost are priority factors, small rotary-wing UAVs, generally quadcopters (Figure 4b) are the most used.

This growing demand for missions assigned to aerial robots in recent years has driven research - in academic and corporate circles - towards increasingly sophisticated trajectory control projects, which promote stability, good disturbance rejection rates and settling time, robustness, optimization of control effort, battery and cost savings, considering the non-linearity of the mathematical model that describes the aircraft. To achieve this, different control techniques are applied and often combined to guarantee the specifications of the mission that the *drone* will carry out. Among these techniques, we highlight the proportional, integral and derivative (PID) controller (BOUABDALLAH, 2007; COSTA, 2018; BRESCIANI, 2008; BOUABDALLAH; NOTH; SIEGWART, 2004; LIMA, 2015; RAGHURAMAN, 2018), the LQR controller (BOUABDALLAH, 2007; BOUABDALLAH; NOTH; SIEGWART, 2004; COSTA, 2018; RAGHURAMAN, 2018), the model-based predictive control (CPBM) (LIMA, 2015), the H_∞ control (RAGHURAMAN, 2018), the controller *BackStepping* (BOUABDALLAH, 2007) and the controller *Sliding-mode* (BOUABDALLAH, 2007).

The complexity of the mathematical description of UAVs is a challenge that any implemented control strategy will have to deal with, as the aerial robot has several non-linear physical phenomena. Several academic works in recent years, such as (BOUABDALLAH, 2007; VALAVANIS; VACHTSEVANOS, 2015), demonstrate that modeling a *drone* quadrirotor can add many levels of detail, from the representation of the movement of a rigid body, to the model of actuators, sensors and aerodynamic effects present in the aircraft's flight environments (SANTANA, 2016). Currently, the presence of the autopilot system present in many modern *drones* has popularized both its civil use, whether recreational or for commercial applications, as well as research into different control techniques at universities. This is due to the fact that this automation module is responsible for stabilizing the vehicle's attitude, or orientations, and is only modified to respond to external commands requested by the operator (SANTANA, 2016). Furthermore, autopilot systems, as they consist of sensors such as video camera, GPS, compasses and others, central processing units and algorithms that perform low-level functions (CHAO; CAO; CHEN, 2010), are responsible for sending signals to the actuators, for processing sensor data and managing communications between the operator and the vehicle. This allows trajectory control algorithms to be developed based on a simplified high-level mathematical model, where abstract inputs - that is, inputs other than those that represent real physical phenomena - are defined in order to facilitate the work of control and remove the need for access to *firmware* from *drone*. Works such as (KRAJNÍK *et al.*, 2011; SANTANA, 2016) demonstrate that this simplification strategy brings satisfactory results for many applications.

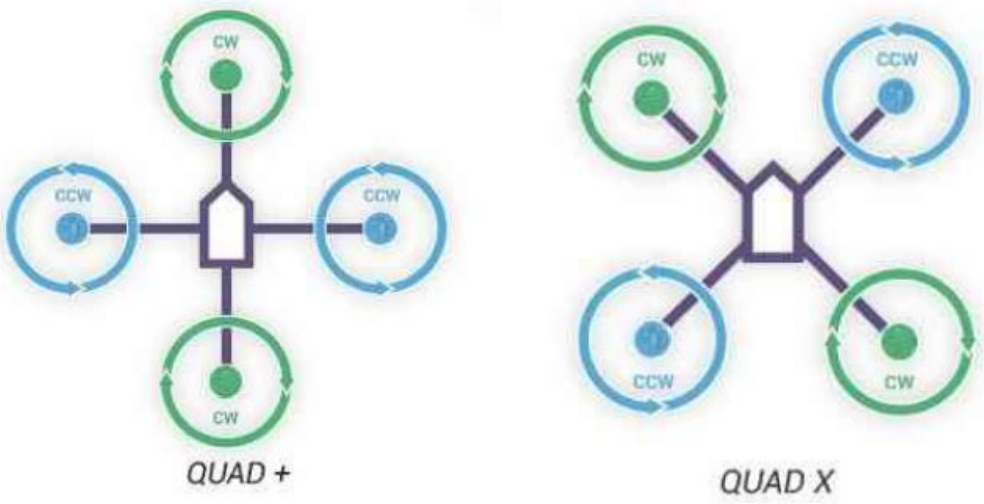
1.1 Basic Concepts of a Quadrotor

The conventional four-rotor or quadcopter unmanned aerial vehicle has an architecture in which the four engines are arranged symmetrically at the corner of rods connected to a central structure (Figure 7). This is the most popular type of *drone* in civilian circles. This popularity is due to the fact that the four propellers are enough to move the vehicle in any direction in space just by varying the propeller speeds, with good stability for most applications, therefore, a lower cost compared to hexacopters and octocopters. Furthermore, using four propellers allows them to be smaller, requiring less kinetic energy from the engines, again reducing the cost and also being more advantageous than conventional central propeller helicopters (DA COSTA, 2008).

Fig. 7 – Conventional quad-rotor UAV.

Source: Adapted from (DA COSTA, 2008).

Therefore, for this type of *drone*, two symmetrical configurations of the axes are possible to guarantee stability: in a cross shape, QUAD +, (Figure 8a) and in an x-shape, QUAD X (Figure 8b). In addition to the symmetry of the rotors on the axes, the direction of rotation of the propellers also needs to be arranged to ensure stability. In the study and development of Vertical Take-Off and Landing type aircraft (from English *Vertical Take-off or Land - VTOL*) it is demonstrated that the rotational action of the propeller produces an opposite torque on the aircraft body (GARCIA P.C.; LOZANO, 2006), a fact that explains the presence of a compensating propeller in the tail of conventional helicopters (Figure 4a).



(a) Cross Shape (b) Xis Format

Fig. 8 – Possible arrangements of the axes of a Quad-rotor UAV.

Source: Adapted from (COSTA, 2018).

In quadcopter UAVs, the balance of these resistant torques is carried out by placing the motors that are on the same axis to rotate in the opposite direction to the other two on the other axis, that is, motors 1 and 2 in Figure 9 rotate in the opposite direction of engines 3 and 4, in order to force the angular acceleration of the vehicle body to zero (DA COSTA, 2008), ensuring aerodynamic balance . Therefore, a tail propeller is unnecessary, allowing all engines to act on the vehicle's lifting force.

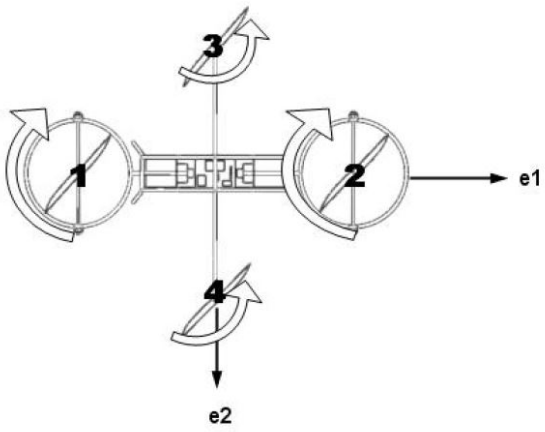
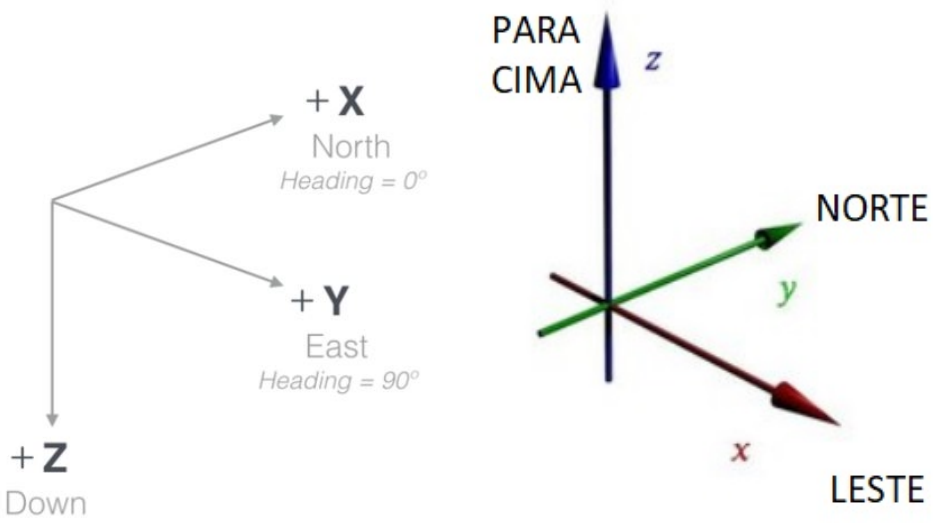


Fig. 9 – Direction of Rotation of Quadcopter Propellers.

Source: Adapted from (DA COSTA, 2008).

Once the balance of the aircraft is guaranteed, depending on the symmetry of its structure and the direction of rotation of the propellers, two inertial references must be adopted for the UAV capable of describing the vehicle's attitude dynamics, that is, the

orientation and movements of the aircraft. *drone* in space. The first reference is that of the earth, being considered fixed at a point on the earth's surface and can have two different types of coordinates, the *North, East, Down* (NED) or the coordinate *East, North, Up* (ENU) (COSTA, 2018). The NED coordinate has the x-axis facing north, the y-axis facing east, and the z-axis facing down Figure (10a). The ENU coordinate has the x-axis facing east, the y-axis facing north and the z-axis facing up (Figure 10b).

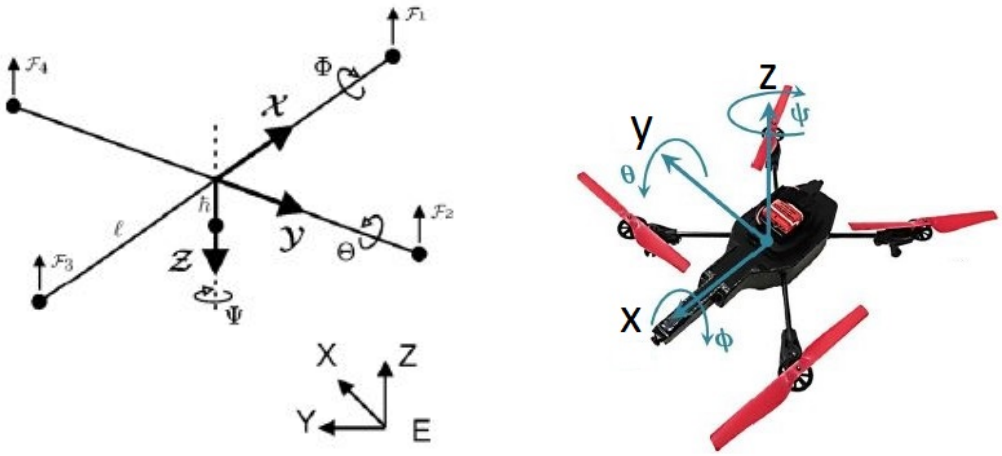


(a) NED Coordinate (b) ENU coordinate

Fig. 10 – Types of Coordinates for Earth's Inertial Reference.

Source: Adapted from (VUELAS SEGURO, 2021; WIKILIVROS, 2021).

The second reference is that of *drone*, being considered fixed at the vehicle's center of gravity, which for mathematical modeling purposes is defined as a rigid body. Likewise, this reference can assume two different types of orthonormal bases, the first has the x axis facing the front of the vehicle, the y axis facing right and the z axis facing downwards, aligned with the gravitational acceleration vector (Figure 11a). The second coordinate has the x axis also facing the front of the aircraft, the y axis facing the left side and the z axis facing upwards, in the direction of the aircraft's thrust (Figure 11b).



(a) Coordinate with Z axis facing Down (b) Coordinate with Z axis facing Up

Fig. 11 – Types of Coordinates for Reference Fixed in the UAV.

Source: Adapted from (BOUABDALLAH, 2007; SANTANA, 2016).

Once the references are defined, it can be seen that there are six types of possible movements for the *drone*: three of translation, in relation to the earth's inertial reference, in the aircraft's three-dimensional flight environment, and three of rotation in relation to the reference of the UAV's rigid body. These rotational movements are called, in the study of aerodynamics, yaw (in English *yaw*), pitch (in English *pitch*) and roll (in English *roll*). Yaw is defined as the rotation around the z axis, the pitch, the rotation around the y axis, the roll, the rotation around the x axis (COSTA, 2018), as can be seen in Figure 12.

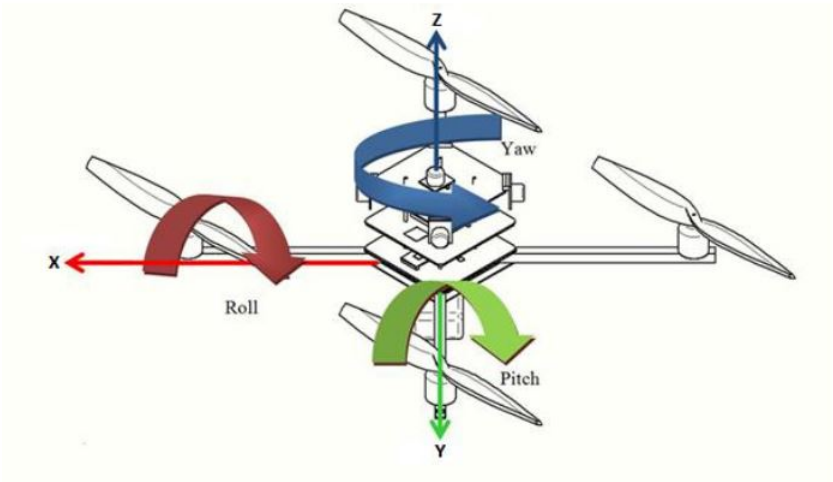


Fig. 12 – Three types of aircraft rotation.

Source: (COSTA, 2018).

Finally, it is observed that for a quadrirotor all six types of movements are carried out just by modifying the rotation speed of each propeller. To this end, Figure 13 shows how varying the speed of the propellers causes all these movements on each axis. For example, forward displacement is carried out when the rear motor rotates at a higher speed than the front motor. And in rotation, the counterclockwise yaw movement is carried out when engines 1 and 3 in Figure 12 rotate at a higher speed than engines 2 and 4, in order to create the angular acceleration mentioned in Figure 9.

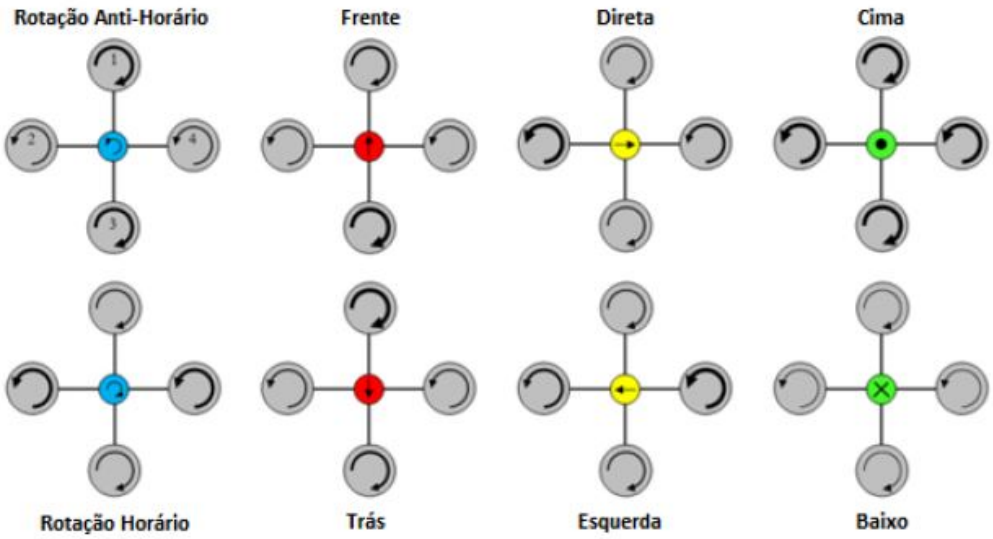


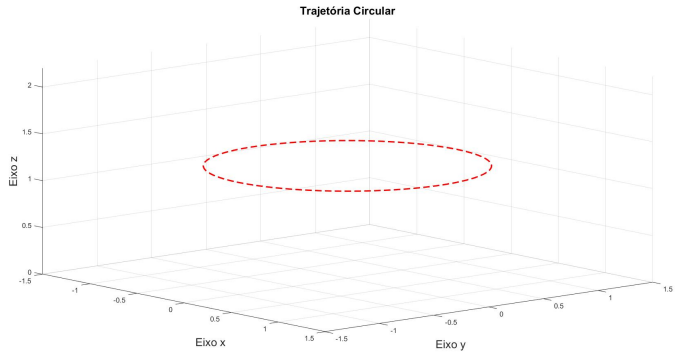
Fig. 13 – Types of Movement of a Quadrotor UAV.

Source: (COSTA, 2018).

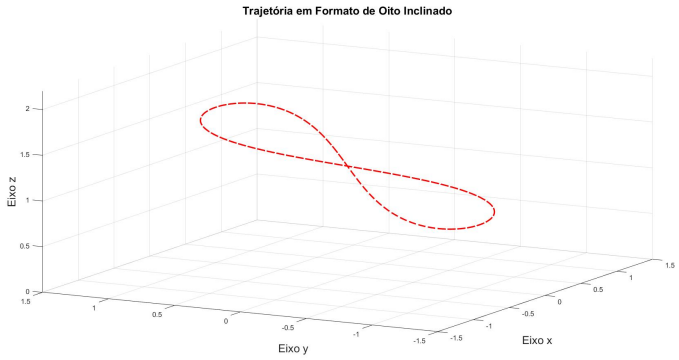
1.2 Work Objective

The objective of this work is to study a four-rotor UAV and design a controller for trajectory tracking, considering a fixed coordinate system on the ground, so that asymptotic stability is guaranteed, minimizing control effort. To achieve this, using a simplified model that disregards some aerodynamic effects and considers the existence of an autopilot, a linearized model of the UAV will be obtained. Subsequently, three trajectory controllers will be designed, an optimal LQR controller with integral action and pole allocation for a model defined as nominal, a robust LQR controller with integral action and pole allocation for a model with parametric uncertainties and a robust LQR controller with integral action and pole allocation for a model with larger ranges of parametric uncertainties. Verification of this trajectory control will be done by analyzing the reference tracking of the system output, in this case, the position of the *drone* in space. The test trajectories for the system in this project are of two types: one circular, at a height of 1.2

m from the ground (Figure 14a), and the other in an inclined eight shape (Figure 14b). Following these trajectories is used to verify the controller's performance, as they require simultaneous control of the four degrees of freedom commanded by the UAV's piloting signals.



(a) Circular trajectory



(b) Trajectory in the shape of a tilted figure of eight

Fig. 14 – Reference Trajectories for the UAV.

2 AR.DRONE 2.0 MODELING

The first stage of the work consists of determining a model that mathematically describes the dynamics of the AR.Drone 2.0 quad-rotor UAV (Figure 15). For this, based on classical methods from the literature, the general model is studied and subsequently a simplified and linearized model is considered for the development of the controller project. Therefore, in this chapter, nonlinear and linear kinematic and dynamic models, programming of the nonlinear model in the SIMULINK environment and its validation will be discussed.

Fig. 15 – UAV AR.Drone 2.0.

Source: Adapted from (PISKORSKI NICOLAS BRULEZ, 2012)

2.1 Kinematic Model

In the UAV modeling process, two frames of reference are considered: the global frame of reference, also called the earth's frame of reference $< w >$ and the frame of reference for the *drone* $< b >$, whose origin is located at the center of gravity of the robot (Figure 16). In addition to the references, the angular speeds of each rotor (ω_1, ω_2, ω_3 and ω_4), the propulsion forces of each propeller (f_1, f_2, f_3 and f_4), which generate a total propulsion force represented by F, the weight of the UAV mg, a force vector $\mathbf{f} = [f_x \quad f_y \quad f_z]^T$ that act on the vehicle, here considered a rigid body, and a vector of torques $\boldsymbol{\tau} = [\tau_x \quad \tau_y \quad \tau_z]^T$ that act on the vehicle.

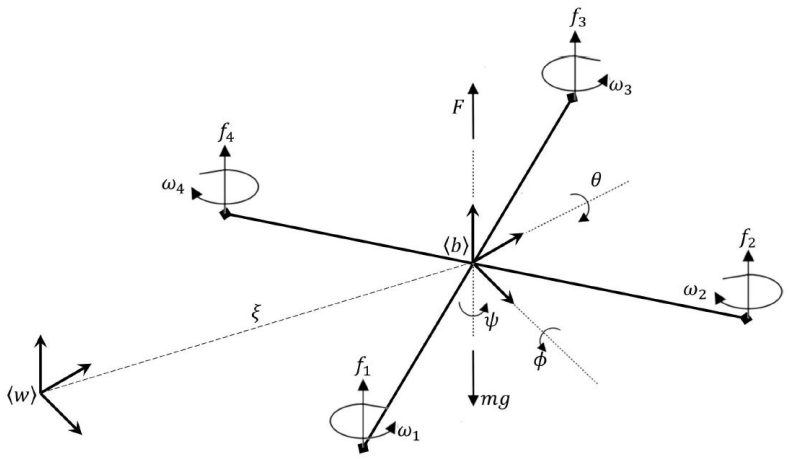


Fig. 16 – Global reference and UAV reference.

Source: Adapted from (SANTANA, 2016)

In order for there to be a relationship between the two references, we define, for the reference $< w >$, $\mathbf{q} = [\xi^T \quad \eta^T]^T$, with $\xi = [x \quad y \quad z]^T$ representing the displacement of the translational movement and $\eta = [\phi \quad \theta \quad \psi]^T$ representing the angles roll, pitch and yaw, respectively (SANTANA, 2016). And, for the reference $< b >$, the linear velocity vector $\mathbf{V}_b = [v_x \quad v_y \quad v_z]^T$ and the angular velocity vector $\Omega aredefined = [p \quad q \quad w]^T$. Therefore, relating the two references, we have:

$$\dot{\xi} = \mathcal{R}\mathbf{V}_b, \tag{2.1}$$

$$\dot{\eta} = \mathcal{W}_n^{-1}\Omega, \tag{2.2}$$

where the rotation matrices $\mathcal{R}$ and the matrix $\mathcal{W}_n$ are given by,

$$\mathcal{R} = \begin{bmatrix} cos(\psi)cos(\theta) & cos(\psi)sen(\theta)sen(\psi) - sen(\psi)cos(\phi) & cos(\psi)sin(\theta)cos(\psi) + sin(\psi)sin(\phi) \\ sin(\psi)cos(\theta) & sen(\psi)sen(\theta)sen(\psi) + cos(\psi)cos(\phi) & sin(\psi)sin(\theta)cos(\psi) - cos(\psi)sen(\phi) \\ -sen(\theta) & cos(\theta)sen(\phi) & cos(\theta)cos(\phi) \end{bmatrix}, \tag{2.3}$$

$$\mathcal{W}_n^{-1} = \begin{bmatrix} 1 & \frac{sen(\phi)sen(\theta)}{(cos(\phi)^2cos(\theta)+cos(\theta)sen(\phi)^2)} & \frac{cos(\phi)sen(\theta)}{cos(\phi)^2cos(\theta)+cos(\theta)sen(\phi)^2)} \\ 0 & \frac{cos(\phi)}{cos(\phi)^2+sin(\phi)^2} & \frac{-sen(\phi)}{cos(\phi)^2+sin(\phi)^2} \\ 0 & \frac{sin(\phi)}{cos(\phi)^2cos(\theta)+cos(\theta)sin(\phi)^2} & \frac{cos(\phi)}{cos(\phi)^2cos(\theta)+cos(\theta)sin(\phi)^2} \end{bmatrix}. \tag{2.4}$$

2.2 Dynamic Model

However, a better description of the system is represented through the dynamic model. Which relates the vector of control signals $\mathbf{u} = [u_\theta \; u_\phi \; u_\psi \; u_{\dot{z}}]$, with the position vectors ξ and vehicle angle η. This model is built didactically through separate blocks and divided (BRANDÃO, 2013) into two main categories: the low-level dynamic model and the high-level dynamic model (Figure 17).

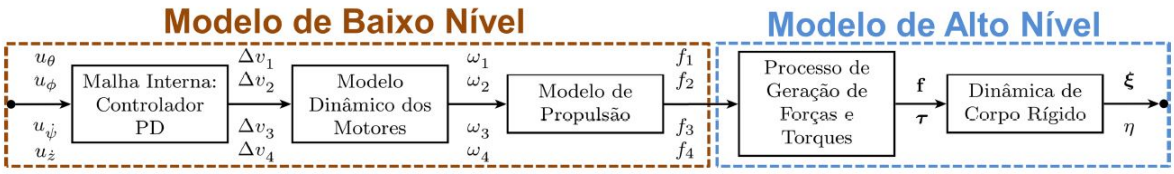


Fig. 17 – Low-level dynamic model and high-level dynamic model.

Source: Adapted from (SANTANA, 2016).

2.2.1 Low Level Model

The low-level model represents the dynamics between the control signal **u** and the propulsion forces (f_1, f_2, f_3 and f_4) that each propeller causes on the propeller shafts. *drone*. And this dynamic is divided into three main blocks. In the first block, Internal Mesh, there is a Proportional-Derivative (PD) controller, whose inputs are the desired values of the pitch angles θ_d and roll ϕ_d, and the yaw speeds $\dot{\psi}_d$ and rising $\dot{z}_d$. And the outputs are the voltage variations caused in the *drone* motors (Δv_1, Δv_2, Δv_3 and Δv_4). However, for the aerodynamic stability of the UAV to be guaranteed, there are maximum values for inclinations and speeds: θ_{max}, ϕ_{max}, $\dot{\psi}_{max}$ and $\dot{z}_{max}$. And these values are related to the normalized control signal **u** through the equations:

$$\theta_d = u_\theta \theta_{max} \quad com \quad u_\theta \in [-1; 1], \tag{2.5}$$

$$\phi_d = u_\phi \phi_{max} \quad com \quad u_\phi \in [-1; 1], \tag{2.6}$$

$$\dot{\psi}_d = u_\psi \dot{\psi}_{max} \quad com \quad u_\psi \in [-1; 1], \tag{2.7}$$

$$\dot{z}_d = u_{\dot{z}} \dot{z}_{max} \quad com \quad u_{\dot{z}} \in [-1; 1], \tag{2.8}$$

and the inner mesh block is defined as

$$\begin{bmatrix} \Delta v_1 \\ \Delta v_2 \\ \Delta v_3 \\ \Delta v_4 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} k_{p\theta} e_\theta + k_{d\theta} \dot{e}_\theta \\ k_{p\phi} e_\phi + k_{d\phi} \dot{e}_\phi \\ k_{p\dot{\psi}} e_{\dot{\psi}} + k_{d\dot{\psi}} \dot{e}_{\dot{\psi}} \\ k_{p\dot{z}} e_{\dot{z}} + k_{d\dot{z}} \dot{e}_{\dot{z}} \end{bmatrix}, \tag{2.9}$$

where k_{pi} are the proportional gains, k_{di} are the derivative gains, where $i = \theta$,ϕ,$\dot{\psi}$ e $\dot{z}$, $e_\theta = \theta_d - \theta$ is the pitch error, $e_\phi = \phi_d - \phi$ is the roll error, $e_{\dot{\psi}} = \dot{\psi}_d - \dot{\psi}$ is the yaw speed error and $e_{\dot{z}} = \dot{z}_d - \dot{z}$ is the climb speed error.

The second block represents the modeling of each *drone* motor, which are generally used brushless motors (*brushless*), whose modeling is the same as that of a conventional DC motor (SANTANA, 2016). Therefore, in this block there is a relationship between the voltage applied to each motor (Δv_1, Δv_2, Δv_3 and Δv_4) and the angular velocity of each

helix (ω_1, ω_2, ω_3 and ω_4), through the following equation:

$$v_m = \frac{RJ_m}{k_m}\dot{\omega_m} + \left(\frac{RB_m}{k_m} + k_b\right)\omega_m + \frac{R}{rk_m}\tau_l, \tag{2.10}$$

where, R is the electrical resistance of the motor, J_m is the moment of inertia of the motor shaft, k_m is the constant that relates the counter-electromotive force to the angular speed of the motor, k_b is the constant that relates the armature current to the motor torque, ω_m is the angular velocity developed by the motor, B_m is the gear dissipative term, r is the gear ratio, τ_l is the load torque, v_m is the voltage applied to the engine, also determined by $v_m = v_o + \Delta v_m$, where v_o is the voltage required to balance the weight of the aircraft and perform a flight hovered, and Δv_m is the voltage variation leaving the first block.

In the third and final block, the propeller propulsion model is considered, and the relationship between the angular speed of each engine (ω_1, ω_2, ω_3 and ω_4) and the propulsion force (f_1, f_2, f_3 and f_4) is described by the following equation (VALAVANIS; VACHTSEVANOS, 2015):

$$f_i = C_f\omega_m^2 \quad com \;\; i = [1,2,3,4], \tag{2.11}$$

where C_f is an aerodynamic parameter that depends on the construction of the rotor, here considered constant.

2.2.2 High Level Model

In the second category, the high-level model represents the dynamics between the propulsion forces of each propeller and the linear and angular position of the *drone* in space. In this model, the first block relates the propulsion forces to the total force vector $\mathbf{f} = [f_x \;\; f_y \;\; f_z]^T$ and the total torque vector $\boldsymbol{\tau} = [\tau_x \;\; \tau_y \;\; \tau_z]^T$. The first relationship, seen in (SANTANA, 2016), is given by:

$$\mathbf{f} = [f_x \;\; f_y \;\; f_z]^T = \mathcal{R}\,[0 \;\; 0 \;\; F]^T, \tag{2.12}$$

where $\mathcal{R}$ is the rotation matrix (2.3) and F is the total propulsion force given by:

$$F = \sum_{i=1}^{i=4} f_i, \quad where \;\; f_i = C_{f_i}\omega_i^2 \;\; with \;\; i = [1,2,3,4]. \tag{2.13}$$

And the second relationship is given by:

$$\boldsymbol{\tau} = [\tau_x \;\; \tau_y \;\; \tau_z]^T = \mathcal{A}\,[f_1 \;\; f_2 \;\; f_3 \;\; f_4]^T, \tag{2.14}$$

where the matrix $\mathcal{A}$ represents the force arms of the torque and force relationship and is given by:

$$\mathcal{A} = \begin{bmatrix} -k_1 & k_1 & k_1 & -k_1 \\ -k_1 & -k_1 & k_1 & k_1 \\ k_2 & -k_2 & k_2 & -k_2 \end{bmatrix}, \tag{2.15}$$

where k_1 is the distance between the UAV's center of gravity and the engine axis, and k_2 is a constant that relates the touch and the propulsion generated by the engine.

In the second block of this category, the rigid body dynamics model relates the forces and torques applied to the aircraft with its linear and angular position in space. And this relationship is obtained by solving the Euler-Lagrange equation (2.16)

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}} = \begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix}, \tag{2.16}$$

where $\mathbf{q} = [\xi^T \quad \eta^T]^T$ is the position and orientation vector, $\mathbf{f} = [f_x \quad f_y \quad f_z]^T$ is the total force vector, $\boldsymbol{\tau} = [\tau_x \quad \tau_y \quad \tau_z]^T$ is the total torque vector and L is the Lagrangian, represented by the difference between the total kinetic energy K (translational and rotational) and the potential energy U (SANTANA, 2016), given by the relationship:

$$L = K - U = \frac{1}{2} m \dot{\xi}^T \dot{\xi} + \frac{1}{2} \Omega^T \mathbb{I} \Omega - mgz, \tag{2.17}$$

where m is the mass of the UAV, ξ is the linear displacement vector, Ω is the angular velocity vector, g is the acceleration due to gravity and $\mathbb{I}$ is the inertia tensor given by:

$$\mathbb{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}. \tag{2.18}$$

However, the equation (2.16) is solved separately, first the translational part and then the rotational part. Thus, by solving the translational Lagrangian of the equations (2.19) and (2.20), the relations are obtained:

$$L_t = \frac{1}{2} m \dot{\xi}^T \dot{\xi} - mgz, \tag{2.19}$$

$$\frac{d}{dt}\left(\frac{\partial L_t}{\partial \dot{\xi}}\right) - \frac{\partial L_t}{\partial \xi} = \boldsymbol{f}, \tag{2.20}$$

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} + g \end{bmatrix} = \mathcal{R}[0 \quad 0 \quad F]^T. \tag{2.21}$$

In the rotational part, solving the rotational Lagrangian of the equations

$$L_r = \frac{1}{2} \Omega^T \mathbb{I} \Omega, \tag{2.22}$$

$$\frac{d}{dt}\left(\frac{\partial L_r}{\partial \dot{\eta}}\right) - \frac{\partial L_r}{\partial \eta} = \boldsymbol{\tau}, \tag{2.23}$$

a more complex relationship is obtained, described by:

$$\boldsymbol{M}_r \ddot{\eta} + C_r \dot{\eta} = \tau, \tag{2.24}$$

on what,

$$\boldsymbol{M}_r = \mathcal{W}_\eta^T \, \mathbb{I} \, \mathcal{W}_\eta,$$

and C_r is the matrix of Coriolis and centripetal forces, given by

$$C_r = \dot{\boldsymbol{M}}_r - \frac{1}{2} \begin{bmatrix} \dot{\eta} & 0 & 0 \\ 0 & \dot{\eta} & 0 \\ 0 & 0 & \dot{\eta} \end{bmatrix}^T \frac{\partial \boldsymbol{M}_r}{\partial \eta}.$$

2.3 Simplified Dynamic Model

The complete mathematical modeling of the four-rotor UAV is described in sections 2.1 and 2.2. However, this type of modeling uses a large number of equations and parameters, making it difficult to design an aircraft trajectory controller. Furthermore, the successive transformations of signs and approximations during the process increase the potential for errors in the calculations (SANTANA, 2016). Therefore, works such as (ENGEL; STURM; CREMERS, 2014; NONAMI *et al.*, 2010; RAFFO; ORTEGA; RUBIO, 2010) present simplifications of this dynamic model, obtaining good practical results, with (CASTILLO; DZUL; LOZANO, 2004) being one of the pioneers to present this simplified modeling. Based on this, the present project will indicate the simplifications made and will use this model for the design of the trajectory controller.

In the translational part, developing the equation (2.21) and explaining the accelerations $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$, we obtain- if:

$$\begin{cases} \ddot{x} = \dfrac{F}{m}(sin\psi sin\phi + cos\psi cos\phi sin\theta), & (2.25) \\ \ddot{y} = \dfrac{F}{m}(-cos\psi sin\phi + sin\psi cos\phi sin\theta), & (2.26) \\ \ddot{z} = \dfrac{F}{m}(cos\phi cos\theta) - g. & (2.27) \end{cases}$$

Subsequently, two translational drag forces $F_{ax} = C_x\dot{x}$ and $F_{ay} = C_y\dot{y}$ are included in the equations (2.25) and (2.26), respectively. These forces oppose the accelerations of the x and y axes, bringing the model closer to the real behavior at the time of flight. Furthermore, the *drone*'s autopilot only allows small inclinations in the pitch angle, so it is possible to approximate $\frac{F}{m} \approx g$ in the equations (2.25) and (2.26), resulting in:

$$\begin{cases} \ddot{x} = g(sen\psi sin\phi + cos\psi cos\phi sin\theta) - C_x\dot{x}, & (2.28) \\ \ddot{y} = g(-cos\psi sin\phi + sin\psi cos\phi sin\theta) - C_y\dot{y}, & (2.29) \\ \ddot{z} = \dfrac{F}{m}(cos\phi cos\theta) - g. & (2.30) \end{cases}$$

where C_x and C_y are drag coefficients, considered approximately constant.

2.3.1 Linearization

In some works on the UAV AR.Drone 2.0, some hypotheses were raised, as in (KRAJNÍK *et al.*, 2011), and verified, as in (SANTANA, 2016), about how to model some relationships between the control signals $\mathbf{u} = [u_\theta \ u_\phi \ u_{\dot{\psi}} \ u_{\dot{z}}]$ and the position variables $[x \ y \ z \ \theta \ \phi \ \psi]$ in a linear way. In (SANTANA, 2016), the following linear equations are presented:

$$\begin{cases} \ddot{z} = \left(\frac{K_{\dot{z}}\dot{z}_{max}}{\tau_{\dot{z}}}\right)u_{\dot{z}} - \left(\frac{1}{\tau_{\dot{z}}}\right)\dot{z}, & (2.31) \\ \ddot{\phi} = K_\phi\omega_\phi^2\phi_{max}u_\phi - 2\zeta_\phi\omega_\phi\dot{\phi} - \omega_\phi^2\phi, & (2.32) \\ \ddot{\theta} = K_\theta\omega_\theta^2\theta_{max}u_\theta - 2\zeta_\theta\omega_\theta\dot{\theta} - \omega_\theta^2\theta, & (2.33) \\ \ddot{\psi} = \left(\frac{K_{\dot{\psi}}\dot{\psi}_{max}}{\tau_{\dot{\psi}}}\right)u_{\dot{\psi}} - \left(\frac{1}{\tau_{\dot{\psi}}}\right)\dot{\psi}. & (2.34) \end{cases}$$

In these approximate relationships, one can interpret the parameters K_θ, K_ϕ, $K_{\dot{\psi}}$ and $K_{\dot{z}}$ as process gains, ω_θ and ω_ϕ as natural frequencies, ζ_θ and ζ_ϕ as damping coefficients and $\tau_{\dot{\psi}}$ and $\tau_{\dot{z}}$ as time constants. Furthermore, the limiting parameters θ_{max}, ϕ_{max}, $\dot{\psi}_{max}$ and $\dot{z}_{max}$ are the same as those discussed in equations (2.5), (2.6), (2.7) and (2.8) .

Subsequently, the equations (2.28) and (2.29) are linearized considering that the autopilot only allows small inclinations around the x and y axis. Furthermore, in order to facilitate analysis, the coordinate systems $< w >$ and $< b >$ are aligned considering a zero yaw angle. Therefore, the following approximations are considered:

$$\begin{cases} sin(\phi) \approx \phi, \\ cos(\phi) \approx 1{,}0, \\ sin(\theta) \approx \theta, \\ \psi \approx 0.0, \end{cases}$$

resulting in:

$$\begin{cases} \ddot{x} = g\theta - C_x\dot{x}, & (2.35) \\ \ddot{y} = -g\phi - C_y\dot{y}, & (2.36) \end{cases}$$

2.4 Simplified Linear Model

Given the considerations discussed in section 2.3.1, the representation of the system that will be used to design the trajectory controller is given by the simplified and linearized

model described by the equations:

$$\begin{cases} \ddot{x} = g\theta - C_x\dot{x}, & (2.37)\\ \ddot{y} = -g\phi - C_y\dot{y}, & (2.38)\\ \ddot{z} = \left(\frac{K_{\dot{z}}\dot{z}_{max}}{\tau_{\dot{z}}}\right)u_{\dot{z}} - \left(\frac{1}{\tau_{\dot{z}}}\right)\dot{z}, & (2.39)\\ \ddot{\phi} = K_\phi\omega_\phi^2\phi_{max}u_\phi - 2\zeta_\phi\omega_\phi\dot{\phi} - \omega_\phi^2\phi, & (2.40)\\ \ddot{\theta} = K_\theta\omega_\theta^2\theta_{max}u_\theta - 2\zeta_\theta\omega_\theta\dot{\theta} - \omega_\theta^2\theta, & (2.41)\\ \ddot{\psi} = \left(\frac{K_{\dot{\psi}}\dot{\psi}_{max}}{\tau_{\dot{\psi}}}\right)u_{\dot{\psi}} - \left(\frac{1}{\tau_{\dot{\psi}}}\right)\dot{\psi}. & (2.42) \end{cases}$$

The equations (2.37) and (2.41) have a coupling, in which the variable θ determined by the solution of the equation (2.41) is the input value for the equation (2.37). This relationship is intuitive when analyzing Figure (11a), in which the variation in the pitch angle θ causes the vehicle to move on the x axis. This same relationship occurs for the roll angle ϕ and the displacement on the y axis. Considering this characteristic of the model, it is possible to reduce the six equations presented into four subsystems: subsystem (θ,x) composed of the equations (2.41) and (2.37), subsystem (ϕ,y) composed of the equations (2.40) and (2.38), subsystem z composed of the equation (2.39) and subsystem ψ composed of the equation (2.42).

Therefore, representing the four subsystems in the state space, we have:

1. **Subsystem (θ,x)**

Choosing the state variables:

$$\begin{cases} q_1 = \theta(t),\\ q_2 = \dot{\theta}(t),\\ q_3 = x(t),\\ q_4 = \dot{x}(t), \end{cases} \Rightarrow \begin{cases} \dot{q_1} = \dot{\theta}(t),\\ \dot{q_2} = \ddot{\theta}(t),\\ \dot{q_3} = \dot{x}(t),\\ \dot{q_4} = \ddot{x}(t), \end{cases}$$

and substituting into the equations (2.41) and (2.37), we obtain:

$$\begin{cases} \dot{q_1} = q_2, & (2.43)\\ \dot{q_2} = -2\zeta_\theta\omega_\theta q2 - \omega_\theta^2 q_1 + (K_\theta\omega_\theta^2\theta_{max})u_\theta, & (2.44)\\ \dot{q_3} = q_4, & (2.45)\\ \dot{q_4} = -C_x q_4 + g q_1. & (2.46) \end{cases}$$

Finally, putting these equations in matrix form, we have the subsystem (θ,x) with

the following representation:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_\theta^2 & -2\zeta_\theta\omega_\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \\ g & 0 & 0 & -C_x \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + \begin{bmatrix} 0 \\ K_\theta\omega_\theta^2\theta_{max} \\ 0 \\ 0 \end{bmatrix} u_\theta, \tag{2.47}$$

$$x(t) = [0\ 0\ 1\ 0] \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + [0]\ u_\theta. \tag{2.48}$$

2. **Subsystem (ϕ,y)**

Choosing the state variables:

$$\begin{cases} q_1 = \phi(t), \\ q_2 = \dot{\phi}(t), \\ q_3 = y(t), \\ q_4 = \dot{y}(t), \end{cases} \Rightarrow \begin{cases} \dot{q_1} = \dot{\phi}(t), \\ \dot{q_2} = \ddot{\phi}(t), \\ \dot{q_3} = \dot{y}(t), \\ \dot{q_4} = \ddot{y}(t), \end{cases}$$

and substituting into the equations (2.40) and (2.38), we obtain:

$$\begin{cases} \dot{q_1} = q_2, & (2.49) \\ \dot{q_2} = -2\zeta_{\omega\phi}q2 - \omega_\phi^2 q_1 + (K_\phi\omega_\phi^2\phi_{max})u_\phi, & (2.50) \\ \dot{q_3} = q_4, & (2.51) \\ \dot{q_4} = -C_y q_4 - g q_1. & (2.52) \end{cases}$$

Finally, putting these equations in matrix form, we have the subsystem (ϕ,y) with the following representation:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\omega_\phi^2 & -2\zeta_\phi\omega_\phi & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -g & 0 & 0 & -C_y \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + \begin{bmatrix} 0 \\ K_\phi\omega_\phi^2\phi_{max} \\ 0 \\ 0 \end{bmatrix} u_\phi, \tag{2.53}$$

$$y(t) = [0\ 0\ 1\ 0] \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + [0]\ u_\phi. \tag{2.54}$$

3. **Subsystem (z)**

Choosing the state variables:

$$\begin{cases} q_1 = z(t), \\ q_2 = \dot{z}(t), \end{cases} \Rightarrow \begin{cases} \dot{q}_1 = \dot{z}(t), \\ \dot{q}_2 = \ddot{z}(t), \end{cases}$$

and substituting in the equation (2.39), we obtain:

$$\begin{cases} \dot{q}_1 = q_2, & (2.55) \\ \dot{q}_2 = \left(\frac{K_{\dot{z}}\dot{z}_{max}}{\tau_{\dot{z}}}\right)u_{\dot{z}} - \left(\frac{1}{\tau_{\dot{z}}}\right)q_2, & (2.56) \end{cases}$$

Finally, put in matrix form, we have the subsystem (z) with the following representation:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{\tau_{\dot{z}}} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{\dot{z}}\dot{z}_{max}}{\tau_{\dot{z}}} \end{bmatrix} u_{\dot{z}}, \quad (2.57)$$

$$z(t) = [1\ 0] \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + [0]\ u_{\dot{z}}. \quad (2.58)$$

4. **Subsystem (ψ)**

Choosing the state variables:

$$\begin{cases} q_1 = \psi(t), \\ q_2 = \dot{\psi}(t), \end{cases} \Rightarrow \begin{cases} \dot{q}_1 = \dot{\psi}(t), \\ \dot{q}_2 = \ddot{\psi}(t), \end{cases}$$

and substituting in the equation (2.39), we obtain:

$$\begin{cases} \dot{q}_1 = q_2, & (2.59) \\ \dot{q}_2 = \left(\frac{K_{\dot{\psi}}\dot{\psi}_{max}}{\tau_{\dot{\psi}}}\right)u_{\dot{\psi}} - \left(\frac{1}{\tau_{\dot{\psi}}}\right)q_2, & (2.60) \end{cases}$$

Finally, put in matrix form, we have the subsystem (ψ) with the following representation:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{\tau_{\dot{\psi}}} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{\dot{\psi}}\dot{\psi}_{max}}{\tau_{\dot{\psi}}} \end{bmatrix} u_{\dot{\psi}}, \quad (2.61)$$

$$\psi(t) = [1\ 0] \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + [0]\ u_{\dot{\psi}}. \quad (2.62)$$

2.5 Simulated Model

The simulations will be carried out with the linear controller designed from the following model, which considers the non-linearities present in the equations in x and y, in order to obtain more realistic results:

$$\begin{cases} \ddot{x} = g(sen\psi sin\phi + cos\psi cos\phi sin\theta) - C_x\dot{x}, \\ \ddot{y} = g(-cos\psi sin\phi + sin\psi cos\phi sin\theta) - C_y\dot{y}, \\ \ddot{z} = \left(\frac{K_{\dot{z}}\dot{z}_{max}}{\tau_{\dot{z}}}\right)u_{\dot{z}} - \left(\frac{1}{\tau_{\dot{z}}}\right)\dot{z}, \\ \ddot{\phi} = K_\phi\omega_\phi^2\phi_{max}u_\phi - 2\zeta_\phi\omega_\phi\dot{\phi} - \omega_\phi^2\phi, \\ \ddot{\theta} = K_\theta\omega_\theta^2\theta_{max}u_\theta - 2\zeta_\theta\omega_\theta\dot{\theta} - \omega_\theta^2\theta, \\ \ddot{\psi} = \left(\frac{K_{\dot{\psi}}\dot{\psi}_{max}}{\tau_{\dot{\psi}}}\right)u_{\dot{\psi}} - \left(\frac{1}{\tau_{\dot{\psi}}}\right)\dot{\psi}. \end{cases} \tag{2.63}$$

Regarding the model parameters (2.63), according to the manufacturer's manual (PISKORSKI NICOLAS BRULEZ, 2012), there is the possibility of configuring the limiting variables ($\theta_{máx}$, ϕ_{max}, $\dot{\psi}_{max}$ and $\dot{z}_{max}$) in the following ranges: $0.2 \leq \dot{z}_{max} \leq 2.0\ m/s$, $0.7 \leq \dot{\psi}_{max} \leq 6.11\ rad/s$ and $0{,}0 \leq (\theta_{max}, \phi_{max}) \leq 0.52\ rad$.

The values chosen for this work are $\dot{z}_{máx} = 1$, $\dot{\psi}_{máx} = 1.74$ and $(\theta_{máx}, \phi_{max}) = 0.26$. Furthermore, in (SANTANA, 2016), the other parameters were determined for indoor environments through trial and error, reaching the values determined in Table 1.

Tab. 1 – UAV AR.Drone 2.0 parameters.

$K_\phi = \begin{cases} 1.0\ se\ \lvert u_\phi\rvert \geq 0.5 \\ 2.0\ se\ \lvert u_\phi\rvert < 0.5 \end{cases}$	$K_\theta = \begin{cases} 1.0\ se\ \lvert u_\theta\rvert \geq 0.5 \\ 2.0\ se\ \lvert u_\theta\rvert < 0.5 \end{cases}$
$K_{\dot{z}} = 1.3$ e $K_{\dot{\psi}} = 1.0$	$m = 0.380\ [kg]$
$\phi_{max} = 0.26\ [rad]$	$\theta_{max} = 0.26\ [rad]$
$\dot{z}_{max} = 1.0\ [m/s]$	$\dot{\psi}_{max} = 1.74\ [rad/s]$
$\omega_\phi = 4.47\ [Hz]$	$\omega_\theta = 4.47\ [Hz]$
$\zeta_\phi = 0.5$	$\zeta_\theta = 0.5$
$\tau_{\dot{z}} = 0.4\ [s]$	$\tau_{\dot{\psi}} = 0.3\ [s]$
$C_x = 0.3\ [1/s]$	$C_y = 0.1\ [1/s]$

3 LQR CONTROLLER

After determining the simplified linear model of AR.Drone 2.0, it is desired to design an optimal LQR controller via LMI with allocation of closed-loop poles in a given convex region of the s plane. Then, an LQR controller design with pole allocation is carried out taking into account the polytopic uncertainties of some model parameters. This chapter, then, presents a brief study of the aforementioned control techniques.

3.1 State Feedback Control

In classical control designs, the strategy for meeting performance criteria can be based on closed-loop modal dominance, that is, the controller design is made in such a way that the dominant poles of the system belong to a region with a coefficient of satisfactory damping ζ and undamped natural frequency ω_n (OGATA, 2014). Alternatively, state feedback design allows the designer to specify all closed-loop poles of the system, as long as all state variables can be measured or estimated. To use this control strategy, all system state variables must be measurable or, alternatively, state observers must be inserted to estimate them. Furthermore, this methodology requires that the system in question is controllable.

Let the control system be described by the equations in state space:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{3.1}$$

$$y(t) = Cx(t) + Du(t), \tag{3.2}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $y(t) \in \mathbb{R}^q$ is the output signal vector, $u(t) \in \mathbb{R}^m$ is the control signal vector, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{q\times n}$ and $D \in \mathbb{R}^{q\times m}$. The state feedback control signal is given by:

$$u(t) = Kx(t), \tag{3.3}$$

where $K \in \mathbb{R}^{m\times n}$ is called the state feedback gain matrix. Replacing (3.3) in (3.1), we obtain:

$$\dot{x}(t) = (A + BK)x(t), \tag{3.4}$$

$$y(t) = (C + DK)x(t). \tag{3.5}$$

The solution to the dynamic equation is given by:

$$x(t) = e^{(A+BK)t}x(0), \tag{3.6}$$

where $x(0)$ is the initial state vector of the system. The block diagram with this control strategy can be seen in Figure 18.

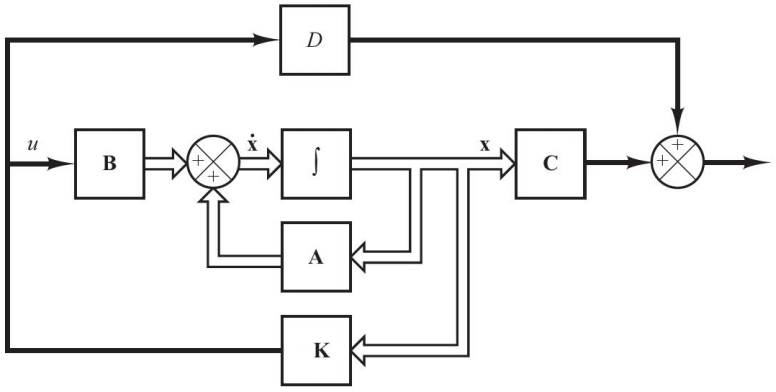


Fig. 18 – Block diagram of a control system with state feedback.

Source: Adapted from (OGATA, 2014).

The pole allocation can be carried out by different strategies, each of which, using a specific mathematical tool, will seek to determine the state feedback gain matrix K capable of taking the roots of the characteristic closed-loop polynomial to the locations desired. The allocation of the system's closed-loop poles is done in strategic positions in the s-plane in order to guarantee stability and performance, such as, for example, desired settling time and overshoot. State feedback controller design can also be done by minimizing error and control effort. This strategy called the optimal quadratic regulator problem is detailed in Section 3.3.

3.2 State Feedback with Full Action

The controller shown in Figure 18 focuses on rejecting disturbances, therefore, it is not concerned with ensuring good tracking for the system (FRANKLIN; POWELL; EMAMI-NAEINI, 2013). This becomes a problem, since control systems generally require both good disturbance rejection and good reference tracking. Therefore, in order to solve the problem of tracking the trajectory of the output signal with state feedback, a gain matrix K_r is used, since good command tracking is done through the correct introduction of the reference input in the system equations (FRANKLIN; POWELL; EMAMI-NAEINI, 2013). This gain K_r is called gain *feedforward* and the block diagram with its insertion is shown in Figure 19.

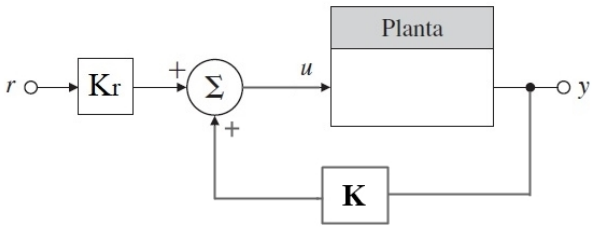


Fig. 19 – Control system with state and gain feedback *feedforward*.

Source: Adapted from (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

However, although the insertion of the gain matrix K_r results in a zero steady-state error for step input, this result is not robust, as changes in the plant parameters may cause the steady-state error to be different from zero (FRANKLIN; POWELL; EMAMI-NAEINI, 2013). Thus, the gain *feedforward* has limited use in solving control problems in which plant parameters are uncertain and/or change over time. Therefore, to achieve robust screening, integral action can be included in order to obtain zero steady-state error in the presence of modeling uncertainties or in the face of variations in plant parameters. The basic idea is to add a state that corresponds to the integral of the error signal, which will also be used in feedback, as shown in Figure 20.

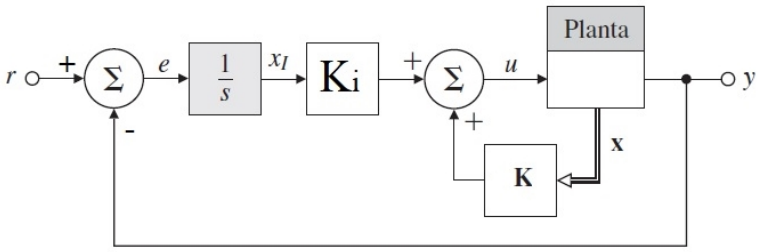


Fig. 20 – Control system with state feedback with integrator.

Source: Adapted from (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

When making this modification to the control system, the plant state vector is increased by adding the integral state $x_I(t) \in \mathbb{R}^p$, now being defined by $x_a(t) \in \mathbb{R}^{n+p}$, given by:

$$x_a(t) = \begin{bmatrix} x(t) \\ x_I(t) \end{bmatrix}. \tag{3.7}$$

As $\dot{x}_I(t) = e(t)$, where $e(t)$ is the system error signal, we have that:

$$\dot{x}_I(t) = r(t) - y(t),$$

$$\dot{x}_I(t) = r(t) - (Cx(t) + Du(t)). \tag{3.8}$$

Thus, the new state equations of the system are given by:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_I(t) \end{bmatrix} = \begin{bmatrix} A & 0_{n\times m} \\ -C & 0_{q\times q} \end{bmatrix} \begin{bmatrix} x(t) \\ x_I(t) \end{bmatrix} + \begin{bmatrix} B \\ -D \end{bmatrix} u(t) + \begin{bmatrix} 0_{n\times j} \\ I_{p\times j} \end{bmatrix} r(t), \tag{3.9}$$

$$y(t) = \begin{bmatrix} C & 0_{q\times q} \end{bmatrix} \begin{bmatrix} x(t) \\ x_I(t) \end{bmatrix} + Du(t), \tag{3.10}$$

where $x(t) \in \mathbb{R}^n$, $x_I(t) \in \mathbb{R}^p$, $y(t) \in \mathbb{R}^q$, $u(t) \in \mathbb{R}^m$ and $r(t) \in \mathbb{R}^j$ is the reference signal vector. The matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{q\times n}$, $D \in \mathbb{R}^{q\times m}$ and 0 and I are, respectively, the null and identity with dimensions described in the equations (3.9) and (3.10).

In closed loop, we have:

$$\begin{cases} \begin{bmatrix} \dot{x}(t) \\ \dot{x}_I(t) \end{bmatrix} = \begin{bmatrix} A + BK & BK_i \\ -(C + DK) & -DK_i \end{bmatrix} \begin{bmatrix} x(t) \\ x_I(t) \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} r(t), \\ \\ y(t) = \begin{bmatrix} (C + DK) & DK_i \end{bmatrix} \begin{bmatrix} x(t) \\ x_I(t) \end{bmatrix}. \end{cases} \tag{3.11}$$

3.3 Optimal Quadratic Linear Control

The state feedback control strategy can be carried out by solving an optimization problem, instead of selecting the closed-loop eigenvalues of the system as is the case in the project via pole allocation. This becomes useful when you want to balance system performance with the magnitudes of control signals (ÅSTRÖM; MURRAY, 2010). Another

favorable point, according to (OGATA, 2014), is that the optimization process provides a systematic way of calculating the state feedback gain matrix K, which facilitates not only the creation of numerical algorithms, but also the control projects for systems with multiple inputs and multiple outputs (MIMO). This proves to be an advantage in relation to the pole allocation problem for MIMO systems, as this, in addition to having multiple solutions, with different specifications of the controlled signals and limitations of the actuators, the relationship between eigenvalues and response requirements is not trivial. and intuitive.

Therefore, following this optimization methodology to design a state feedback controller, it is clear that in the literature the linear quadratic regulator, known as LQR, is one of the most common optimization problems (ÅSTRÖM; MURRAY, 2010). In it, the strategy is to find an optimal K feedback gain that promotes a balance between system performance and control effort. Let the system be described by the equations (3.1) and (3.2), we want to minimize the following cost function, here called the quadratic performance index J_∞:

$$J_\infty = \int_0^\infty \Big[x^T(t)Qx(t) + u^T(t)Ru(t)\Big]dt, \tag{3.12}$$

where $Q \succeq 0$ ($Q \in \mathbb{R}^{n\times n}$) and $R \succ 0$ ($R \in \mathbb{R}^{m\times m}$) are, respectively, the weighting matrices of the system state vector and the control signal. And the terms $x^T(t)Qx(t)$ and $u^T(t)Ru(t)$ represent the distance from the state to the origin and the cost of the control signal, respectively.

The linear quadratic problem can be described by the following optimization problem:

$$\min_u \quad J_\infty = \int_0^\infty \Big[x^T(t)Qx(t) + u^T(t)Ru(t)\Big]dt, \tag{3.13}$$

subject to

$$\dot{x}(t) = Ax(t) + Bu(t),$$

$$x(0) = x_0.$$

Therefore, to solve the problem (3.13) it is first necessary to define the weighting matrices Q and R. The matrix Q specifies the importance relative to the components of the state vector and the matrix R, the importance relative to the control cost. The choice of these weighting matrices can be done by trial and error by analyzing the simulations.

Once Q and R are chosen, the control law that minimizes the index J_∞ is defined by state feedback, where the gain matrix K is determined by (OGATA, 2014):

$$K = -R^{-1}B^T P, \tag{3.14}$$

where $P = P^T \in \mathbb{R}^{n\times n}$ is a positive definite matrix ($P \succ 0$) that satisfies the algebraic equation of *Riccati*:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \tag{3.15}$$

3.4 LQR control via LMI

The solution to the problem (3.13) can also be obtained by solving the convex optimization problem (ARAÚJO, 2020):

$$\min_{W_1 \succ 0, W_2} \quad \gamma, \tag{3.16}$$

subject to

$$\begin{bmatrix} \gamma I & x(0)^T \\ x(0) & W_1 \end{bmatrix} \succ 0, \tag{3.17}$$

$$\begin{bmatrix} AW_1 + W_1A^T + BW_2 + W_2^T B^T & W_1 Q^{\frac{1}{2}} & W_2^T R^{\frac{1}{2}} \\ Q^{\frac{1}{2}} W_1 & -I & 0 \\ R^{\frac{1}{2}} W_2 & 0 & -I \end{bmatrix} \prec 0, \tag{3.18}$$

where $W_1 \succ 0$, $W_1 \in \mathbb{R}^{n\times n}$, $W_2 \in \mathbb{R}^{m\times n}$ and $\gamma \succ 0$. In the solution, the gain K is given by:

$$K = W_2 W_1^{-1}, \tag{3.19}$$

allowing to scale K.

3.5 LQR Control with Pole Allocation

Since the LQR problem (3.16) is described through the LMIs (3.17) and (3.18), it is also possible to restrict a region in the plane s where the closed-loop poles must be allocated. That is, in addition to minimizing the index J_∞, which promotes the balance between system performance and control effort, it is possible to determine a feedback gain K that allocates the closed-loop poles in a specific region of the s plane, related to good specifications of the transient response, such as maximum overshoot and settling time.

To this end, in (ARAÚJO, 2020) it is demonstrated that for the allocation of poles within a vertical range delimited by α and β (Figure 21), it is necessary that the matrices $W_1 \in \mathbb{R}^{n\times n}$ and $W_2 \in \mathbb{R}^{m\times n}$, which define $K = W_2 W_1^{-1}$, satisfy the following LMIs:

$$\begin{cases} W_1 \succ 0, \\ AW_1 + W_1A^T + BW_2 + W_2^T B^T + 2\alpha W_1 \prec 0, \\ -AW_1 - W_1A^T - BW_2 - W_2^T B^T - 2\beta W_1 \prec 0. \end{cases} \tag{3.20}$$

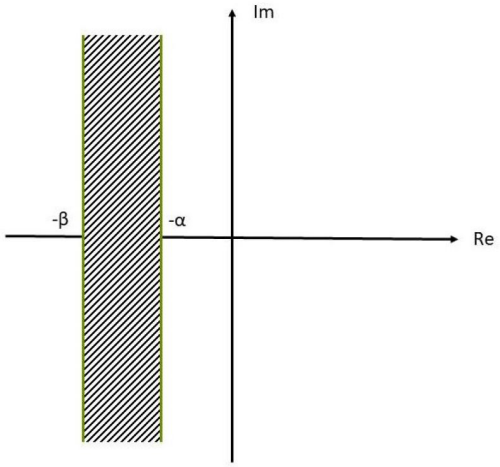


Fig. 21 – Vertical range of plane s.

Source: (ARAÚJO, 2020).

Furthermore, it is also possible to allocate the closed-loop poles in a conical sector defined by straight lines with angle ρ as shown in Figure 22. The angle ρ satisfies the relationship $\rho = arccos(\zeta)$, where ζ is the damping coefficient associated with the pair of complex poles on these straight lines. For this, in (CHILALI; GAHINET; APKARIAN, 1999) it is demonstrated that $W_1 \in \mathbb{R}^{n\times n}$ and $W_2 \in \mathbb{R}^{m\times n}$ must satisfy the following LMI

$$\begin{bmatrix} sin(\rho)[AW_1 + W_1A^T + BW_2 + W_2^TB^T] & cos(\rho)[AW_1 - W_1A^T + BW_2 + W_2^TB^T] \\ cos(\rho)[-AW_1 + W_1A^T - BW_2 + W_2^TB^T] & sin(\rho)[AW_1 + W_1A^T + BW_2 + W_2^TB^T] \end{bmatrix} \prec 0. \quad (3.21)$$

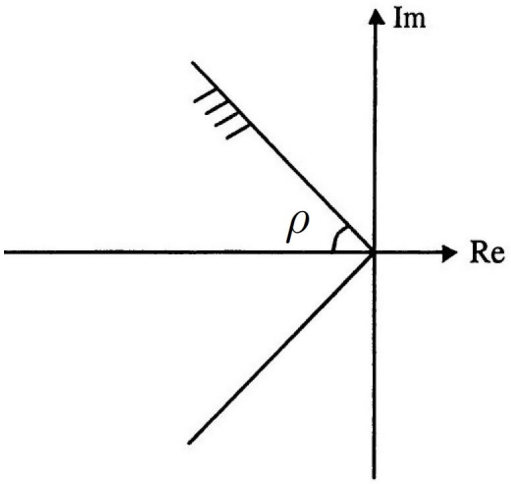


Fig. 22 – Conic Sector of plane s.

Source: (ARAÚJO, 2020).

3.6 Robust LQR Control

Dynamic systems may present uncertainties in the parameters of the mathematical model or these parameters may vary over time with slow dynamics compared to the system's time constants. In these cases, it is necessary to design a robust controller, that is, one that guarantees the stability and performance of the system in the face of parametric uncertainties. In this work, the uncertainties considered are of the interval type. In modeling the AR.Drone, the time constants $\tau_{\dot{z}}$ and $\tau_{\dot{\psi}}$ and the translational drag coefficients C_x and C_y will be treated as parametric uncertainties. This is because there is an inaccuracy in determining these parameters. Furthermore, as the coefficients C_x and C_y represent the dynamics of air resistance during flight, in outdoor environments, these drag coefficients tend to be higher than the values determined for indoor environments, shown in Table 1.

Let the continuous-time linear system be subject to polytopic parametric uncertainties:

$$S_c : \begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}, \tag{3.22}$$

where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^q$, $u \in \mathbb{R}^m$, $u(t) = Kx(t)$, $A \in \mathcal{D}_A$ and $B \in \mathcal{D}_B$. $\mathcal{D}_A$ and $\mathcal{D}_B$ are sets given by:

$$\mathcal{D}_A = \left\{ A \in \mathbb{R}^{n\times n} : A = \sum_{i=1}^{N} \alpha_i A_i, \alpha \in \Lambda_N \right\}, \tag{3.23}$$
$$with \;\; \Lambda_N = \left\{ \alpha \in \mathbb{R}^N : \sum_{i=1}^{N} \alpha_i = 1, \alpha_i \geq 0, i = 1,2,...,N \right\},$$

It is

$$\mathcal{D}_B = \left\{ B \in \mathbb{R}^{n\times m} : B = \sum_{j=1}^{M} \beta_j B_j, \beta \in \Lambda_M \right\}, \tag{3.24}$$
$$with \;\; \Lambda_M = \left\{ \beta \in \mathbb{R}^M : \sum_{j=1}^{M} \beta_j = 1, \beta_j \geq 0, j = 1,2,...,M \right\}.$$

The uncertain system S_c is asymptotically stable in closed loop state feedback if

$$A_i W_1 + W_1 A_i^T + B_j W_2 + W_2^T B_j^T \prec 0, \forall i = 1,2,...,N, \quad e \;\; \forall j = 1,2,...,M, \tag{3.25}$$

and the robust gain is given by

$$K = W_2 W_1^{-1}.$$

This theorem can be demonstrated by multiplying by α_i each LMI of the equation (3.25) for a fixed j, and subsequently, adding them all up, we have:

$$\left(\sum_{i=1}^{N}\alpha_i A_i\right)W_1 + W_1\left(\sum_{i=1}^{N}\alpha_i A_i\right)^T + B_j W_2 + W_2^T B_j^T \prec 0.$$

Then, carrying out the same procedure for all j, by multiplying each of these M LMIs by β_j and adding them together, we have:

$$\left(\sum_{i=1}^{N}\alpha_i A_i\right)W_1 + W_1\left(\sum_{i=1}^{N}\alpha_i A_i\right)^T + \left(\sum_{j=1}^{M}\beta_j B_j\right)W_2 + W_2^T\left(\sum_{j=1}^{M}\beta_j B_j\right)^T \prec 0.$$

This proves the stability of the infinite models generated by the convex combination of the matrices A_i and B_j.

The Robust LQR problem can be defined as:

$$\min_u \quad J_\infty = \int_0^\infty [x^T(t)Qx(t) \quad u^T(t)Ru(t)]dt,$$

subject to

$$\dot{x}(t) = Ax(t) + Bu(t), A \in \mathcal{D}_A \; e \; B \in \mathcal{D}_B$$

$$x(0) = x_0,$$

with $Q \succeq 0$ and $R \succ 0$.

The solution to this convex problem can be obtained by solving the following optimization problem:

$$\min_{W_1 \succ 0, W_2} \quad \gamma, \tag{3.26}$$

subject to

$$\begin{bmatrix} \gamma I & x(0)^T \\ x(0) & W_1 \end{bmatrix} \succ 0, \tag{3.27}$$

$$\begin{bmatrix} A_i W_1 + W_1 A_i^T + B_j W_2 + W_2^T B_j^T & W_1 Q^{\frac{1}{2}} & W_2^T R^{\frac{1}{2}} \\ Q^{\frac{1}{2}} W_1 & -I & 0 \\ R^{\frac{1}{2}} W_2 & 0 & -I \end{bmatrix} \prec 0, \tag{3.28}$$

$\forall i = 1,2,...,N \; e \; \forall j = 1,2,...,M$. The gain that stabilizes the uncertain system is determined by

$$K = W_2 W_1^{-1}. \tag{3.29}$$

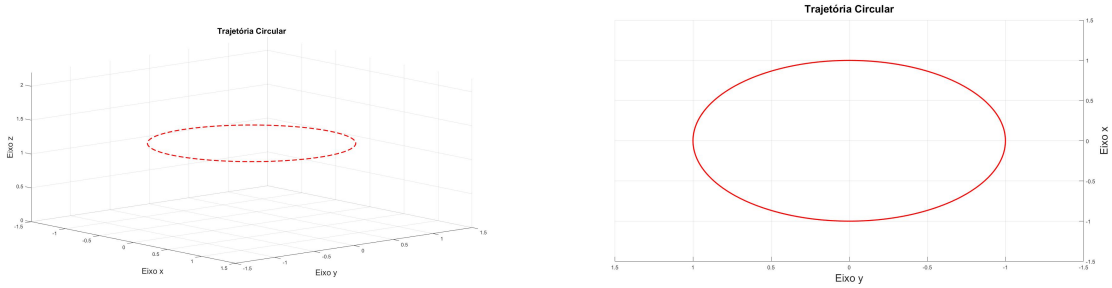
4 CONTROLLERS DESIGN

The Figure 14 shows the importance of circle and inclined figure-eight trajectories for verifying the performance of the designed controller is discussed. The reference trajectories are shown in Figures 23 and 24. For the circular trajectory, the reference vector $\mathbf{r}(t) = [r_x \ r_y \ r_z \ r_\psi]$ is given by:

$$\mathbf{r}(t) = [sin(0.8t) \quad cos(0.8t) \quad 1.2 \quad 0]^T, \tag{4.1}$$

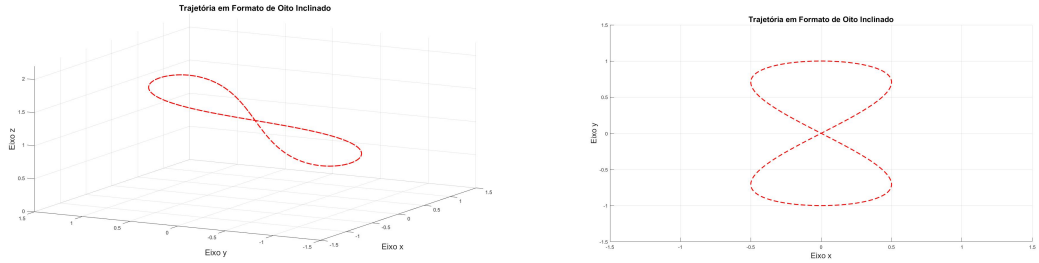
and the eight-shaped trajectory is given by:

$$\mathbf{r}(t) = [0.5sin(0.8t) \quad sin(0.4t) \quad 1.2 + 0.5sin(0.4t) \quad -\frac{\pi}{6}sin(0{,}4t)]^T, \tag{4.2}$$



(a) Trajectory seen from the side. (b) Trajectory seen from above.

Fig. 23 – Circular Reference Trajectory.



(a) Trajectory seen from the side. (b) Trajectory seen from above.

Fig. 24 – Reference Tilt-Eight Shape Trajectory.

The frequencies used in the trajectories were chosen in such a way that the dynamic behavior of the *drone* is not far from that defined by the simplified model considered.

4.1 LQR control with allocation

For the LQR controller design with pole allocation, the result of trial and error tuning of the Q and R matrices, the limits of the vertical range and the sector in the pole allocation are shown below:

Subsystem(θ,x):

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1000 \end{bmatrix} \\ R = 3, \\ \alpha = 0.05, \ \ \beta = 22, \ \ \rho = 85°, \\ K = [-10.46 \ \ -1.26 \ \ -14.81 \ \ -5.46], \\ K_i = 18.81, \end{cases}$$

Subsystem (z):

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 120 \end{bmatrix} \\ R = 1, \\ \alpha = 0.0, \ \ \beta = 3, \ \ \rho = 43°, \\ K = [-4.74 \ \ -1.24], \\ K_i = 4.41. \end{cases} \tag{4.3}$$

Subsystem(ϕ,y):

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 240 \end{bmatrix} \\ R = 0.1, \\ \alpha = 0.05, \ \ \beta = 22, \ \ \rho = 85°, \\ K = [-29.65 \ \ -3.12 \ 34.78 \ \ -13.76], \\ K_i = -42.09, \end{cases}$$

Subsystem (ψ):

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \\ R = 24.5, \\ \alpha = 0.4 \ \ \beta = 4.0, \ \ \rho = 60°, \\ K = [-0.65 \ \ -0.20], \\ K_i = 0.25. \end{cases} \tag{4.4}$$

This allocation region was chosen so that the closed-loop poles result in a relatively well-damped and fast transient response.

Using these values in the project, the three-dimensional trajectory of the UAV is analyzed, in addition to observing the individual output of each subsystem, the control signals u_θ, u_ϕ, $u_{\dot{\psi}}$, $u_{\dot{z}}$, taking care to avoid saturation of these, and the closed-loop poles of the system. The result of this analysis is presented in the following sections.

4.1.1 Reference Tracking

To analyze the vehicle's trajectory tracking, the position vectors x, y and z were collected from the simulation in the SIMULINK environment. The results shown in Figures 25 to 28 were obtained for the circular trajectory. For the inclined eight-shaped trajectory, the results are shown in Figures 29 to 32.

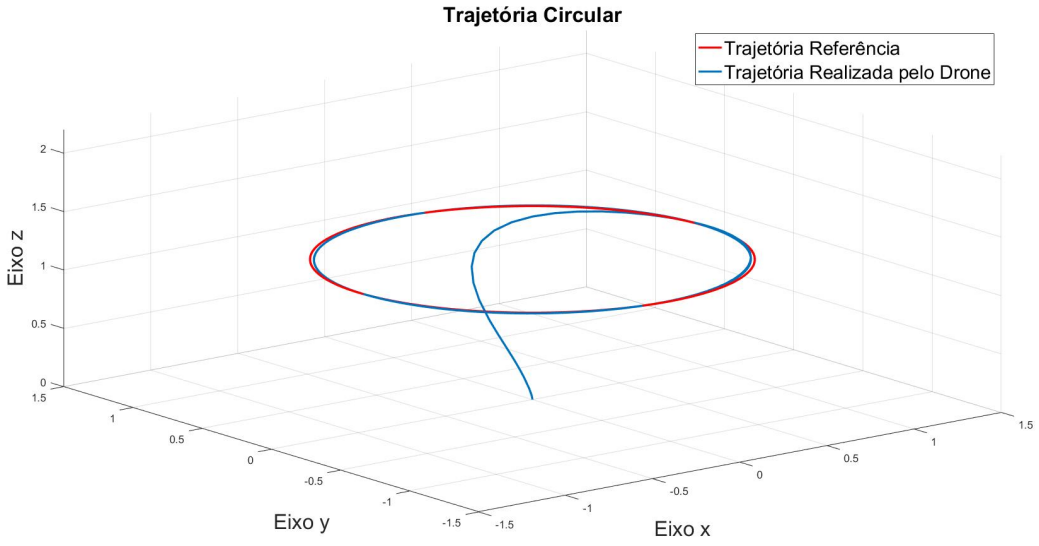


Fig. 25 – Circular Trajectory Simulation 01.

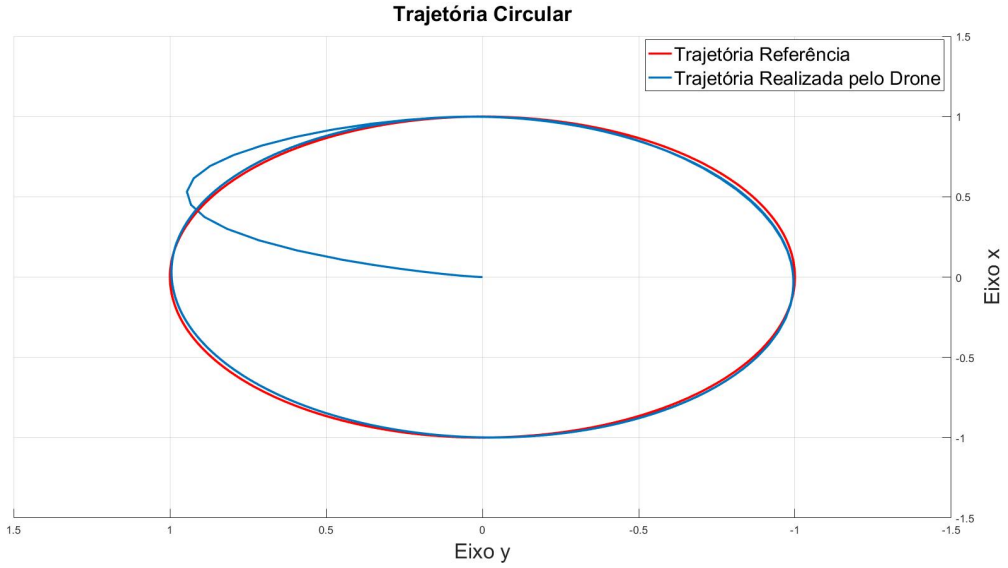


Fig. 26 – Circular Trajectory Simulation 02.

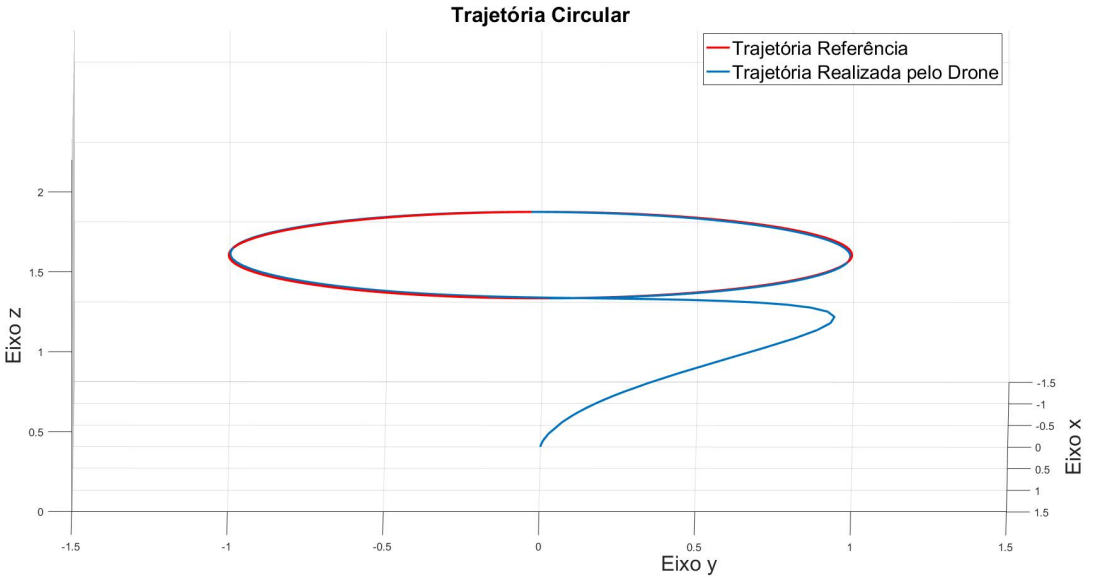


Fig. 27 – Circular Trajectory Simulation 03.

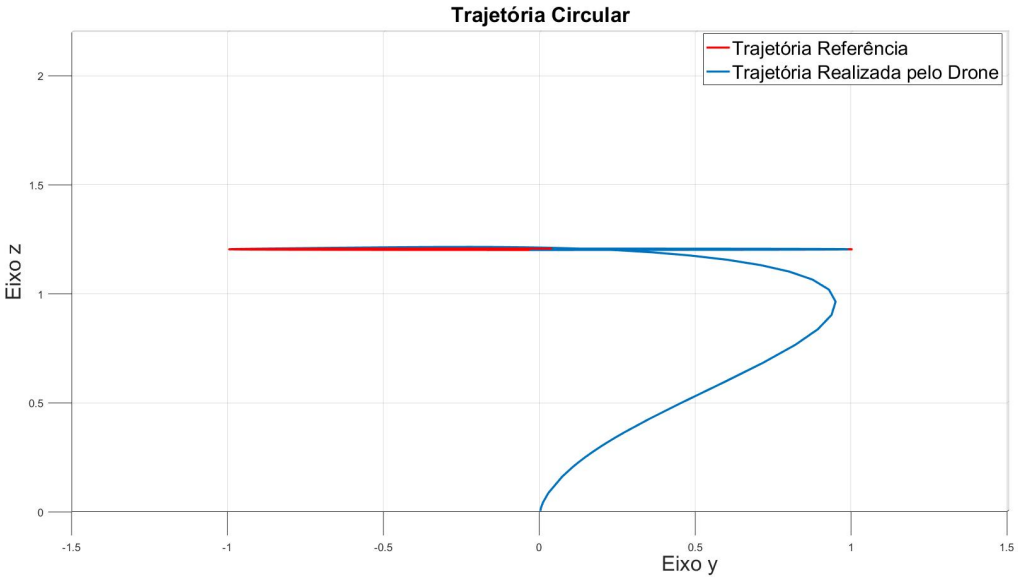


Fig. 28 – Circular Trajectory Simulation 04.

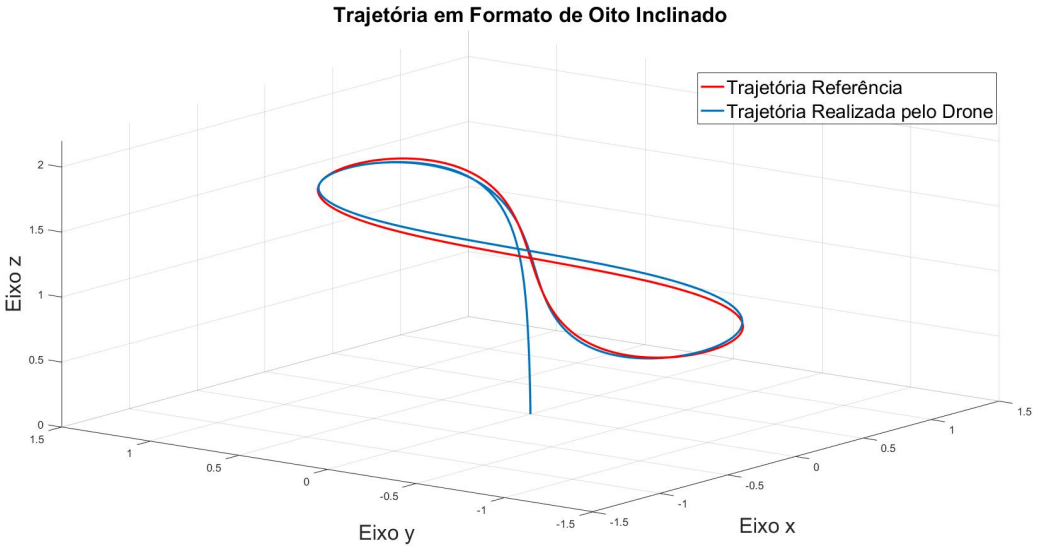


Fig. 29 – Trajectory Simulation in Eight Shape 01.

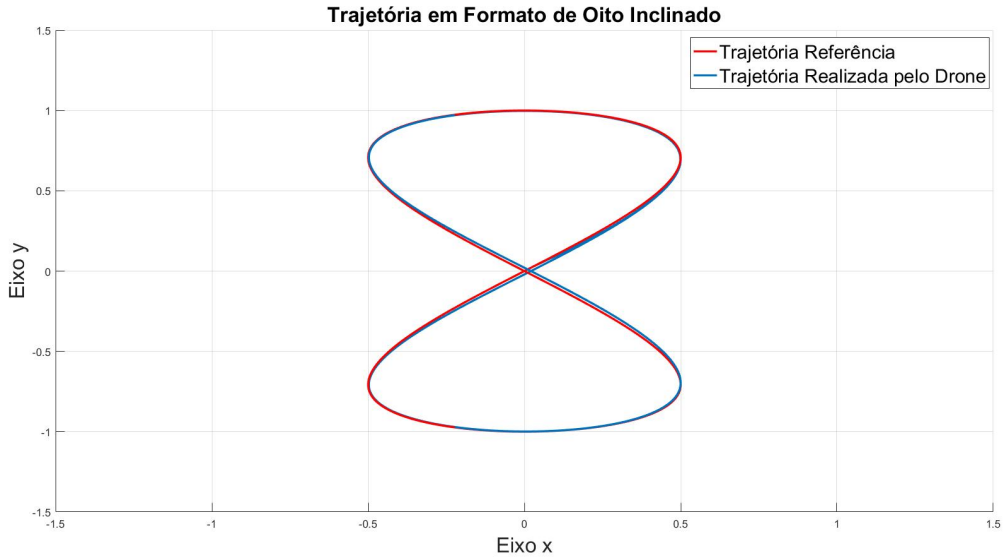


Fig. 30 – Trajectory Simulation in Eight Shape 02.

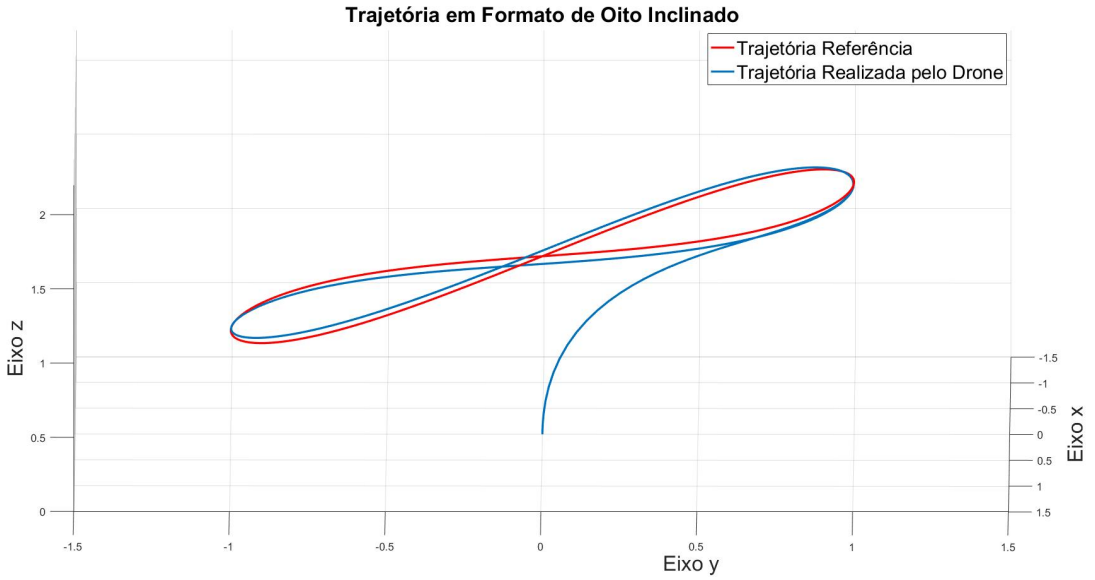


Fig. 31 – Trajectory Simulation in Eight Shape 03.

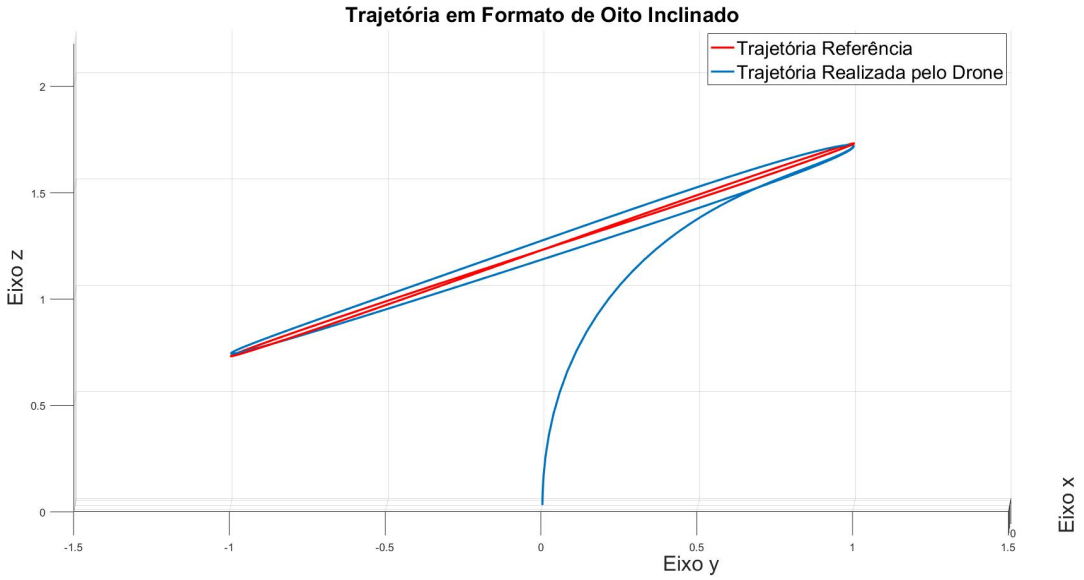


Fig. 32 – Trajectory Simulation in Eight Shape 04.

The Q and R matrices are tuned based on the following weighting logic:

- Increasing the terms of the matrix Q implies making the states of the system go to the reference, penalizing the speed of its movement. While reducing the terms of the matrix implies forcing the states to go to the reference with greater speed;
- On the other side of the weighting, the R matrix is designed to avoid control signals that the actuators cannot support. Therefore, increasing the terms of the R matrix implies forcing the system to reach the reference with the lowest possible control signal cost. While reducing the terms of the matrix implies reaching the reference allowing high control signals.

Based on this, the circular trajectory was first chosen, and the matrices were tuned by graphically comparing the individual output of each subsystem with its respective reference signal (4.1), and analyzing the associated control signal, avoiding saturation and thus obeying the limits determined by the equations (2.5), (2.6), (2.8) and (2.7). Once each subsystem presented a satisfactory response with the tuned controller, the three-dimensional trajectory carried out by the *drone* was analyzed, thus validating the control of the plant. Subsequently, the region in which the closed-loop poles are located was mapped in each subsystem.

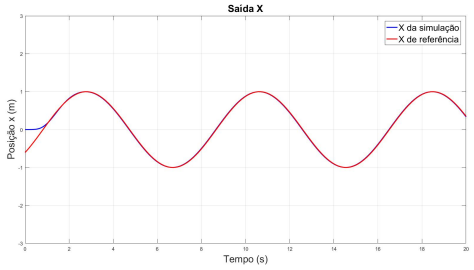
When analyzing the simulation of the circular trajectory, it is clear that the system presented a good reference tracking, with discrete deviations of the outputs x, y and z observed in Figures 26 a 28. In the figure-of-eight trajectory, the system also showed good results, especially for the x and y outputs, since in Figure 30 the deviations are also almost imperceptible. However, at the output z of this trajectory, we observe - see Figure 32 - a steady-state error on the axis z of the trajectory. This occurred due to the limit that the control signal $u_{\dot{z}}$ has, in which, resetting this error in steady state is only possible by saturating the control signal $u_{\dot{z}}$. Furthermore, it was also observed, when tuning the Q and R matrices and the tuning parameters for pole allocation, that there are values that present a good response for the circular trajectory and other values for the inclined figure-eight trajectory. However, it is necessary to choose, through fine adjustment, a dimension that accepts both trajectories in an acceptable way. In this work, the parameters chosen are those presented in (4.3) and (4.4). These results, for example, are capable of nullifying the steady-state error of subsystem z for the circular trajectory (Figure 28).

4.1.2 Outputs

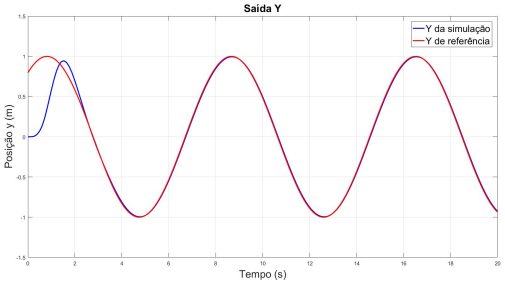
The tuning of the controllers is carried out separately for each subsystem, in which the reference segment considered, a priori, are the individual outputs of each subsystem, represented by the reference vector $\mathbf{r}(t)$ (equations (4.1) and (4.2)). In this subsection, the

results of these outputs, together with the reference signal are shown in Figures 33a to 36b.

Circular Trajectory:

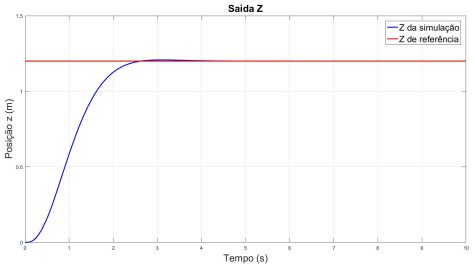


(a) Output x.

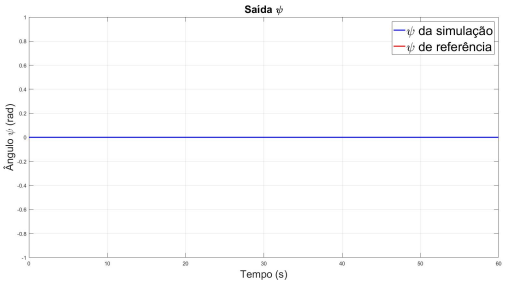


(b) Output y.

Fig. 33 – Outputs x and y of system 01.



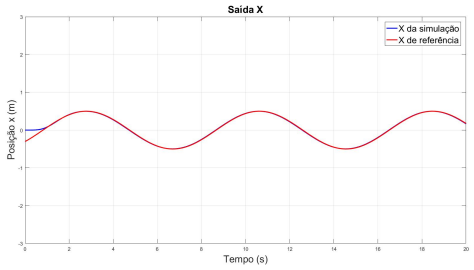
(a) Output z.



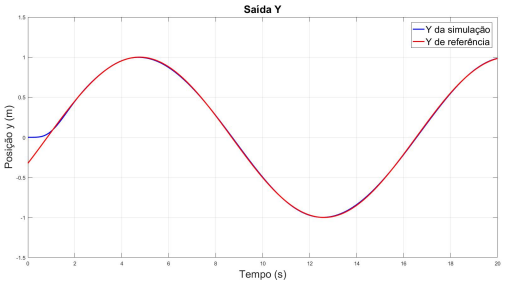
(b) Output ψ.

Fig. 34 – Outputs z and ψ of system 01.

Eight Shaped Trajectory:

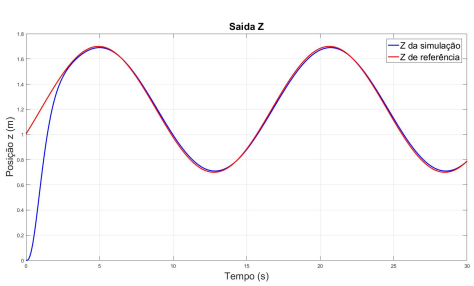


(a) Output x.

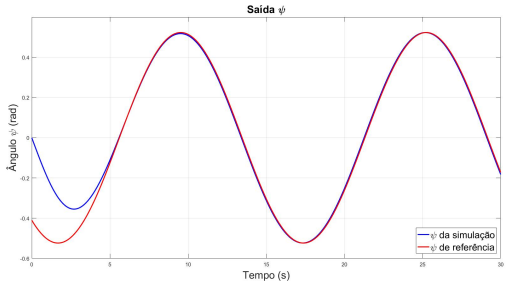


(b) Output y.

Fig. 35 – Outputs x and y of system 02.



(a) Output z.



(b) Output ψ.

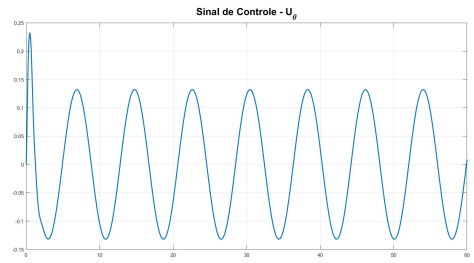
Fig. 36 – Outputs z and ψ of system 02.

4.1.3 Control Signals

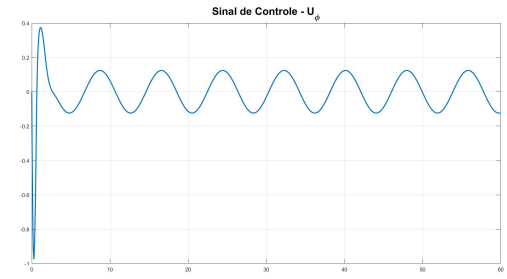
As discussed in the 2.2 section, the control signals u_θ, u_ϕ, $u_{\dot{\psi}}$, $u_{\dot{z}}$ are normalized, and represent the percentage of the limiting parameters (θ_{max}, ϕ_{max}, $\dot{\psi}_{max}$, $\dot{z}_{max}$) desired in the system dynamics (equations (2.5) to (2.8)). That is, the module of the values of u_θ, u_ϕ, $u_{\dot{\psi}}$, $u_{\dot{z}}$ cannot exceed 1. The unit value represents using 100% the maximum value allowed by the autopilot.

Therefore, in this project, it is necessary to monitor the control signals during tuning, in order to avoid saturation and compromising the stability of the system. The results of these signals for the designed controller are shown in the following figures.

Circular Trajectory:

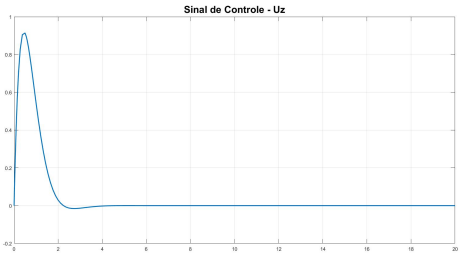


(a) Control Signal u_θ.

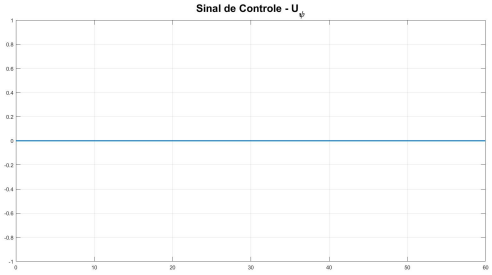


(b) Control Signal u_ϕ.

Fig. 37 – Control Signals u_θ and u_ϕ.



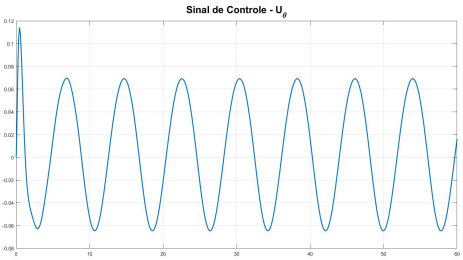
(a) Control Signal $u_{\dot{z}}$.



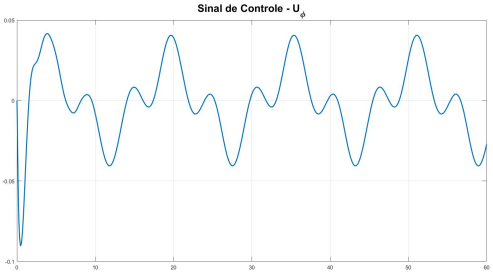
(b) Control Signal $u_{\dot{\psi}}$.

Fig. 38 – Control Signals $u_{\dot{z}}$ and $u_{\dot{\psi}}$.

Eight Shaped Trajectory:

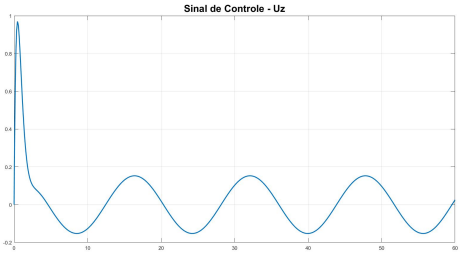


(a) Control Signal u_θ.

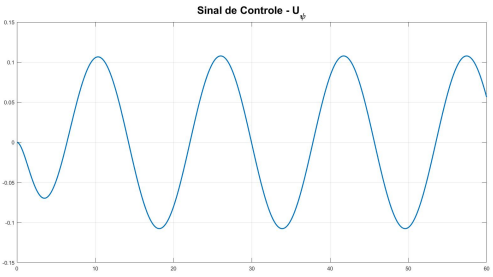


(b) Control Signal u_ϕ.

Fig. 39 – Control Signals u_θ and u_ϕ.



(a) Control Signal $u_{\dot{z}}$.



(b) Control Signal $u_{\dot{\psi}}$.

Fig. 40 – Control Signals $u_{\dot{z}}$ and $u_{\dot{\psi}}$.

It is possible to observe in Figure 40a how the control signal $u_{\dot{z}}$ comes close to the limit when the UAV climbs. Therefore, forcing a higher value in the control signal - to guarantee more accurate trajectory tracking - would imply saturation of $u_{\dot{z}}$.

4.1.4 Closed-loop poles

For this project, the values of the system's closed-loop poles are (see Figure 41):

$$\begin{cases} s_{z1} = -2.9243, \\ s_{z2} = -1.7946 + j1.2982, \\ s_{z3} = -1.7946 - j1.2982, \end{cases} \quad \begin{cases} s_{\psi 1} = -3.5217, \\ s_{\psi 2} = -0.4781 + j0.4227, \\ s_{\psi 3} = -0.4781 - j0.4227, \end{cases} \tag{4.5}$$

$$\begin{cases} s_{\phi 1} = -17.3997, \\ s_{\phi 2} = -4.8675, \\ s_{\phi 3} = -3.6268, \\ s_{\phi 4} = -1.5054 + j2.8634, \\ s_{\phi 5} = -1.5054 - j2.8634, \end{cases} \quad \begin{cases} s_{\theta 1} = -4.0447, \\ s_{\theta 2} = -3.4620 + j3.1282, \\ s_{\theta 3} = -3.4620 - j3.1282, \\ s_{\theta 4} = -1.8089 + j3.6116, \\ s_{\theta 5} = -1.8089 - j3.6116. \end{cases} \tag{4.6}$$

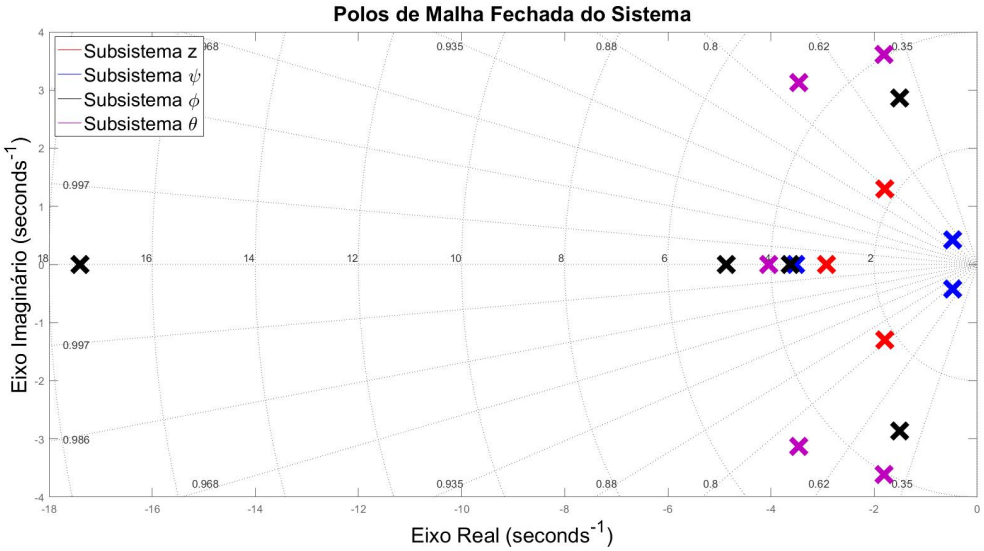


Fig. 41 – Closed-loop poles of the LQR Controller.

It is observed that the system poles are stable, have a damping coefficient $\zeta > 0.45$ and are within the region delimited by the tuning parameters (4.3) and (4.4).

4.2 Robust LQR Control with allocation

In (SANTANA, 2016) it is experimentally verified that all parameters in Table 1 can vary, including by the vehicle's autopilot, depending on the environment in which the UAV is flying. Therefore, it is ideal that any trajectory controller designed has a robustness capable of maintaining system stability in different scenarios. In this project, the designed controller takes into account the uncertainties in the time constants $\tau_{\dot{z}}$ and $\tau_{\dot{\psi}}$, and in the drag coefficients translational C_x and C_y, discussed in section 2.3. These parameters are present in the A matrix of each subsystem, thus having a greater impact on the system dynamics in case of variation. Furthermore, the increase in the coefficients C_x and C_y means an increase in the translational drag force that opposes the movement, that is, in outdoor environments it is possible for the *drone* to encounter coefficients greater than those which were modeled indoors. Initially, the following parameters are considered:

Tab. 2 – UAV AR.Drone 2.0 parameters with uncertainties.

$K_\phi = \begin{cases} 1.0 \ se \ \lvert u_\phi \rvert \geq 0.5 \\ 2.0 \ se \ \lvert u_\phi \rvert < 0.5 \end{cases}$	$K_\theta = \begin{cases} 1.0 \ se \ \lvert u_\theta \rvert \geq 0.5 \\ 2.0 \ se \ \lvert u_\theta \rvert < 0.5 \end{cases}$
$K_{\dot{z}} = 1.3$ e $K_{\dot{\psi}} = 1.0$	$m = 0.380 \ [kg]$
$\phi_{max} = 0.26 \ [rad]$	$\theta_{max} = 0.26 \ [rad]$
$\dot{z}_{max} = 1.0 \ [m/s]$	$\dot{\psi}_{max} = 1.74 \ [rad/s]$
$\omega_\phi = 4.47 \ [Hz]$	$\omega_\theta = 4.47 \ [Hz]$
$\zeta_\phi = 0.5$	$\zeta_\theta = 0.5$
$\tau_{\dot{z}} \in [0{,}4; 0{,}45] \ [s]$	$\tau_{\dot{\psi}} \in [0,3; 0{,}4] \ [s]$
$C_x \in [0.3; 0.9] \ [1/s]$	$C_y \in [0.1; 0.5] \ [1/s]$

And the same tune as LQR:

Subsystem(θ,x):

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1000 \end{bmatrix} \\ R = 3, \\ \alpha = 0.05, \ \ \beta = 22, \ \ \rho = 85°, \\ K = [-11.59 \ \ -1.37 \ \ -17.44 \ \ -5.99], \\ K_i = 22.07, \end{cases}$$

Subsystem (z):

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 120 \end{bmatrix} \\ R = 1, \\ \alpha = 0.0, \ \ \beta = 3, \ \ \rho = 43°, \\ K = [-4.23 \ \ -1.14], \\ K_i = 3.65. \end{cases} \tag{4.7}$$

Subsystem(ϕ,y):

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 240 \end{bmatrix} \\ R = 0.1, \\ \alpha = 0.05, \ \ \beta = 22, \ \ \rho = 85°, \\ K = [-30.44 \ \ -3.14 \ 36.38 \ \ -13.89], \\ K_i = -43.58, \end{cases}$$

Subsystem (ψ):

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \\ R = 24.5, \\ \alpha = 0.4 \ \ \beta = 4.0, \ \ \rho = 60°, \\ K = [-0.85 \ \ -0.31], \\ K_i = 0.31. \end{cases} \tag{4.8}$$

Therefore, the same tests and simulations presented in the 4.1 section are carried out. The results are shown below.

4.2.1 Reference Tracking

For trajectory tracking, we obtained,

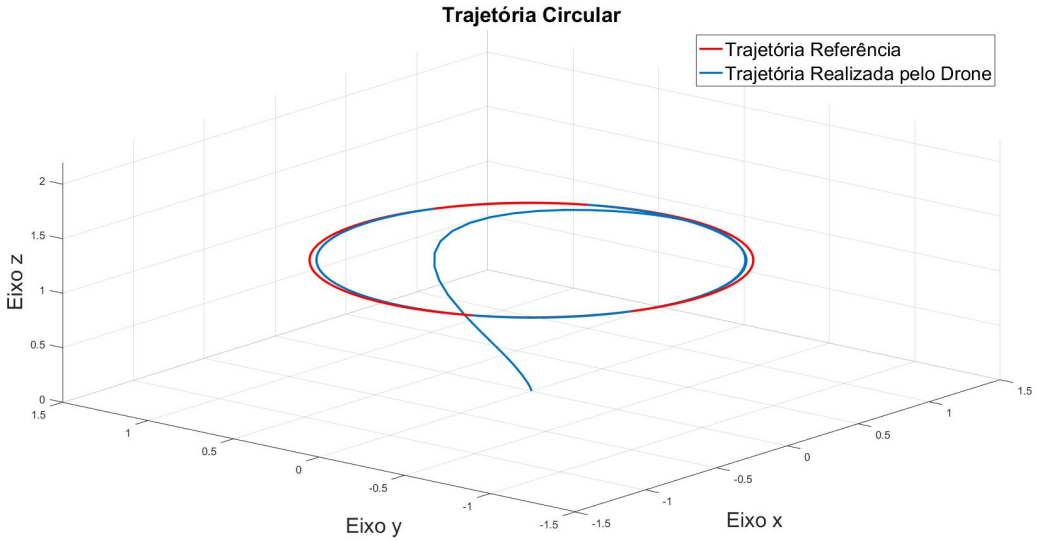


Fig. 42 – Circular Trajectory Simulation 05.

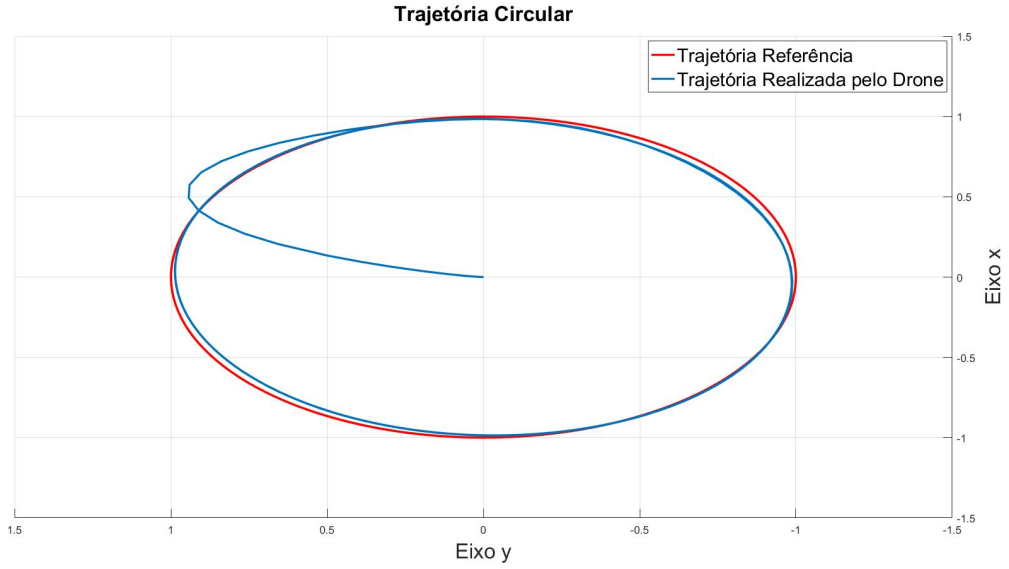


Fig. 43 – Circular Trajectory Simulation 06.

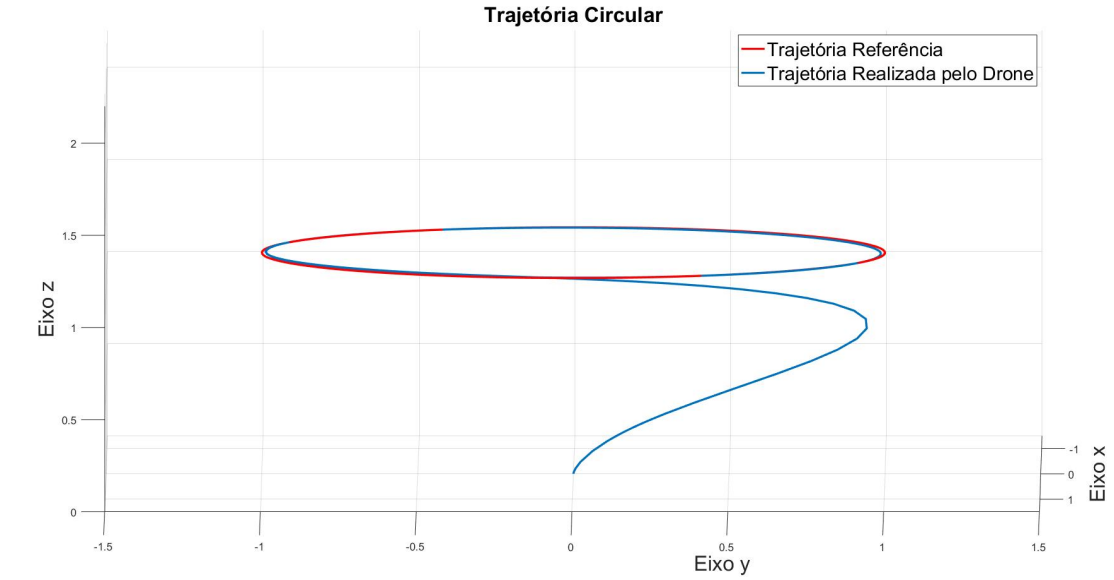


Fig. 44 – Circular Trajectory Simulation 07.

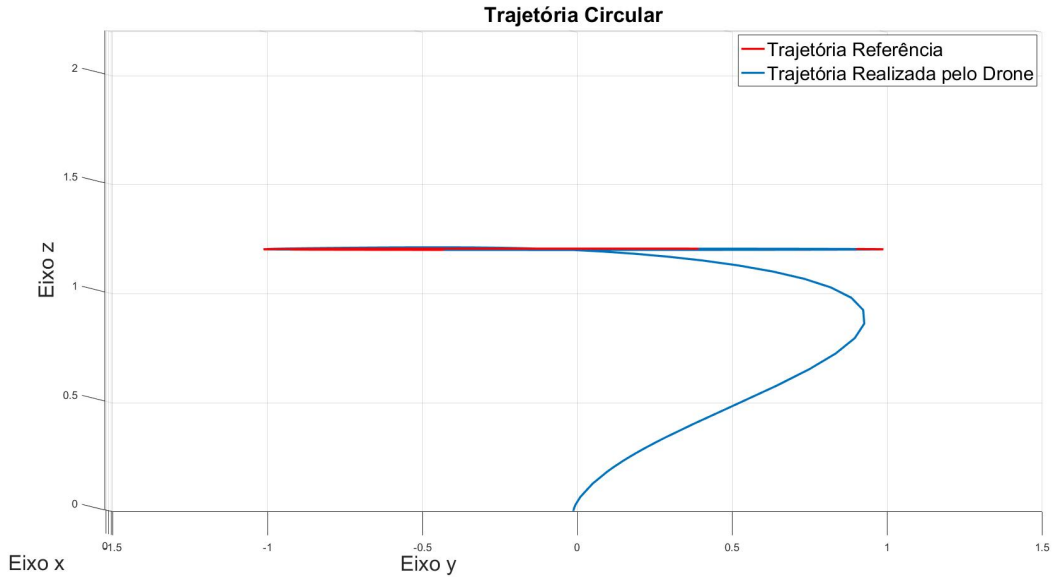


Fig. 45 – Circular Trajectory Simulation 08.

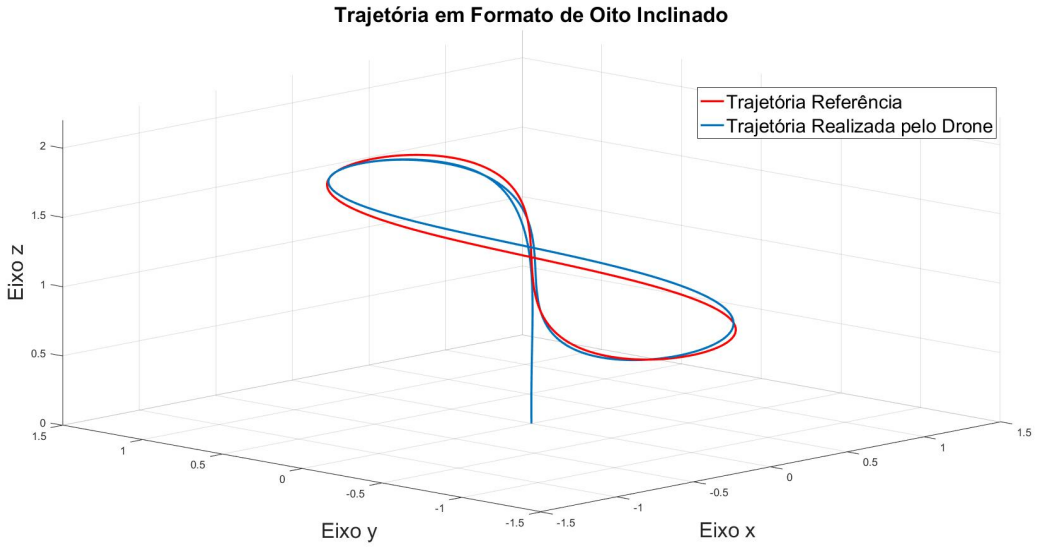


Fig. 46 – Trajectory Simulation in Eight Shape 05.

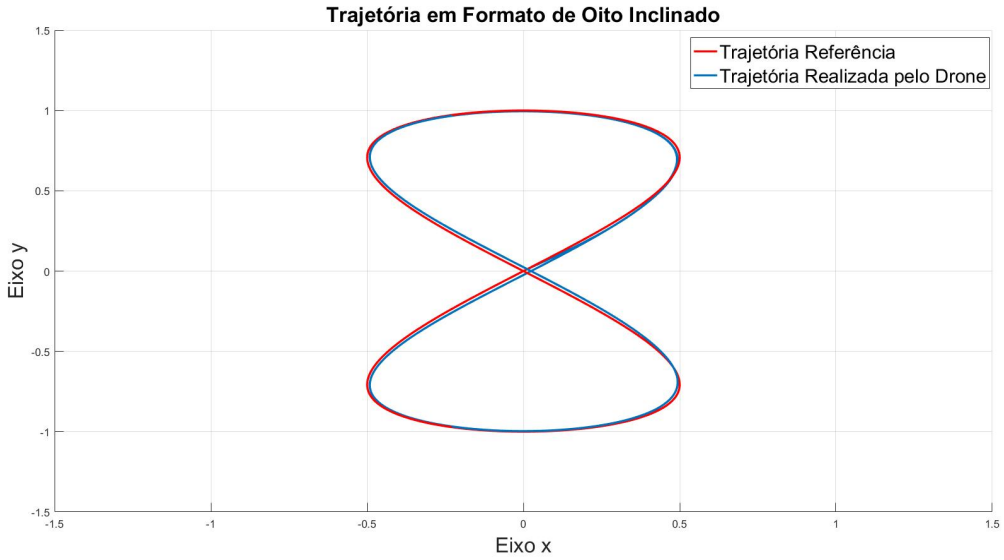


Fig. 47 – Trajectory Simulation in Eight Shape 06.

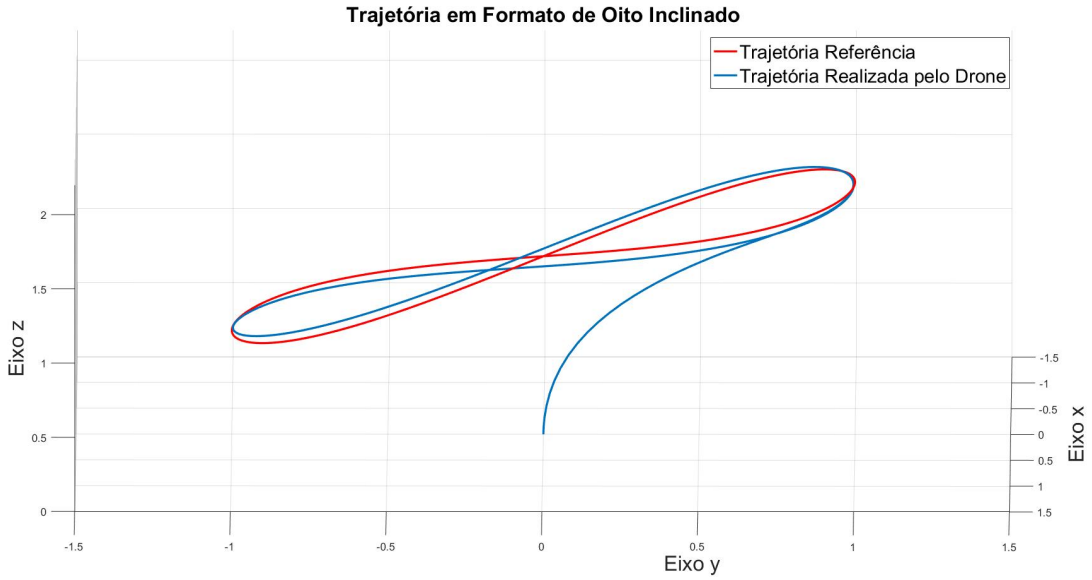


Fig. 48 – Trajectory Simulation in Eight Shape 07.

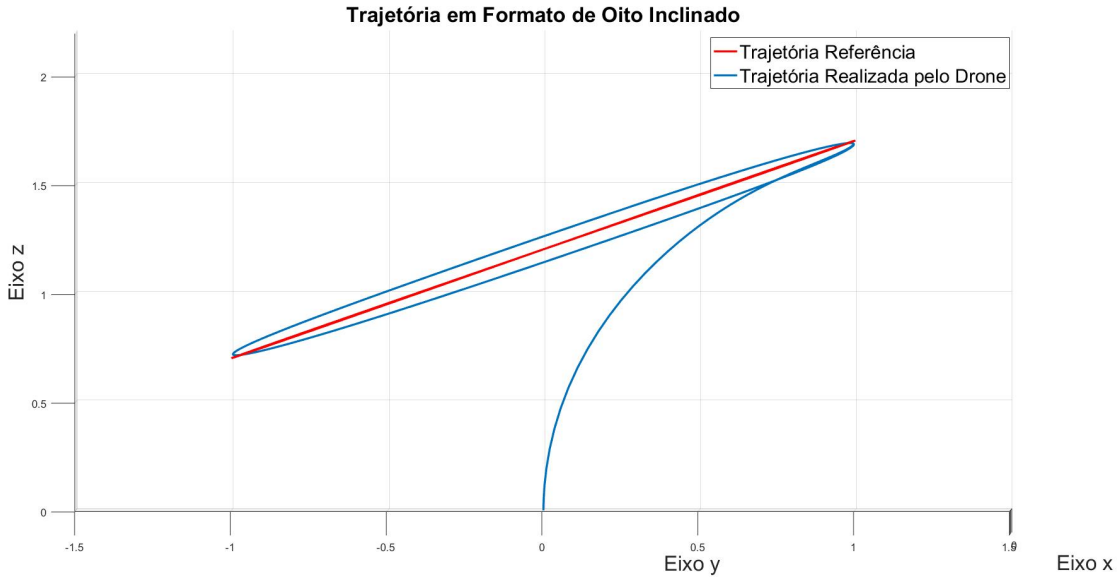


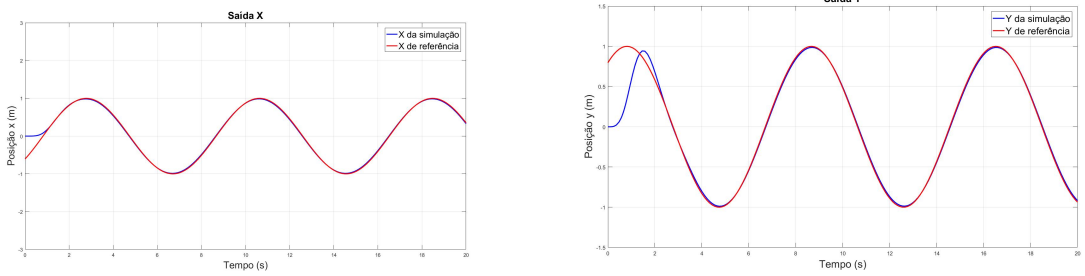
Fig. 49 – Trajectory Simulation in Eight Shape 08.

It can be observed, therefore, that the results were similar to those obtained in the first project. However, the controller is robust, that is, stability and performance are guaranteed considering all the uncertainties associated with the AR.Drone 2.0 parameters (Table 2).

4.2.2 Outputs

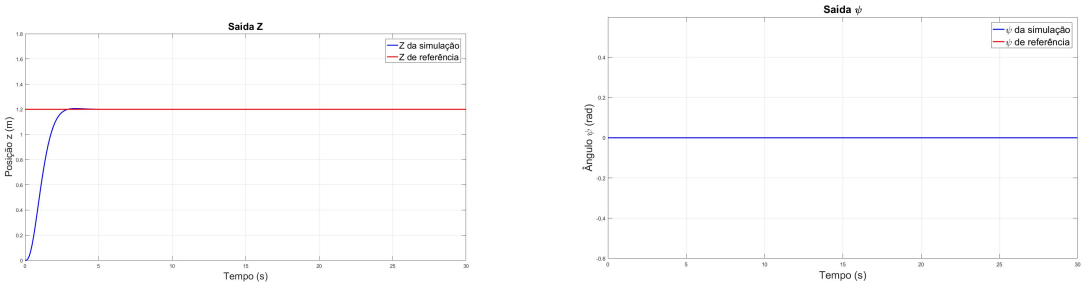
For the output of each subsystem, the results are also similar.

Circular Trajectory:



(a) Output x. (b) Output y.

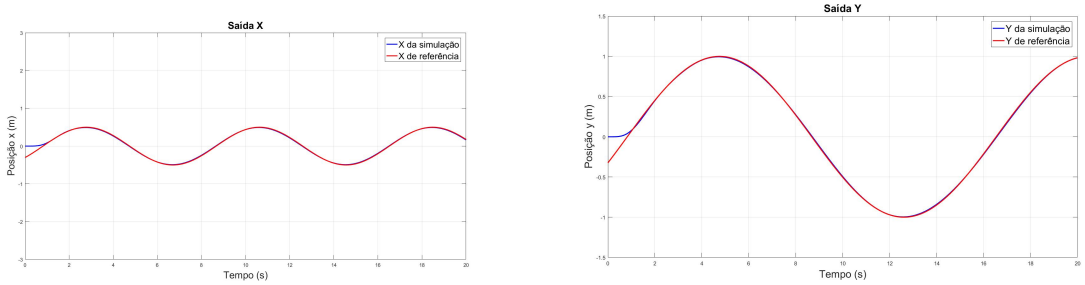
Fig. 50 – Outputs x and y from the second project.



(a) Output z. (b) Output ψ.

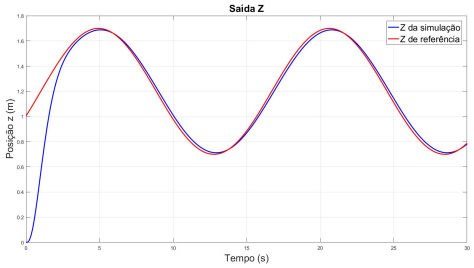
Fig. 51 – Outputs z and ψ from the second project.

Eight Shaped Trajectory:

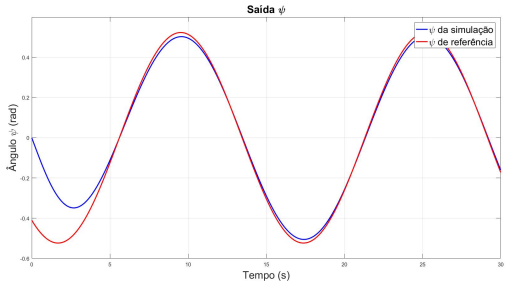


(a) Output x. (b) Output y.

Fig. 52 – Outputs x and y from the second project 02.



(a) Output z.



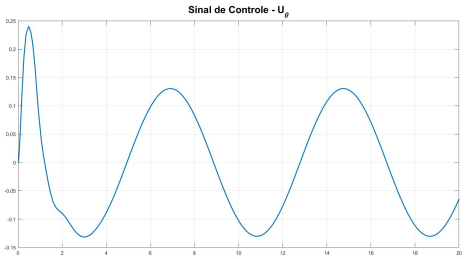
(b) Output ψ.

Fig. 53 – Outputs z and ψ from the second project 02.

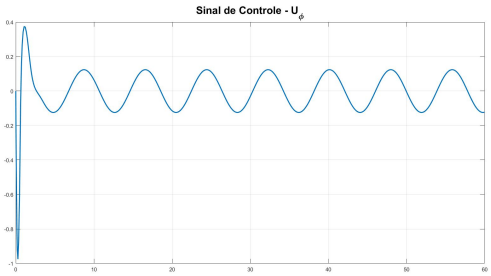
4.2.3 Control Signals

Also for control signals:

Circular Trajectory:

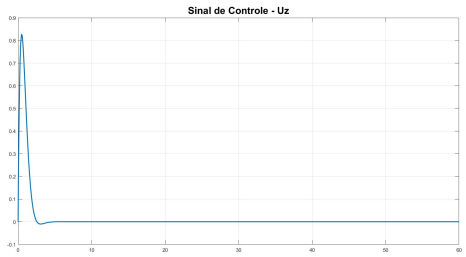


(a) Control Signal u_θ.

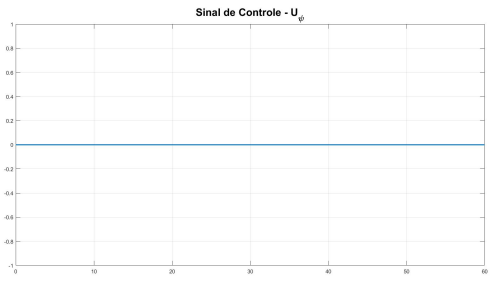


(b) Control Signal u_ϕ.

Fig. 54 – Control Signals u_θ and u_ϕ of the second project.



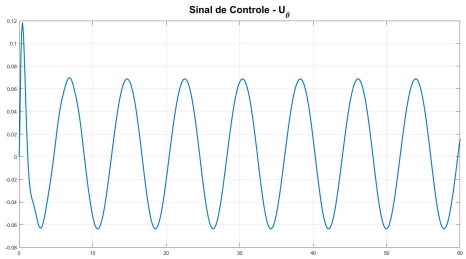
(a) Control Signal $u_{\dot{z}}$.



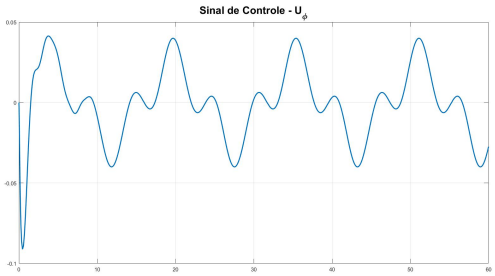
(b) Control Signal $u_{\dot{\psi}}$.

Fig. 55 – Control Signals $u_{\dot{z}}$ and $u_{\dot{\psi}}$ of the second project.

Eight Shaped Trajectory:

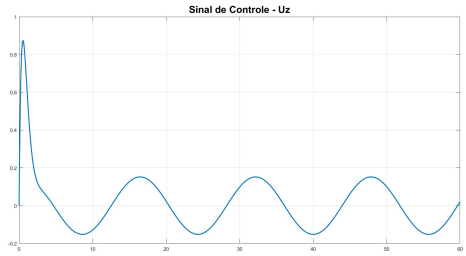


(a) Control Signal u_θ.

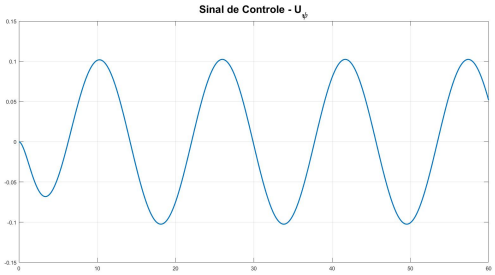


(b) Control Signal u_ϕ.

Fig. 56 – Control Signals u_θ and u_ϕ from the second project 02.



(a) Control Signal $u_{\dot{z}}$.



(b) Control Signal $u_{\dot{\psi}}$.

Fig. 57 – Control Signals $u_{\dot{z}}$ and $u_{\dot{\psi}}$ of the second project 02.

It is observed that, as well as the results of trajectory tracking, the outputs and control signals present results similar to the first project.

4.2.4 Closed-loop poles

For this project, the values of the system's closed-loop poles are (see Figure 58):

$$\begin{cases} s_{z1} = -2.7894, \\ s_{z2} = -1.7003 + j1.1663, \\ s_{z3} = -1.7003 - j1.1663, \end{cases} \quad \begin{cases} s_{\psi 1} = -3.9993, \\ s_{\psi 2} = -0.5562 + j0.3775, \\ s_{\psi 3} = -0.5562 - j0.3775, \end{cases} \tag{4.9}$$

$$\begin{cases} s_{\phi 1} = -16.9996, \\ s_{\phi 2} = -6.4726, \\ s_{\phi 3} = -2.7538, \\ s_{\phi 4} = -1.4184 + j2.9953, \\ s_{\phi 5} = -1.4184 - j2.9953, \end{cases} \quad \begin{cases} s_{\theta 1} = -3.0677, \\ s_{\theta 2} = -4.7094 + j3.1759, \\ s_{\theta 3} = -4.7094 - j3.1759, \\ s_{\theta 4} = -1.4681 + j3.8560, \\ s_{\theta 5} = -1.4681 - j3.8560. \end{cases} \tag{4.10}$$

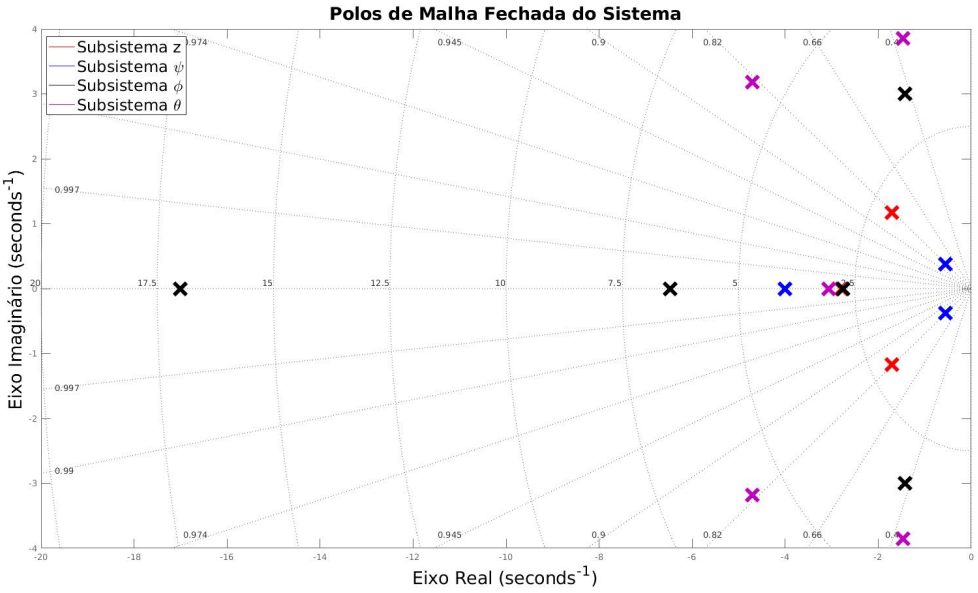


Fig. 58 – Closed-loop poles of the LQR Robust Controller

It is observed that the poles in this second project remain asymptotically stable and are also located close to the poles of the first project, with a damping coefficient $\zeta > 0.35$ and are within the region delimited by the tuning parameters (4.3) and (4.4).

4.3 Controller Robustness Analysis

The Robust LQR controller design presents stability and good performance both for simulations with the nominal model (using the parameters in Table 1), and for a combination of values within the ranges determined in Table 2. Furthermore, when simulating the system with the non-robust LQR controller, for some combinations of the parameters presented in Table 2, the system also showed stability and good trajectory tracking results. However, as it is not possible to test the infinite possible combinations of uncertain parameters, and these were not considered in the controller design, it cannot be said that it is robust. Therefore, it is not possible to guarantee the stability of the system with these associated polytopic uncertainties. to the time constants $\tau_{\dot{z}}$ and $\tau_{\dot{\psi}}$, and to the translational drag coefficients C_x and C_y.

Furthermore, a new project was carried out for the robust LQR controller with the same tuning parameters as the previous projects and with the parameter ranges $\tau_{\dot{z}}$, C_x and C_y increased, as shown in the following table:

Tab. 3 – UAV AR.Drone 2.0 parameters with increased range.

$K_\phi = \begin{cases} 1.0 \ se \ \lvert u_\phi \rvert \geq 0.5 \\ 2.0 \ se \ \lvert u_\phi \rvert < 0.5 \end{cases}$	$K_\theta = \begin{cases} 1.0 \ se \ \lvert u_\theta \rvert \geq 0.5 \\ 2.0 \ se \ \lvert u_\theta \rvert < 0.5 \end{cases}$
$K_{\dot{z}} = 1.3$ e $K_{\dot{\psi}} = 1.0$	$m = 0.380 \ [kg]$
$\phi_{max} = 0.26 \ [rad]$	$\theta_{max} = 0.26 \ [rad]$
$\dot{z}_{max} = 1.0 \ [m/s]$	$\dot{\psi}_{max} = 1.74 \ [rad/s]$
$\omega_\phi = 4.47 \ [Hz]$	$\omega_\theta = 4.47 \ [Hz]$
$\zeta_\phi = 0.5$	$\zeta_\theta = 0.5$
$\tau_{\dot{z}} \in [0{,}4; 0{,}8] \ [s]$	$\tau_{\dot{\psi}} \in [0,3; 0{,}4] \ [s]$
$C_x \in [0{,}3; 1{,}0] \ [1/s]$	$C_y \in [0{,}1; 3{,}0] \ [1/s]$

Obtaining the following state feedback gains:

Subsystem(θ,x):

$$\begin{cases} K = [-11.81 \ \ -1.39 \ \ -17.94 \ \ -6.10], \\ K_i = 22.71, \end{cases}$$

Subsystem (z):

$$\begin{cases} K = [-1.15 \ \ -0.50], \\ K_i = 0.24. \end{cases}$$

Subsystem(ϕ,y): Subsystem (ψ):

$$\begin{cases} K = [-34.18 \ \ -3.25 \ 48.14 \ 14.97], \\ K_i = -54.80, \end{cases} \qquad \begin{cases} K = [-0.85 \ \ -0.31], \\ K_i = 0.31. \end{cases}$$

Then, considering the values of the upper limit of each range of uncertain parameters, that is,

$$\begin{cases} \tau_{\dot{z}} = 0.8, \\ \tau_{\dot{\psi}} = 0.4, \end{cases} \qquad \begin{cases} C_x = 1.0, \\ C_y = 3.0, \end{cases} \tag{4.11}$$

The system was simulated with this new robust controller, obtaining the results in Figures (59), (60) and (63). Then, still considering the model with increased intervals (4.11), it was simulated using the non-robust controller from the first project, obtaining the results in Figures (61), (62) and (64).

When analyzing these results, it is clearly observed that the LQR controller cannot guarantee the following of the specified trajectory, and that the LQR Robust controller has degraded its performance, however it maintains a certain consistency in the movement and has robust stability guaranteed. Furthermore, it can be seen that both controllers do not allow the system to lose stability, since the poles remain in the left half-plane (Figures 63 and 64). However, in addition to the lack of robustness guarantee of the LQR controller, it can be seen in Figure 64 that most of the poles are concentrated close to the imaginary axis, at the threshold of stability.

The complete, partitioned code for this project is in the Appendix B and C, and the entire controller project (.me .slx files) can be accessed through GitHub: Project.

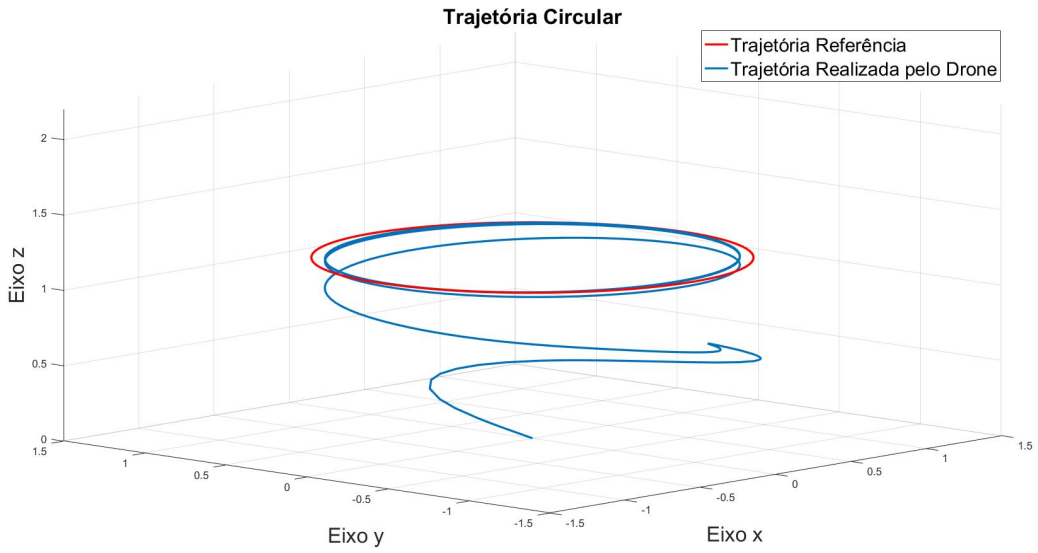


Fig. 59 – Robust LRQ Controller Simulation I.

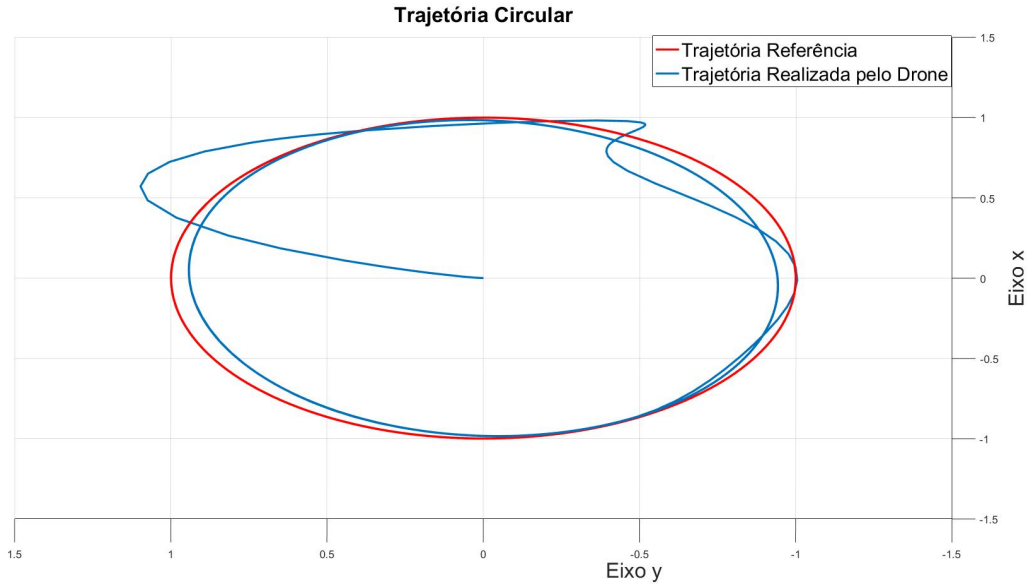


Fig. 60 – Robust II LRQ Controller Simulation.

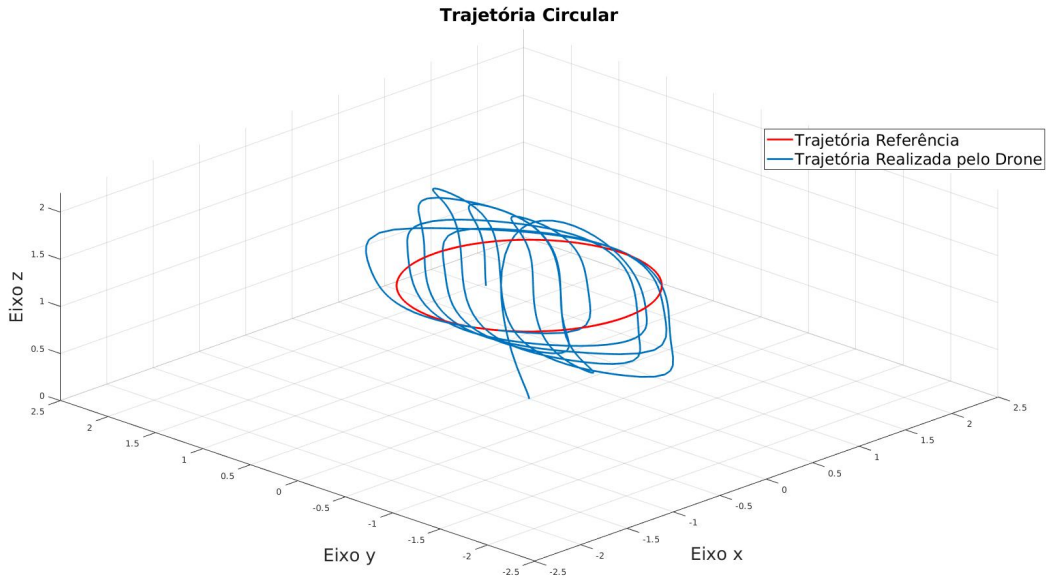


Fig. 61 – LRQ I Controller Simulation.

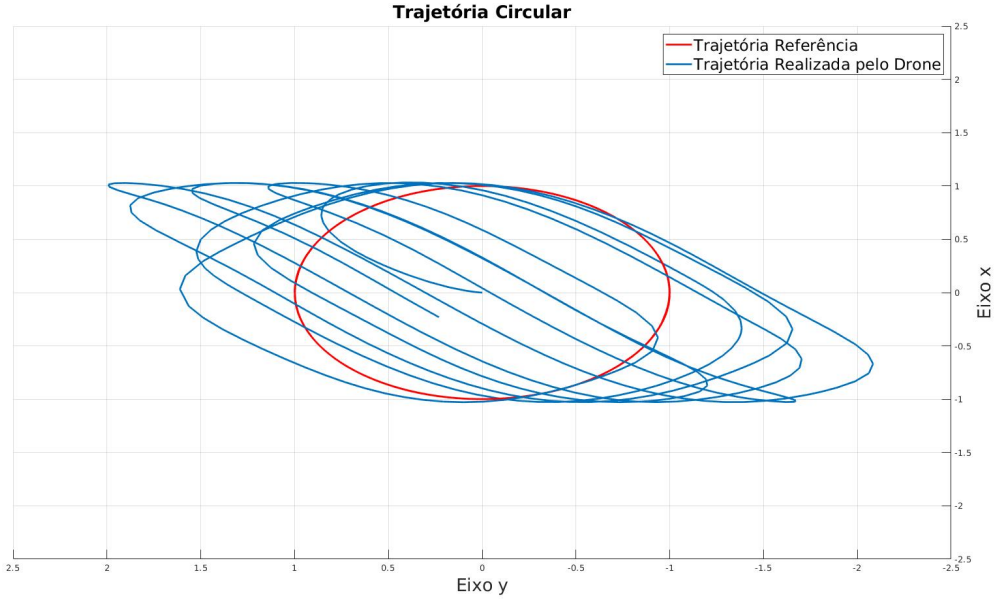


Fig. 62 – LRQ II Controller Simulation.

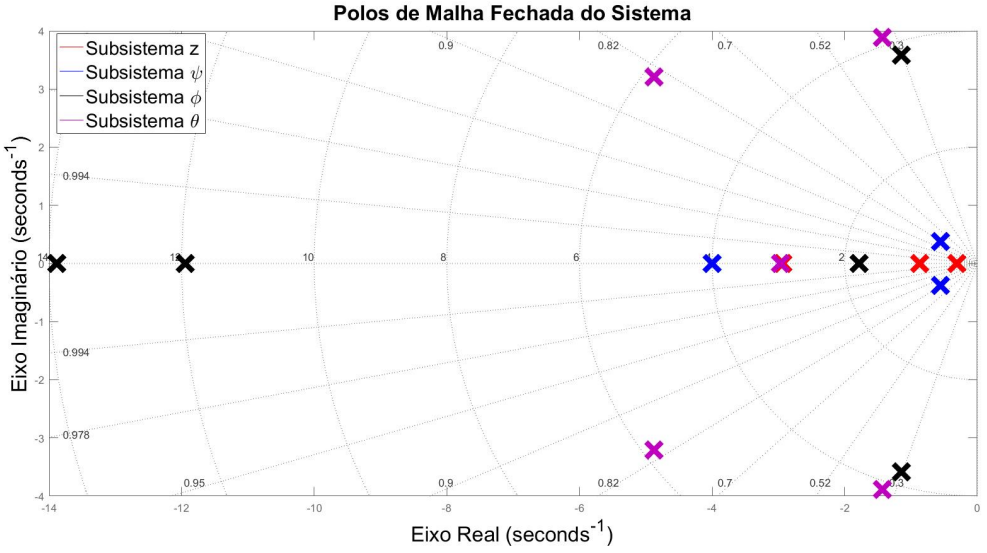


Fig. 63 – Poles with the Robust LRQ Controller.

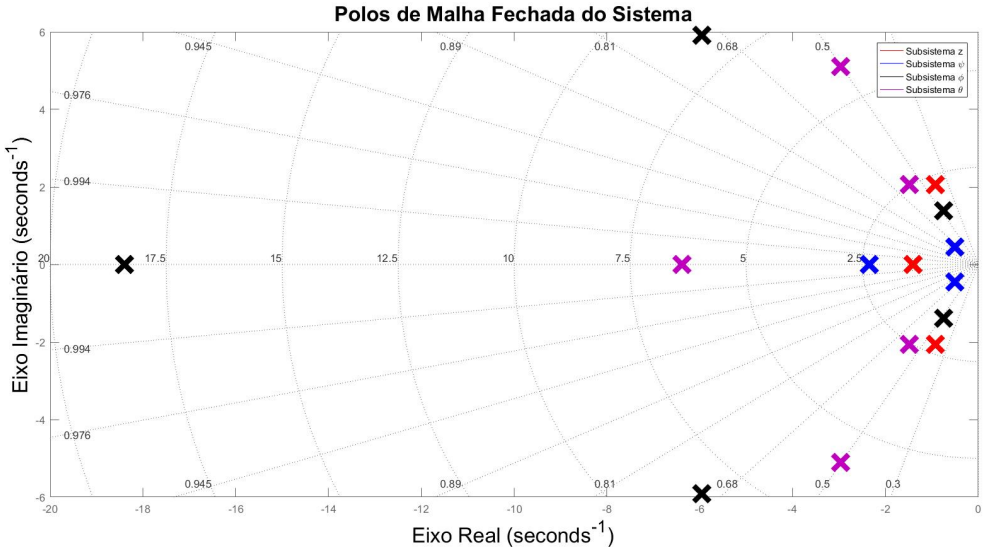


Fig. 64 – Poles with the LRQ Controller.

5 CONCLUSIONS

This work carried out a study of a quad-rotor Unmanned Aerial Vehicle , in particular the AR.Drone 2.0 model. Presenting a brief chronology of the development of *drones* in the world, basic concepts of the operation of a quadrirotor , the modeling of the UAV AR.Drone 2.0 and the design of the trajectory controller.

The study of the development history of UAVs shows that, for years, the first investments and applications were for military use. Over the past three decades, civil applications, model diversity and innovations in aircraft size have grown exponentially. This growing popularity has demanded investment in research into the type of flight technology, trajectory and stabilization controls, battery autonomy, among others.

It is also concluded that the literature on the mathematical modeling of UAVs quad rotors is complex, the vast majority of which present elaborate and challenging methods for developing a control system. However, in (ENGEL; STURM; CREMERS, 2014; NONAMI *et al.*, 2010; RAFFO; ORTEGA; RUBIO, 2010; SANTANA, 2016), for example, there are developments and validation of simplified and linearized models that present good practical results. This simplification, combined with the technological and financial accessibility of AR.Drone , allows for the academic democratization of the study of control systems applied to UAVs quadrirotors . In this work, this accessibility is demonstrated in the simulation results of the LQR and robust LQR controllers, in which designs based on the simplified model guaranteed stability and performance.

The innovation of embedded systems with more precise sensors and the development of computer vision have contributed to the application of control techniques for UAVs . One of these contributions is the possibility of measuring and estimating all aircraft state variables, which, combined with the system having a completely controllable state, allows the development of state feedback controllers. In this work, the designed controller, in addition to using the state feedback strategy, minimizes the index J_∞, which promotes the balance between system performance and control effort, determining a feedback gain K that allocates the closed-loop poles in a specific region of the s plane, related to good specifications of the transient response, such as maximum overshoot and settling time, and guarantees the stability and performance of the system considering parametric uncertainties in the time constants $\tau_{\dot{z}}$ and $\tau_{\dot{\psi}}$ and in the translational drag coefficients C_x and C_y.

The results of the simulations carried out demonstrate that the simplified model has a good approximation to the non-linear model, that the LQR controller presented good results in trajectory tracking, that the allocation of closed-loop poles in a specific region of the plane s decreased the maximum overshoot at the altitude of the *drone* and the settling time following the reference, and that the robust LQR design guarantees stability and performance even with relatively large variations in time constants and translational drag coefficients.

5.1 Future Work

One possibility for future work is to compare the LQR technique studied with other control techniques, such as PID, optimal control with minimization of the H_∞ norm or the H_2 norm, predictive control and even non-linear control such as sliding mode control (*Sliding Mode*). Furthermore, understanding the dynamics of this *drone* model facilitates future studies of other models. Furthermore, it is proposed as a future work the learning of robotic simulators, such as *ROS* and *CoppeliaSim*, to carry out tests of the controllers designed in this work, and subsequently, to carry out practical experiments in a laboratory.

Referências

ARAÚJO, H. X. de. *FUNDAMENTOS DE CONTROLE ÓTIMO, ROBUSTO E PREDITIVO*. [S.l.]: NOTAS DE AULA, 2020. 52, 53

ÅSTRÖM, K. J.; MURRAY, R. M. *Feedback systems*. [S.l.]: Princeton university press, 2010. 50, 51

BOUABDALLAH, S. *Design and control of quadrotors with application to autonomous flying*. [S.l.], 2007. 27, 31

BOUABDALLAH, S.; NOTH, A.; SIEGWART, R. Pid vs lq control techniques applied to an indoor micro quadrotor. In: IEEE. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. [S.l.], 2004. v. 3, p. 2451–2456. 27

BRANDÃO, A. S. Projeto de controladores não lineares para voo autônomo de veiculos aereos de pas rotativas. Universidade Federal do Espírito Santo, 2013. 36

BRESCIANI, T. Modelling, identification and control of a quadrotor helicopter. MSc theses, 2008. 27

CASTILLO, P.; DZUL, A.; LOZANO, R. Real-time stabilization and tracking of a four-rotor mini rotorcraft. IEEE Transactions on control systems technology, v. 12, n. 4, 2004. 40

CHAO, H.; CAO, Y.; CHEN, Y. Autopilots for small unmanned aerial vehicles: a survey. International publisher of Control, Automation and Systems, v. 8, n. 1, p. 36–44, 2010. 27

CHILALI, M.; GAHINET, P.; APKARIAN, P. Robust pole placement in lmi regions. IEEE transactions on Automatic Control, v. 44, n. 12, p. 2257–2270, 1999. 53

CHINA, M. in. *Agricultural Power Sprayer Drone*. 2021. Disponível em: <https://agriculture-drone.en.made-in-china.com/product/tKjQUlMEbwrv/China-Agricultural-Power-Sprayer-Drone-with-High-Pressure-Italy-Famous-Brand-Nozzles.html>. Acesso em: 25 de setembro de 2021. 25

COSTA, D. A. Modelagem e controle de um veículo aéreo não tripulado. Universidade Federal da Bahia, p. 39–53, 2018. 27, 29, 30, 31, 32

DA COSTA, S. E. A. P. Controlo e simulação de um quadrirotor convencional. Lisboa, Portugal: Instituto Superior Técnico, 2008. 28, 29

DRONES, E. de. *VENTOS E DRONES, COMBINAÇÃO PERIGOSA*. 2021. Disponível em: <https://www.escoladedrones.com.br/ventos-e-drones-combinacao-perigosa/>. Acesso em: 25 de setembro de 2021. 25

ENGEL, J.; STURM, J.; CREMERS, D. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. Robotics and Autonomous Systems, v. 62, n. 11, p. 1646–1656, 2014. 40, 83

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. *Sistemas de controle para engenharia*. [S.l.]: Bookman Editora, 2013. 48, 49

GARCIA P.C.; LOZANO, R. D. A. *Modelling and Control of Mini-Flying Machines*. [S.l.]: Springer Science & Business Media, 2006. 2–3 p. 23, 24, 25, 28

KRAJNÍK, T. *et al*. Ar-drone as a platform for robotic research and education. In: SPRINGER. *International conference on research and education in robotics*. [S.l.], 2011. p. 172–186. 27, 41

LIMA, G. V. Modelagem dinâmica e controle para navegação de um veículo aéreo não tripulado do tipo quadricóptero. Universidade Federal de Uberlândia, p. 53–62, 2015. 27

NONAMI, K. *et al*. *Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles*. [S.l.]: Springer Science & Business Media, 2010. 7–12 p. 23, 24, 25, 26, 40, 83

OGATA, K. *Engenharia de Controle Moderno*. [S.l.]: Pearson, 2014. 47, 48, 51

PISKORSKI NICOLAS BRULEZ, P. E. F. D. S. Ar.drone developer guide. SDK Version 2.0, p. 6–7, 2012. 25, 35, 45

RAFFO, G. V.; ORTEGA, M. G.; RUBIO, F. R. An integral predictive/nonlinear h control structure for a quadrotor helicopter. Automatica, v. 46, n. 1, 2010. 40, 83

RAGHURAMAN, V. Modeling and h-infinity loop shaping control of a vertical takeoff and landing drone. ARIZONA STATE UNIVERSITY, p. 31–42, 2018. 27

SANTANA, L. V. Sistemas de navegação e controle para veículos aéreos não tripulados e suas aplicações. Universidade Federal do Espírito Santo, p. 18–56, 2016. 27, 31, 36, 37, 38, 39, 40, 41, 45, 68, 83

VALAVANIS, K. P.; VACHTSEVANOS, G. J. *Handbook of unmanned aerial vehicles*. [S.l.]: Springer, 2015. v. 2077. 83–88 p. 24, 27, 38

VUELAS SEGURO. *Control de vuelo*. 2021. Disponível em: <https://www.vuelaseguro.cl/conceptos-para-entender/>. Acesso em: 17 de outubro de 2021. 30

WIKILIVROS. *Coordenadas 3D*. 2021. Disponível em: <https://pt.wikibooks.org/wiki/Tornando-se_profissional_em_Blender_3D/Pensando_em_3D>. Acesso em: 17 de outubro de 2021. 30

APÊNDICE A – Modelos Programados no SIMULINK

A.1 Modelo não Linear

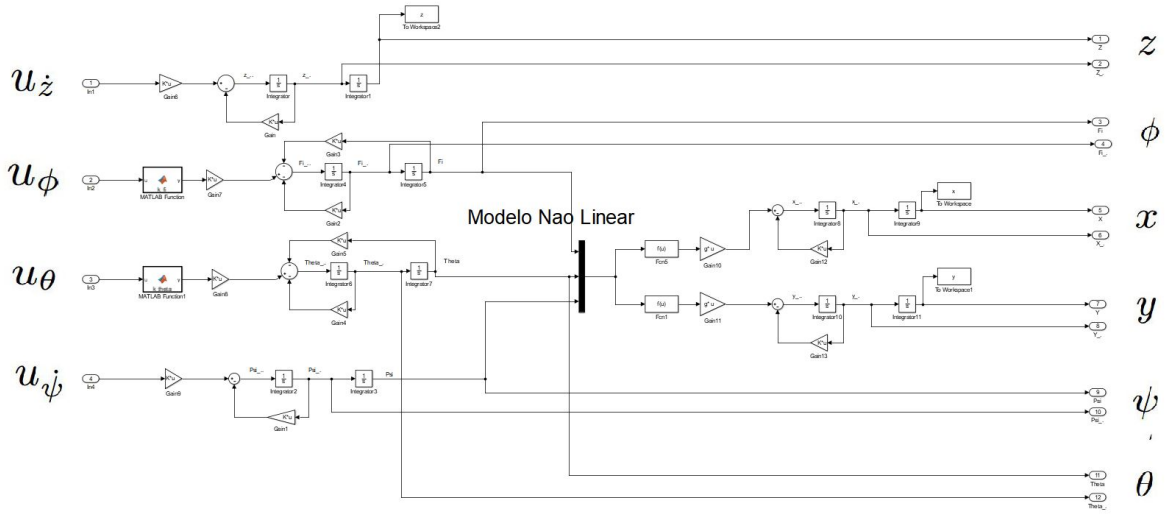


Fig. 65 – Modelo Não Linear programado no Simulink.

A.2 Sistema de Controle do AR.Drone 2.0

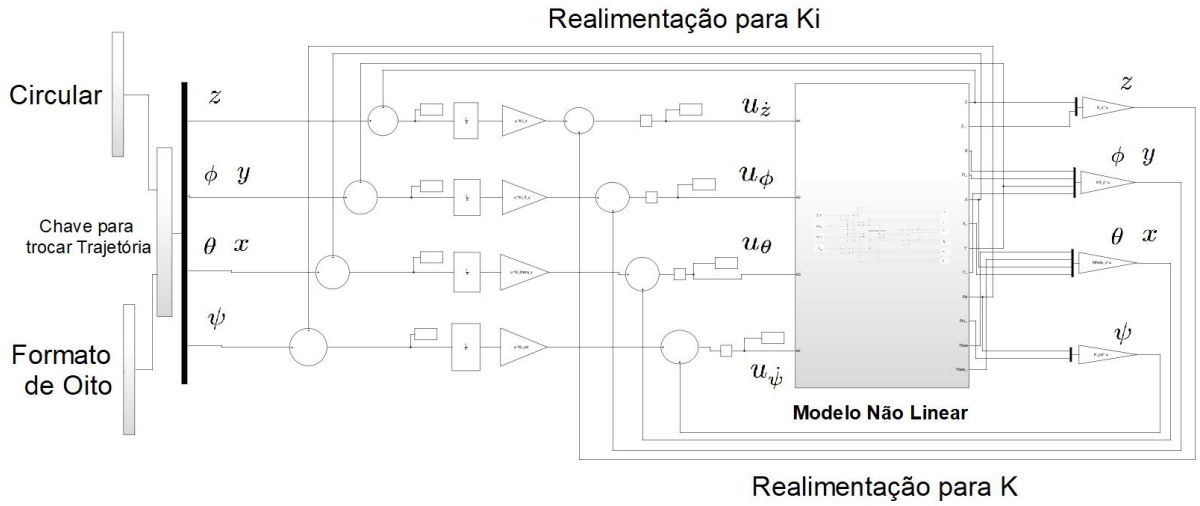


Fig. 66 – Sistema de Controle do AR.Drone 2.0 por realimentação de estado com integrador.

A.3 Trajetórias

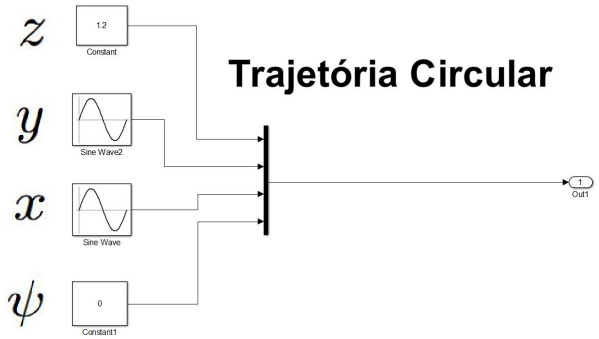


Fig. 67 – Programação da trajetória circular no SIMULINK.

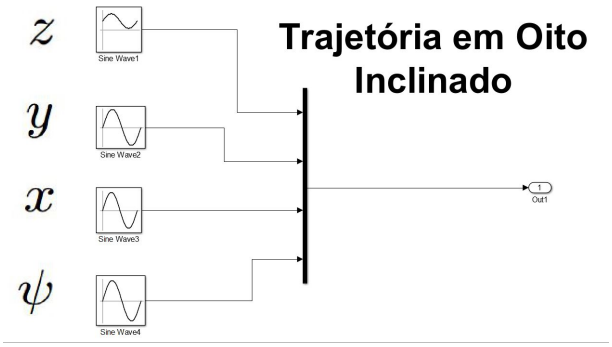


Fig. 68 – Programação da trajetória em formato de oito inclinado no SIMULINK.

APÊNDICE B – Códigos Particionados do Projeto do Controlador

B.1 Parâmetros do AR.Drone

```
%% ==========> Parametros do AR.Drone <============
g = 9.8;
Cx = 0.3; Cy = 0.1;
Kz = 1.3; Kpsi = 1; Kfi = 1.5; Ktheta = 1.5;
tz = 0.4; tpsi = 0.3;
zmax = 1.0; fimax = 0.26; thetamax = 0.26; psimax = 1.74;
wfi = 4.47; wtheta = 4.47;
zetafi = 0.5; zetatheta = 0.5;
```

Listing B.1 – Parâmetros do AR.Drone

B.2 Matrizes do Espaço de Estado

```
%% VARIAVEL FI-Y
Afi_y = [0 1 0 0;
        -wfi^2 -2*zetafi*wfi 0 0;
          0 0 0 1;
          -g 0 0 -Cy];
Bfi_y = [0;
         Kfi*fimax*wfi^2;
         0;
         0];
Cfi_y = [0 0 1 0];
Dfi_y = [0];

%% VARIAVEL THETA-X
Atheta_x = [0 1 0 0;
        -wtheta^2 -2*zetatheta*wtheta 0 0;
           0 0 0 1;
           g 0 0 -Cx];
Btheta_x = [0;
         Ktheta*thetamax*wtheta^2;
           0;
           0];
Ctheta_x = [0 0 1 0];
Dtheta_x = [0];

```

```
%% VARIAVEL Z
Az = [0 1;
      0 -1/tz];
Bz = [0;
      (Kz*zmax)/(tz)];
Cz = [1 0];
Dz = 0;

%% VARIAVEL PSI
Apsi = [0 1;
        0 -1/tpsi];
Bpsi = [0;
       (Kpsi*psimax)/(tpsi)];
Cpsi = [1 0];
Dpsi = 0;
```

Listing B.2 – Matrizes do Espaço de Estado

B.3 Matrizes Aumentadas

```

%% VARIAVEL FI-Y
Afi_y_a = [Afi_y zeros([size(Afi_y,1),size(Bfi_y,2)]);-Cfi_y zeros(size(
    Cfi_y,1))];
Bfi_y_a = [Bfi_y;-Dfi_y];
Cfi_y_a = [Cfi_y zeros(size(Cfi_y,1))];
Dfi_y_a = [0];
Gfi_y = ss(Afi_y_a,Bfi_y_a,Cfi_y_a,Dfi_y_a);

%% VARIAVEL THETA-X
Atheta_x_a = [Atheta_x zeros([size(Atheta_x,1),size(Btheta_x,2)]);
              -Ctheta_x zeros(size(Ctheta_x,1))];
Btheta_x_a = [Btheta_x;-Dtheta_x];
Ctheta_x_a = [Ctheta_x zeros(size(Ctheta_x,1))];
Dtheta_x_a = [0];
Gtheta_x = ss(Atheta_x_a,Btheta_x_a,Ctheta_x_a,Dtheta_x_a);

%% VARIAVEL Z
Az_a = [Az zeros([size(Az,1),size(Bz,2)]);
        -Cz zeros(size(Cz,1))];
Bz_a = [Bz;-Dz];
Cz_a = [Cz zeros(size(Cz,1))];
Dz_a - 0;
Gz = ss(Az_a,Bz_a,Cz_a,Dz_a);

%% VARIAVEL PSI
Apsi_a = [Apsi zeros([size(Apsi,1),size(Bpsi,2)]);
```

```
          -Cpsi zeros(size(Cpsi,1))];
Bpsi_a = [Bpsi;-Dpsi];
Cpsi_a = [Cpsi zeros(size(Cpsi,1))];
Dpsi_a = 0;
Gpsi = ss(Apsi_a,Bpsi_a,Cpsi_a,Dpsi_a);
```

Listing B.3 – Matrizes Aumentadas

B.4 Primeira versão da Função lqrvialmi()

```
%=================================================================
%======================== LQR VIA LMI ============================
%=================================================================

function K = lqrvialmi(SYS,Q,R,x0)
%% Obtendo matrizes e as dimensoes
A = SYS.A;
B = SYS.B;
C = SYS.C;
D = SYS.D;
n = size(A,1);
m = size(B,2);

%% LMI
setlmis([])

%% Declaracao de Variaveis
gama = lmivar(1,[1 1]);
[W1,n_W1] = lmivar(1,[n 1]);
[W2,n_W2] = lmivar(2,[m n]);

%% ===========> Declaracao de LMIs
%% 1 LMI
lmiterm([-1 1 1 gama],1,1);
lmiterm([-1 2 1 0],x0);
lmiterm([-1 2 2 W1],1,1);
%% 2 LMI
lmiterm([2 1 1 W1],A,1,'s');
lmiterm([2 1 1 W2],B,1,'s');
lmiterm([2 2 1 W1],Q^0.5,1);
lmiterm([2 3 1 W2],R^0.5,1);
lmiterm([2 2 2 0],-1);
lmiterm([2 3 2 0],0);
lmiterm([2 3 3 0],-1);
%% 3 LMI
lmiterm([-3 1 1 gama],1,1);
%% 4 LMI
```

```
lmiterm([-4 1 1 W1],1,1);

%% Solver
LQR_lmi = getlmis;
c = [1 zeros(1,n_W2-1)]';
options = [1e-8,300,1e9,20,1];
[c_opt,x_opt] = mincx(LQR_lmi,c,options);

%% Obtencao dos Resultados
gama_opt = dec2mat(LQR_lmi,x_opt,gama);
W1_opt = dec2mat(LQR_lmi,x_opt,W1);
W2_opt = dec2mat(LQR_lmi,x_opt,W2);

K = W2_opt*inv(W1_opt);
```

Listing B.4 – Função *lqrvialmi()*

B.5 Cálculo do Ganho K aumentado

```
%% CALCULO DOS GANHOS Ks AUMENTADOS - VIA LMI
%% Z
K_z2 = lqrvialmi(Gz,Qz,Rz,x0)
%% PSI
K_psi2 = lqrvialmi(Gpsi,Qpsi,Rpsi,x0)
%% FI
Kfi_y2 = lqrvialmi(Gfi_y,Qfi_y,Rfi_y,x0)
%% THETA
Ktheta_x2 = lqrvialmi(Gtheta_x,Qtheta_x,Rtheta_x,x0)
```

Listing B.5 – Cálculo do Ganho K aumentado

B.6 Separação dos ganhos K e K_i

```
%% ============================================================
%%  ===== GANHO DE REALIMENTACAO DE ESTADO E INTEGRAL ==========
%%  ============================================================
%% Z
Ki_z = K_z2(:,3);
K_z = K_z2(:,1:2);
%% PSI
Ki_psi = K_psi2(:,3);
K_psi = K_psi2(:,1:2);
%% FI-Y
Ki_fi_y = Kfi_y2(:,5);
Kfi_y = Kfi_y2(:,1:4);
%% THETA-X
```

```
Ki_theta_x = Ktheta_x2(:,5);
Ktheta_x = Ktheta_x2(:,1:4);
```

Listing B.6 – Separação dos ganhos K e K_i

B.7 Polos do Sistema

```
%% ===================== POLOS DO SISTEMA ======================
%% Z
Af_z = [Az+Bz*K_z Bz*Ki_z;
        -(Cz+Dz*K_z) -Dz*Ki_z];
polosZ = eig(Af_z)
%% PSI
Af_psi = [Apsi+Bpsi*K_psi Bpsi*Ki_psi;
        -(Cpsi+Dpsi*K_psi) -Dpsi*Ki_psi];
polosPSI = eig(Af_psi)
%% FI
Af_fi = [Afi_y+Bfi_y*Kfi_y Bfi_y*Ki_fi_y;
        -(Cfi_y+Dfi_y*Kfi_y) -Dfi_y*Ki_fi_y];
polosFI = eig(Af_fi)
%% THETA
Af_theta = [Atheta_x+Btheta_x*Ktheta_x Btheta_x*Ki_theta_x;
        -(Ctheta_x+Dtheta_x*Ktheta_x) -Dtheta_x*Ki_theta_x];
polosTHETA = eig(Af_theta)
```

Listing B.7 – Polos do Sistema

B.8 Simulação Gráfica da Trajetória do VANT

```
%% Tipo de Trajetoria
% circulo = 1 /// oito inclinado = 0
tipo = 1;
%% ===============> Plot da Trajetoria em 3D <==================
%% ===============> Plot da Trajet ria em 3D <==================
if tipo == 1
    %REFER NCIA CIRCULAR
    figure
    t = tout;
    xref = sin(0.8*t);
    yref = cos(0.8*t);
    zref = 1.2*ones([1 length(t)]);
    psiref = 0;
    plot3(xref,yref,zref,'r','LineWidth',2);
    title('Trajet ria Circular','FontSize',fontsize)

    xref = sin(0.8*t-0.65);
```

```
    yref = cos(0.8*t-0.65);
else
    %REFER NCIA OITO INCLINADO
    figure
    t = tout;
    xref = 0.5*sin(0.8*t);
    yref = sin(0.4*t);
    zref = 1.2 + 0.5*sin(0.4*t);
    psiref = -((pi/6)*sin(0.4*t));
    plot3(xref,yref,zref,'r','LineWidth',2);
    title('Trajet ria em Formato de Oito Inclinado','FontSize',fontsize
    )

    xref = 0.5*sin(0.8*t-0.65);
    yref = sin(0.4*t-0.33);
    zref = 1.2 + 0.5*sin(0.4*t-0.4);
    psiref = -((pi/6)*sin(0.4*t+0.9));
end

%Simula  o
hold on
plot3(x,y,z,'LineWidth',2);
xlabel('Eixo x','FontSize',fontsize)
ylabel('Eixo y','FontSize',fontsize)
zlabel('Eixo z','FontSize',fontsize)
lgd = legend('Trajet ria Refer ncia','Trajet ria Realizada pelo Drone
    ')
lgd.FontSize = fontsize2;
grid on
axis([-1.5 1.5 -1.5 1.5 0 2.2]) %Ajuste do Gr fico
% ================================================================
```

Listing B.8 – Simulação Gráfica da Trajetória do VANT

B.9 Cálculo dos Ganhos de Realimentação com Alocação

```
%% CALCULO DOS GANHOS Ks AUMENTADOS - VIA LMI
K_z2 = lqrvialmi(Gz,Qz,Rz,x0,alfa,beta,theta)
K_psi2 = lqrvialmi(Gpsi,Qpsi,Rpsi,x0,alfa,beta,theta)
Kfi_y2 = lqrvialmi(Gfi_y,Qfi_y,Rfi_y,x0,alfa,beta,theta)
Ktheta_x2 = lqrvialmi(Gtheta_x,Qtheta_x,Rtheta_x,x0,alfa,beta,theta)
```

Listing B.9 – Cálculo dos Ganhos de Realimentação com Alocação

B.10 Declaração dos parâmetros incertos

```
%% PARAMETROS INCERTOS
%% Limite Inferior
tz1 = 0.4;
tpsi1 = 0.3;
Cx1 = 0.3;
Cy1 = 0.1;
%% Limite Superior
tz2 = 0.45;
tpsi2 = 0.4;
Cx2 = 0.9;
Cy2 = 0.5;
```

Listing B.10 – Declaração dos parâmetros incertos

B.11 Declaração das Matrizes dos Vértices

```
%% VARIAVEL Z
Az1 = [0 1;
       0 -1/tz1];
Az2 = [0 1;
       0 -1/tz2];
Az_a1 = [Az1 zeros([size(Az1,1),size(Bz1,2)]);-Cz zeros(size(Cz,1))];
Az_a2 = [Az2 zeros([size(Az2,1),size(Bz1,2)]);-Cz zeros(size(Cz,1))];
Az_a=[];
Az_a(:,:,1) = Az_a1;
Az_a(:,:,2) = Az_a2;
% ------------------------------------------------------------------------
Bz1 = [0;
       (Kz*zmax)/(tz1)];
Bz2 = [0;
       (Kz*zmax)/(tz2)];
Bz_a1 = [Bz1;-Dz];
Bz_a2 = [Bz2;-Dz];
Bz_a=[];
Bz_a(:,:,1) = Bz_a1;
Bz_a(:,:,2) = Bz_a2;
%% VARIAVEL PSI
Apsi1 = [0 1;
         0 -1/tpsi1];
Apsi2 = [0 1;
         0 -1/tpsi2];
Apsi_a1 = [Apsi1 zeros([size(Apsi1,1),size(Bpsi1,2)]);-Cpsi zeros(size(
    Cpsi,1))];
Apsi_a2 = [Apsi2 zeros([size(Apsi2,1),size(Bpsi1,2)]);-Cpsi zeros(size(
    Cpsi,1))];
Apsi_a=[];
Apsi_a(:,:,1) = Apsi_a1;
```

```
Apsi_a(:,:,2) = Apsi_a2;
% ------------------------------------------------------------------------
Bpsi1 = [0;
       (Kpsi*psimax)/(tpsi1)];
Bpsi2 = [0;
       (Kpsi*psimax)/(tpsi2)];
Bpsi_a1 = [Bpsi1;-Dpsi];
Bpsi_a2 = [Bpsi2;-Dpsi];
Bpsi_a=[];
Bpsi_a(:,:,1) = Bpsi_a1;
Bpsi_a(:,:,2) = Bpsi_a2;
%% VARIAVEL FI-Y
Afi_y1 = [0 1 0 0;
       -wfi^2 -2*zetafi*wfi 0 0;
         0 0 0 1;
         -g 0 0 -Cy1];
Afi_y2 = [0 1 0 0;
       -wfi^2 -2*zetafi*wfi 0 0;
         0 0 0 1;
         -g 0 0 -Cy2];
Afi_y_a1 = [Afi_y1 zeros([size(Afi_y1,1),size(Bfi_y,2)]);
          -Cfi_y zeros(size(Cfi_y,1))];
Afi_y_a2 = [Afi_y2 zeros([size(Afi_y2,1),size(Bfi_y,2)]);
          -Cfi_y zeros(size(Cfi_y,1))];
Afi_y_a=[];
Afi_y_a(:,:,1) = Afi_y_a1;
Afi_y_a(:,:,2) = Afi_y_a2;
%% VARIAVEL THETA-X
Atheta_x1 = [0 1 0 0;
       -wtheta^2 -2*zetatheta*wtheta 0 0;
          0 0 0 1;
          g 0 0 -Cx1];
Atheta_x2 = [0 1 0 0;
       -wtheta^2 -2*zetatheta*wtheta 0 0;
          0 0 0 1;
          g 0 0 -Cx2];
Atheta_x_a1 = [Atheta_x1 zeros([size(Atheta_x1,1),size(Btheta_x,2)]);
              -Ctheta_x zeros(size(Ctheta_x,1))];
Atheta_x_a2 = [Atheta_x2 zeros([size(Atheta_x2,1),size(Btheta_x,2)]);
              -Ctheta_x zeros(size(Ctheta_x,1))];
Atheta_x_a=[];
Atheta_x_a(:,:,1) = Atheta_x_a1;
Atheta_x_a(:,:,2) = Atheta_x_a2;
```

Listing B.11 – Declaração das Matrizes B dos vértices

APÊNDICE C – Códigos Completos do Projeto do Controlador

Este apêndice contém alguns *scripts* desenvolvidos para o projeto dos controladores LQR e LQR Robusto.

C.1 Função *lqrvialmi()* completa

```
%=====================================================================
%======================== LQR VIA LMI ================================
%=====================================================================

function K = lqrvialmi(SYS,Q,R,x0,alfa,beta,theta)
%% Tratando argumentos passados
A = SYS.A;
B = SYS.B;
C = SYS.C;
D = SYS.D;

n = size(A,1);
m = size(B,2);


%% LMI
setlmis([])

%% Declaracao de Variaveis
gama = lmivar(1,[1 1]);
[W1,n_W1] = lmivar(1,[n 1]);
[W2,n_W2] = lmivar(2,[m n]);

%% ===========> Declaracao de LMIs
%% 1 LMI
lmiterm([-1 1 1 gama],1,1);
lmiterm([-1 2 1 0],x0);
lmiterm([-1 2 2 W1],1,1);
%% 2 LMI
lmiterm([2 1 1 W1],A,1,'s');
lmiterm([2 1 1 W2],B,1,'s');
lmiterm([2 2 1 W1],Q^0.5,1);
lmiterm([2 3 1 W2],R^0.5,1);
lmiterm([2 2 2 0],-1);
```

```
lmiterm([2 3 2 0],0);
lmiterm([2 3 3 0],-1);

%% 3 LMI
lmiterm([-3 1 1 gama],1,1);
%% 4 LMI
lmiterm([-4 1 1 W1],1,1);

%% ALOCACAO DE POLOS
% Primeira Faixa
lmiterm([5 1 1 W1],A,1,'s');
lmiterm([5 1 1 W2],B,1,'s');
lmiterm([5 1 1 W1],2*alfa,1);

% Segunda Faixa
lmiterm([6 1 1 W1],-A,1,'s');
lmiterm([6 1 1 W2],-B,1,'s');
lmiterm([6 1 1 W1],-2*beta,1);

% Setor
lmiterm([7 1 1 W1],sind(theta)*A,1,'s');
lmiterm([7 1 1 W2],sind(theta)*B,1,'s');

lmiterm([7 1 2 W1],cosd(theta)*A,1);
lmiterm([7 1 2 W1],-cosd(theta),A');
lmiterm([7 1 2 W2],cosd(theta)*B,1);
lmiterm([7 1 2 -W2],-cosd(theta),B');

lmiterm([7 2 2 W1],sind(theta)*A,1,'s');
lmiterm([7 2 2 W2],sind(theta)*B,1,'s');
%% Solver
LQR_lmi = getlmis;
c = [1 zeros(1,n_W2-1)]';
options = [1e-8,300,1e9,20,1];

[c_opt,x_opt] = mincx(LQR_lmi,c,options);

%% Obtencao dos Resultados
gama_opt = dec2mat(LQR_lmi,x_opt,gama);
W1_opt = dec2mat(LQR_lmi,x_opt,W1);
W2_opt = dec2mat(LQR_lmi,x_opt,W2);

K = W2_opt*inv(W1_opt);
```

Listing C.1 – Função *lqrvialmi()* Completa

C.2 Função *lqrrobusto()* completa

```
%=================================================================
%=================== LQR ROBUSTO VIA LMI =======================
%=================================================================

function K = lqrrobusto(A,B,C,D,Q,R,x0,alfa,beta,theta)
%% Dimensionamento das Matrizes
[n,p,ra] = size(A);
[p,m,rb] = size(B);

%% LMI
setlmis([])

%% Declara  o de Vari veis
gama = lmivar(1,[1 1]);
[W1,n_W1] = lmivar(1,[n 1]);
[W2,n_W2] = lmivar(2,[m n]);

%% ===========> Declara  o de LMIs
%% 1 LMI
lmiterm([-1 1 1 gama],1,1);
lmiterm([-1 2 1 0],x0);
lmiterm([-1 2 2 W1],1,1);
%% 2 LMI
lmiterm([-2 1 1 gama],1,1);
%% 3 LMI
lmiterm([-3 1 1 W1],1,1);

%% LMIs DO LQR
q = 4;

for j=1:ra
    for i=1:rb
        lmiterm([q 1 1 W1],A(:,:,j),1,'s');
        lmiterm([q 1 1 W2],B(:,:,i),1,'s');
        lmiterm([q 1 1 W1],Q^0.5,1);
        lmiterm([q 3 1 W2],R^0.5,1);
        lmiterm([q 2 2 0],-1);
        lmiterm([q 3 2 0],0);
        lmiterm([q 3 3 0],-1);

        q = q + 1;
    end
end
%% LMIs da ALOCA  O DE POLOS
for j=1:ra
```

```
    for i=1:rb
        % Primeira Faixa
        lmiterm([q 1 1 W1],A(:,:,j),1,'s');
        lmiterm([q 1 1 W2],B(:,:,i),1,'s');
        lmiterm([q 1 1 W1],2*alfa,1);

        % Segunda Faixa
        q = q + 1;
        lmiterm([q 1 1 W1],-A(:,:,j),1,'s');
        lmiterm([q 1 1 W2],-B(:,:,i),1,'s');
        lmiterm([q 1 1 W1],-2*beta,1);

        % Setor C nico
        q = q + 1;
        lmiterm([q 1 1 W1],sind(theta)*A(:,:,j),1,'s');
        lmiterm([q 1 1 W2],sind(theta)*B(:,:,i),1,'s');
        lmiterm([q 1 2 W1],cosd(theta)*A(:,:,j),1);
        lmiterm([q 1 2 W1],-cosd(theta),A(:,:,j)');
        lmiterm([q 1 2 W2],cosd(theta)*B(:,:,i),1);
        lmiterm([q 1 2 -W2],-cosd(theta),B(:,:,i)');
        lmiterm([q 2 2 W1],sind(theta)*A(:,:,j),1,'s');
        lmiterm([q 2 2 W2],sind(theta)*B(:,:,i),1,'s');

        q = q + 1;
    end
end
%% Solver
LQR_lmi = getlmis;
c = [1 zeros(1,n_W2-1)]';
options = [1e-8,300,1e9,20,1];

[c_opt,x_opt] = mincx(LQR_lmi,c,options);

%% Obten  o dos Resultados
gama_opt = dec2mat(LQR_lmi,x_opt,gama);
W1_opt = dec2mat(LQR_lmi,x_opt,W1);
W2_opt = dec2mat(LQR_lmi,x_opt,W2);

K = W2_opt*inv(W1_opt);
```

Listing C.2 – Função *lqrrobusto()* Completa

C.3 Projeto do Controlador LQR com Restrição

```
%
    ================================================================================

% ============ PROJETO DE CONTROLE COM SISTEMA DESACOPLADO
    ================
%
    ================================================================================

clear all;
clc;
close all;
%% Tipo de Trajet ria
% circulo = 1 /// oito inclinado = 0
tipo = 1;
if tipo == 1
    chave_circular = 1;
    chave_oito = 0;
else
    chave_circular = 0;
    chave_oito = 1;
end
%% ==========> Parametros com Kfi e Ktheta = 1.5 <============
Cx = 0.3;
Cy = 0.1;
g = 9.8;
Kz = 1.3;
Kpsi = 1;
Kfi = 1.5;
Ktheta = 1.5;
tz = 0.4;
tpsi = 0.3;
zmax = 1.0;
fimax = 0.26;
thetamax = 0.26;
psimax = 1.74;
wfi = 4.47;
wtheta = 4.47;
zetafi = 0.5;
zetatheta = 0.5;

%% Par metros LIMITE INFERIOR
% zmax = 1;
% psimax = 1.74;
% fimax = 0.26;
% thetamax = 0.26;
%
```

```
% %Limite superior
% zmax = 2;
% psimax = 5.15;
% fimax = 0.45;
% thetamax = 0.52;
%% ================================================
%  ========== SISTEMA LINEARIZADO ================
%  ================================================
s = tf([1 0],[1]);

%% VARI VEL FI-Y
Afi_y = [0 1 0 0;
       -wfi^2 -2*zetafi*wfi 0 0;
         0 0 0 1;
         -g 0 0 -Cy];
Bfi_y = [0;
        Kfi*fimax*wfi^2;
        0;
        0];
Cfi_y = [0 0 1 0];
Dfi_y = [0];

Afi_y_a = [Afi_y zeros([size(Afi_y,1),size(Bfi_y,2)]);-Cfi_y zeros(size(
    Cfi_y,1))];
Bfi_y_a = [Bfi_y;-Dfi_y];
Cfi_y_a = [Cfi_y zeros(size(Cfi_y,1))];
Dfi_y_a = [0];

Gfi_y = ss(Afi_y_a,Bfi_y_a,Cfi_y_a,Dfi_y_a);
%% VARI VEL THETA-X
Atheta_x = [0 1 0 0;
       -wtheta^2 -2*zetatheta*wtheta 0 0;
          0 0 0 1;
          g 0 0 -Cx];
Btheta_x = [0;
        Ktheta*thetamax*wtheta^2;
          0;
          0];
Ctheta_x = [0 0 1 0];
Dtheta_x = [0];

Atheta_x_a = [Atheta_x zeros([size(Atheta_x,1),size(Btheta_x,2)]);
              -Ctheta_x zeros(size(Ctheta_x,1))];
Btheta_x_a = [Btheta_x;-Dtheta_x];
Ctheta_x_a = [Ctheta_x zeros(size(Ctheta_x,1))];
Dtheta_x_a = [0];

```

```
Gtheta_x = ss(Atheta_x_a,Btheta_x_a,Ctheta_x_a,Dtheta_x_a);
%% VARI VEL Z
Az = [0 1;
      0 -1/tz];
Bz = [0;
      (Kz*zmax)/(tz)];
Cz = [1 0];
Dz = 0;

Az_a = [Az zeros([size(Az,1),size(Bz,2)]);
        -Cz zeros(size(Cz,1))];
Bz_a = [Bz;-Dz];
Cz_a = [Cz zeros(size(Cz,1))];
Dz_a = 0;

Gz = ss(Az_a,Bz_a,Cz_a,Dz_a);
%% VARI VEL PSI
Apsi = [0 1;
        0 -1/tpsi];
Bpsi = [0;
       (Kpsi*psimax)/(tpsi)];
Cpsi = [1 0];
Dpsi = 0;

Apsi_a = [Apsi zeros([size(Apsi,1),size(Bpsi,2)]);
          -Cpsi zeros(size(Cpsi,1))];
Bpsi_a = [Bpsi;-Dpsi];
Cpsi_a = [Cpsi zeros(size(Cpsi,1))];
Dpsi_a = 0;

Gpsi = ss(Apsi_a,Bpsi_a,Cpsi_a,Dpsi_a);
%% ================================================
%  ========= PROJETO DOS CONTROLADORES ===========
%  ================================================
%% ================ MATRIZES DE PONDERA  O Q e R ====================
% VARI VEL Z
Qz = [1 0 0;
      0 1 0;
      0 0 120];
Rz = 1;
% VARI VEL PSI
Qpsi = [1 0 0;
        0 1 0;
        0 0 0.01];
Rpsi = 24.5;
% VARI VEL FI-Y
Qfi_y = [1 0 0 0 0;
```

```
135          0 1 0 0 0;
136          0 0 1 0 0;
137          0 0 0 1 0;
138          0 0 0 0 240]; %240
139 Rfi_y = 0.1;
140 % VARI VEL THETA-X
141 Qtheta_x = [1 0 0 0 0;
142              0 1 0 0 0;
143              0 0 1 0 0;
144              0 0 0 1 0;
145              0 0 0 0 1000];
146 Rtheta_x = 3;
147 %% C LCULO DOS GANHOS Ks AUMENTADOS - VIA LMI
148
149 %Estado inicial
150 x0 = [1 1 1]';
151 %% Z
152 alfa = 0; beta = 3; theta = 43; %Aloca  o de Polos
153 disp('CONTROLE LQR')
154 K_z2 = lqrvialmi(Gz,Qz,Rz,x0,alfa,beta,theta)
155 %% PSI
156 alfa = 0.4; beta = 4; theta = 60; %Aloca  o de Polos
157 K_psi2 = lqrvialmi(Gpsi,Qpsi,Rpsi,x0,alfa,beta,theta)
158 %% Estado inicial
159 x0 = [1 1 1 1 1]';
160 %% FI
161 alfa = 0.05; beta = 22; theta = 85; %Aloca  o de Polos
162 Kfi_y2 = lqrvialmi(Gfi_y,Qfi_y,Rfi_y,x0,alfa,beta,theta)
163 %% THETA
164 alfa = 0.05; beta = 22; theta = 85; %Aloca  o de Polos
165 Ktheta_x2 = lqrvialmi(Gtheta_x,Qtheta_x,Rtheta_x,x0,alfa,beta,theta)
166 %% ============================================================
167 %%  ===== GANHO DE REALIMENTA  O DE ESTADO E INTEGRAL ==========
168 %%  ============================================================
169 %% Z
170 Ki_z = K_z2(:,3);
171 K_z = K_z2(:,1:2);
172 %% PSI
173 Ki_psi = K_psi2(:,3);
174 K_psi = K_psi2(:,1:2);
175 %% FI-Y
176 Ki_fi_y = Kfi_y2(:,5);
177 Kfi_y = Kfi_y2(:,1:4);
178 %% THETA-X
179 Ki_theta_x = Ktheta_x2(:,5);
180 Ktheta_x = Ktheta_x2(:,1:4);
181 %% Executar arquivo .slx
```

```
182 sim('Malha_de_Controle_Drone')
183 %% ================================================================
184 %% ================ SISTEMA EM MALHA FECHADA ======================
185 %% ================================================================
186 %% Tamanho da Fonte
187 fontsize = 20;
188 fontsize2 = 18;
189 %% Z
190 Af_z = [Az+Bz*K_z Bz*Ki_z;
191         -(Cz+Dz*K_z) -Dz*Ki_z];
192 Bf_z = [zeros(size(Az,1),1);eye(1,1)];
193 Cf_z = [Cz+Dz*Kz Dz*Ki_z];
194 Df_z = Dz;
195
196 SYSz = ss(Af_z,Bf_z,Cf_z,Df_z)
197 polosZ = eig(Af_z)
198 %% PSI
199 Af_psi = [Apsi+Bpsi*K_psi Bpsi*Ki_psi;
200         -(Cpsi+Dpsi*K_psi) -Dpsi*Ki_psi];
201 Bf_psi = [zeros(size(Apsi,1),1);eye(1,1)];
202 Cf_psi = [Cpsi+Dpsi*Kpsi Dpsi*Ki_psi];
203 Df_psi = Dpsi;
204
205 SYSpsi = ss(Af_psi,Bf_psi,Cf_psi,Df_psi)
206 polosPSI = eig(Af_psi)
207 %% FI
208 Af_fi = [Afi_y+Bfi_y*Kfi_y Bfi_y*Ki_fi_y;
209         -(Cfi_y+Dfi_y*Kfi_y) -Dfi_y*Ki_fi_y];
210 Bf_fi = [zeros(size(Afi_y,1),1);eye(1,1)];
211 Cf_fi = [Cfi_y+Dfi_y*Kfi_y Dfi_y*Ki_fi_y];
212 Df_fi = Dfi_y;
213
214 SYSfi = ss(Af_fi,Bf_fi,Cf_fi,Df_fi)
215 polosFI = eig(Af_fi)
216 %% THETA
217 Af_theta = [Atheta_x+Btheta_x*Ktheta_x Btheta_x*Ki_theta_x;
218         -(Ctheta_x+Dtheta_x*Ktheta_x) -Dtheta_x*Ki_theta_x];
219 Bf_theta = [zeros(size(Atheta_x,1),1);eye(1,1)];
220 Cf_theta = [Ctheta_x+Dtheta_x*Ktheta_x Dtheta_x*Ki_theta_x];
221 Df_theta = Dtheta_x;
222
223 SYStheta = ss(Af_theta,Bf_theta,Cf_theta,Df_theta)
224 polosTHETA = eig(Af_theta)
225
226 %% ====================== PZMAP ==============================
227
228 %% Z
```

```
figure
pzplot(SYSz,'r')

%Ajustes
title('Polos de Malha Fechada do Sistema','FontSize',fontsize)
%Configura  es
xlabel('Eixo Real','FontSize',fontsize)
ylabel('Eixo Imagin rio','FontSize',fontsize)
grid on
hm = findobj(gca, 'Type', 'Line');          % Handle To 'Line' Objects
hm(2).MarkerSize = 20;                      %    Zero     Marker
hm(3).MarkerSize = 20;                      %    Pole     Marker
hm(2).LineWidth = 5;                       %    Zero     Marker
hm(3).LineWidth = 5;                       %    Pole     Marker
%% PSI
hold on
pzplot(SYSpsi,'b')
hm = findobj(gca, 'Type', 'Line');          % Handle To 'Line' Objects
hm(2).MarkerSize = 20;                      %    Zero     Marker
hm(3).MarkerSize = 20;                      %    Pole     Marker
hm(2).LineWidth = 5;                       %    Zero     Marker
hm(3).LineWidth = 5;                       %    Pole     Marker

%% FI
hold on
pzplot(SYSfi,'k')
hm = findobj(gca, 'Type', 'Line');          % Handle To 'Line' Objects
hm(2).MarkerSize = 20;                      %    Zero     Marker
hm(3).MarkerSize = 20;                      %    Pole     Marker
hm(2).LineWidth = 5;                       %    Zero     Marker
hm(3).LineWidth = 5;                       %    Pole     Marker

%% THETA
hold on
pzplot(SYStheta,'m')
hm = findobj(gca, 'Type', 'Line');          % Handle To 'Line' Objects
hm(2).MarkerSize = 20;                      %    Zero     Marker
hm(3).MarkerSize = 20;                      %    Pole     Marker
hm(2).LineWidth = 5;                       %    Zero     Marker
hm(3).LineWidth = 5;                       %    Pole     Marker

legend('Subsistema z','Subsistema \psi','Subsistema \phi','Subsistema \
    theta')

%% ============================================================
%%  ================= RESULTADOS DA SIMULA  O =================
%%  ============================================================
```

```
%% ===============> Plot da Trajet ria em 3D <====================
if tipo == 1
    %REFER NCIA CIRCULAR
    figure
    t = tout;
    xref = sin(0.8*t);
    yref = cos(0.8*t);
    zref = 1.2*ones([1 length(t)]);
    psiref = 0;
    plot3(xref,yref,zref,'r','LineWidth',2);
    title('Trajet ria Circular','FontSize',fontsize)

    xref = sin(0.8*t-0.65);
    yref = cos(0.8*t-0.65);
else
    %REFER NCIA OITO INCLINADO
    figure
    t = tout;
    xref = 0.5*sin(0.8*t);
    yref = sin(0.4*t);
    zref = 1.2 + 0.5*sin(0.4*t);
    psiref = -((pi/6)*sin(0.4*t));
    plot3(xref,yref,zref,'r','LineWidth',2);
    title('Trajet ria em Formato de Oito Inclinado','FontSize',fontsize
    )

    xref = 0.5*sin(0.8*t-0.65);
    yref = sin(0.4*t-0.33);
    zref = 1.2 + 0.5*sin(0.4*t-0.4);
    psiref = -((pi/6)*sin(0.4*t+0.9));
end

%Simula  o
hold on
plot3(x,y,z,'LineWidth',2);
xlabel('Eixo x','FontSize',fontsize)
ylabel('Eixo y','FontSize',fontsize)
zlabel('Eixo z','FontSize',fontsize)
lgd = legend('Trajet ria Refer ncia','Trajet ria Realizada pelo Drone
    ')
lgd.FontSize = fontsize2;
grid on
axis([-1.5 1.5 -1.5 1.5 0 2.2]) %Ajuste do Gr fico
% ====================================================================

%% TEMPOR RIO
%close all
```

```
320
321
322 %% Plot dos Sinais de Saida e Sinais de Controle Individuais
323 %% X
324 figure
325 plot(tout,x,'b','LineWidth',2)
326 hold on
327 plot(t,xref,'r','LineWidth',2)
328 lgd = legend('X da simula  o','X de refer ncia')
329 lgd.FontSize = fontsize2;
330 title('Sa da X','FontSize',fontsize)
331 grid on
332 axis([0 20 -3 3])
333 xlabel('Tempo (s)','FontSize',fontsize)
334 ylabel('Posi  o x (m)','FontSize',fontsize)
335
336 figure
337 plot(tout,u_theta,'LineWidth',2)
338 title('Sinal de Controle - U_{\theta}','FontSize',fontsize)
339 grid on
340 %% Y
341 figure
342 plot(tout,y,'b','LineWidth',2)
343 hold on
344 plot(t,yref,'r','LineWidth',2)
345 lgd = legend('Y da simula  o','Y de refer ncia','FontSize',fontsize)
346 lgd.FontSize = fontsize2;
347 title('Sa da Y','FontSize',fontsize)
348 grid on
349 axis([0 20 -1.5 1.5])
350 xlabel('Tempo (s)','FontSize',fontsize)
351 ylabel('Posi  o y (m)','FontSize',fontsize)
352
353 figure
354 plot(tout,u_fi,'LineWidth',2)
355 title('Sinal de Controle - U_{\phi}','FontSize',fontsize)
356 grid on
357 %% Z
358 figure
359 plot(tout,z,'b','LineWidth',2)
360 hold on
361 plot(t,zref,'r','LineWidth',2)
362 lgd = legend('Z da simula  o','Z de refer ncia','FontSize',fontsize)
363 lgd.FontSize = fontsize2;
364 title('Sa da Z','FontSize',fontsize)
365 grid on
366 xlabel('Tempo (s)','FontSize',fontsize)
```

```
ylabel('Posi  o z (m)','FontSize',fontsize)
axis([0 10 0 1.5]) %circulo
axis([0 30 0 1.8]) %oito

figure
plot(tout,uz,'LineWidth',2)
title('Sinal de Controle - Uz','FontSize',fontsize)
grid on

%% PSI
figure
plot(tout,e_psi,'b','LineWidth',2)
hold on
plot(t,psiref,'r','LineWidth',2)
lgd = legend('\psi da simula  o','\psi de refer ncia')
lgd.FontSize = fontsize2;
title('Sa da \psi','FontSize',fontsize)
grid on
xlabel('Tempo (s)','FontSize',fontsize)
ylabel(' ngulo  \psi (rad)','FontSize',fontsize)
axis([0 30 -0.6 0.6])

figure
plot(tout,u_psi,'LineWidth',2)
title('Sinal de Controle - U_{\psi}','FontSize',fontsize)
grid on
```

Listing C.3 – Projeto do Controlador LQR Completo

C.4 Projeto do Controlador LQR Robusto com Restrição

```
% =======================================================
% === PROJETO DE CONTROLE CONSIDERANDO 4 SUBSISTEMAS ===
% =======================================================
clear all;
clc;
close all;
%% Tipo de Trajetoria
% circulo = 1 /// oito inclinado = 0
tipo = 1;
if tipo == 1
    chave_circular = 1;
    chave_oito = 0;
else
    chave_circular = 0;
    chave_oito = 1;
end
```

```
17 %% ==========> Parametros do AR.Drone 2.0 <============
18 Cx = 0.3;
19 Cy = 0.1;
20 g = 9.8;
21 Kz = 1.3;
22 Kpsi = 1;
23 Kfi = 1.5;
24 Ktheta = 1.5;
25 tz = 0.4;
26 tpsi = 0.3;
27 zmax = 1.0;
28 fimax = 0.26;
29 thetamax = 0.26;
30 psimax = 1.74;
31 wfi = 4.47;
32 wtheta = 4.47;
33 zetafi = 0.5;
34 zetatheta = 0.5;
35
36 %% PARAMETROS INCERTOS
37 %% Limite Inferior
38 tz1 = 0.4;
39 tpsi1 = 0.3;
40 Cx1 = 0.3;
41 Cy1 = 0.1;
42 %% Limite Superior
43 tz2 = 0.45;
44 tpsi2 = 0.4;
45 Cx2 = 0.9;
46 Cy2 = 0.5;
47
48
49 tz2 = 0.8;
50 tpsi2 = 0.4;
51 Cx2 = 1;
52 Cy2 = 3;
53
54
55 %% ================================================
56 %  ========== SISTEMA LINEARIZADO ================
57 %  ================================================
58 s = tf([1 0],[1]);
59
60 %% VARIAVEL Z
61 Az1 = [0 1;
62       0 -1/tz1];
63 Az2 = [0 1;
```

```
64        0 -1/tz2];
65
66 Bz1 = [0;
67        (Kz*zmax)/(tz1)];
68 Bz2 = [0;
69        (Kz*zmax)/(tz2)];
70 Cz = [1 0];
71 Dz = 0;
72
73 Az_a1 = [Az1 zeros([size(Az1,1),size(Bz1,2)]);-Cz zeros(size(Cz,1))];
74 Az_a2 = [Az2 zeros([size(Az2,1),size(Bz1,2)]);-Cz zeros(size(Cz,1))];
75 Az_a=[];
76 Az_a(:,:,1) = Az_a1;
77 Az_a(:,:,2) = Az_a2;
78
79 Bz_a1 = [Bz1;-Dz];
80 Bz_a2 = [Bz2;-Dz];
81 Bz_a=[];
82 Bz_a(:,:,1) = Bz_a1;
83 Bz_a(:,:,2) = Bz_a2;
84
85 Cz_a = [Cz zeros(size(Cz,1))];
86 %% VARIAVEL PSI
87 Apsi1 = [0 1;
88         0 -1/tpsi1];
89 Apsi2 = [0 1;
90         0 -1/tpsi2];
91
92 Bpsi1 = [0;
93        (Kpsi*psimax)/(tpsi1)];
94 Bpsi2 = [0;
95        (Kpsi*psimax)/(tpsi2)];
96 Cpsi = [1 0];
97 Dpsi = 0;
98
99 Apsi_a1 = [Apsi1 zeros([size(Apsi1,1),size(Bpsi1,2)]);-Cpsi zeros(size(
      Cpsi,1))];
100 Apsi_a2 = [Apsi2 zeros([size(Apsi2,1),size(Bpsi1,2)]);-Cpsi zeros(size(
      Cpsi,1))];
101 Apsi_a=[];
102 Apsi_a(:,:,1) = Apsi_a1;
103 Apsi_a(:,:,2) = Apsi_a2;
104
105 Bpsi_a1 = [Bpsi1;-Dpsi];
106 Bpsi_a2 = [Bpsi2;-Dpsi];
107 Bpsi_a=[];
108 Bpsi_a(:,:,1) = Bpsi_a1;
```

```
Bpsi_a(:,:,2) = Bpsi_a2;

Cpsi_a = [Cpsi zeros(size(Cpsi,1))];
%% VARIAVEL FI-Y
Afi_y1 = [0 1 0 0;
       -wfi^2 -2*zetafi*wfi 0 0;
         0 0 0 1;
         -g 0 0 -Cy1];
Afi_y2 = [0 1 0 0;
       -wfi^2 -2*zetafi*wfi 0 0;
         0 0 0 1;
         -g 0 0 -Cy2];

Bfi_y = [0;Kfi*fimax*wfi^2;0;0];

Cfi_y = [0 0 1 0];
Dfi_y = [0];

Afi_y_a1 = [Afi_y1 zeros([size(Afi_y1,1),size(Bfi_y,2)]);
          -Cfi_y zeros(size(Cfi_y,1))];
Afi_y_a2 = [Afi_y2 zeros([size(Afi_y2,1),size(Bfi_y,2)]);
          -Cfi_y zeros(size(Cfi_y,1))];

Afi_y_a=[];
Afi_y_a(:,:,1) = Afi_y_a1;
Afi_y_a(:,:,2) = Afi_y_a2;

Bfi_y_a = [Bfi_y;-Dfi_y];

Cfi_y_a = [Cfi_y zeros(size(Cfi_y,1))];
%% VARIAVEL THETA-X
Atheta_x1 = [0 1 0 0;
       -wtheta^2 -2*zetatheta*wtheta 0 0;
          0 0 0 1;
          g 0 0 -Cx1];
Atheta_x2 = [0 1 0 0;
       -wtheta^2 -2*zetatheta*wtheta 0 0;
          0 0 0 1;
          g 0 0 -Cx2];

Btheta_x = [0;Ktheta*thetamax*wtheta^2;0;0];

Ctheta_x = [0 0 1 0];
Dtheta_x = [0];

Atheta_x_a1 = [Atheta_x1 zeros([size(Atheta_x1,1),size(Btheta_x,2)]);
              -Ctheta_x zeros(size(Ctheta_x,1))];
```

```
156 Atheta_x_a2 = [Atheta_x2 zeros([size(Atheta_x2,1),size(Btheta_x,2)]);
157               -Ctheta_x zeros(size(Ctheta_x,1))];
158
159 Atheta_x_a=[];
160 Atheta_x_a(:,:,1) = Atheta_x_a1;
161 Atheta_x_a(:,:,2) = Atheta_x_a2;
162
163 Btheta_x_a = [Btheta_x;-Dtheta_x];
164
165 Ctheta_x_a = [Ctheta_x zeros(size(Ctheta_x,1))];
166 %% ================================================
167 %%  ========= PROJETO DOS CONTROLADORES ===========
168 %%  ================================================
169 %% ================ MATRIZES DE PONDERACAO Q e R ====================
170 % VARIAVEL Z
171 Qz = [1 0 0;
172       0 1 0;
173       0 0 120]; %120
174 Rz = 1; %17
175 % VARIAVEL PSI
176 Qpsi = [1 0 0;
177         0 1 0;
178         0 0 0.01]; %1
179 Rpsi = 24.5; %15
180 % VARIAVEL FI-Y
181 Qfi_y = [1 0 0 0 0;
182          0 1 0 0 0;
183          0 0 1 0 0;
184          0 0 0 1 0;
185          0 0 0 0 240];%240
186 Rfi_y = 0.1;%0.1
187 % VARIAVEL THETA-X
188 Qtheta_x = [1 0 0 0 0;
189             0 1 0 0 0;
190             0 0 1 0 0;
191             0 0 0 1 0;
192             0 0 0 0 1000];
193 Rtheta_x = 3;
194 %% CALCULO DO GANHO K AUMENTADO - VIA LMI
195 disp('CONTROLE VIA LMI')
196 %Estado inicial
197 x0 = [1 1 1]';
198
199 alfa = 0; beta = 3; theta = 45; %Alocacao de Polos
200 K_z2 = lqrrobusto(Az_a,Bz_a,Cz_a,Dz,Qz,Rz,x0,alfa,beta,theta)
201
202 alfa = 0.4; beta = 4; theta = 60; %Alocacao de Polos
```

```
203 K_psi2 = lqrrobusto(Apsi_a,Bpsi_a,Cpsi_a,Dpsi,Qpsi,Rpsi,x0,alfa,beta,
        theta)
204
205 %Estado inicial
206 x0 = [1 1 1 1 1]';
207
208 alfa = 0.05; beta = 22; theta = 85; %Alocacao de Polos
209 Kfi_y2 = lqrrobusto(Afi_y_a,Bfi_y_a,Cfi_y_a,Dfi_y,Qfi_y,Rfi_y,x0,alfa,
        beta,theta)
210
211 alfa = 0.05; beta = 22; theta = 85; %Alocacao de Polos
212 Ktheta_x2 = lqrrobusto(Atheta_x_a,Btheta_x_a,Ctheta_x_a,Dtheta_x,
        Qtheta_x,Rtheta_x,x0,alfa,beta,theta)
213
214 %% ================================================================
215 %  ===== GANHO DE REALIMENTACAO DE ESTADO E INTEGRAL ===========
216 %  ================================================================
217 %% Z
218 Ki_z = K_z2(:,3);
219 K_z = K_z2(:,1:2);
220 %% PSI
221 Ki_psi = K_psi2(:,3);
222 K_psi = K_psi2(:,1:2);
223 %% FI-Y
224 Ki_fi_y = Kfi_y2(:,5);
225 Kfi_y = Kfi_y2(:,1:4);
226 %% THETA-X
227 Ki_theta_x = Ktheta_x2(:,5);
228 Ktheta_x = Ktheta_x2(:,1:4);
229 %% PARAMETROS VARIAVEIS
230 %% Limite Inferior
231 tz = tz1;
232 tpsi = tpsi1;
233 Cx = Cx1;
234 Cy = Cy1;
235 %% Limite Superior
236 tz = tz2;
237 tpsi = tpsi2;
238 Cx = Cx2;
239 Cy = Cy2;
240
241 %% Valor escolhido
242 tz = 0.4;
243 tpsi = 0.3;
244 Cx = 0.3;
245 Cy = 0.1;
246
```

```
247
248 %% Executar arquivo .slx
249 sim('Malha_de_Controle_Drone')
250
251 %% ================================================================
252 %% ================ SISTEMA EM MALHA FECHADA ======================
253 %% ================================================================
254 %% Tamanho da Fonte
255 fontsize = 20;
256 fontsize2 = 18;
257 %% Z
258 Af_z = [Az1+Bz1*K_z Bz1*Ki_z;
259         -(Cz+Dz*K_z) -Dz*Ki_z];
260 Bf_z = [zeros(size(Az1,1),1);eye(1,1)];
261 Cf_z = [Cz+Dz*Kz Dz*Ki_z];
262 Df_z = Dz;
263
264 SYSz = ss(Af_z,Bf_z,Cf_z,Df_z);
265 polosZ = eig(Af_z)
266 %% PSI
267 Af_psi = [Apsi1+Bpsi1*K_psi Bpsi1*Ki_psi;
268         -(Cpsi+Dpsi*K_psi) -Dpsi*Ki_psi];
269 Bf_psi = [zeros(size(Apsi1,1),1);eye(1,1)];
270 Cf_psi = [Cpsi+Dpsi*Kpsi Dpsi*Ki_psi];
271 Df_psi = Dpsi;
272
273 SYSpsi = ss(Af_psi,Bf_psi,Cf_psi,Df_psi);
274 polosPSI = eig(Af_psi)
275 %% FI
276 Af_fi = [Afi_y1+Bfi_y*Kfi_y Bfi_y*Ki_fi_y;
277         -(Cfi_y+Dfi_y*Kfi_y) -Dfi_y*Ki_fi_y];
278 Bf_fi = [zeros(size(Afi_y1,1),1);eye(1,1)];
279 Cf_fi = [Cfi_y+Dfi_y*Kfi_y Dfi_y*Ki_fi_y];
280 Df_fi = Dfi_y;
281
282 SYSfi = ss(Af_fi,Bf_fi,Cf_fi,Df_fi);
283 polosFI = eig(Af_fi)
284 %% THETA
285 Af_theta = [Atheta_x1+Btheta_x*Ktheta_x Btheta_x*Ki_theta_x;
286         -(Ctheta_x+Dtheta_x*Ktheta_x) -Dtheta_x*Ki_theta_x];
287 Bf_theta = [zeros(size(Atheta_x1,1),1);eye(1,1)];
288 Cf_theta = [Ctheta_x+Dtheta_x*Ktheta_x Dtheta_x*Ki_theta_x];
289 Df_theta = Dtheta_x;
290
291 SYStheta = ss(Af_theta,Bf_theta,Cf_theta,Df_theta);
292 polosTHETA = eig(Af_theta)
293
```

```
294 %% ====================== PZMAP ============================
295
296 %% Z
297 figure
298 pzplot(SYSz,'r')
299
300 %Ajustes
301 title('Polos de Malha Fechada do Sistema','FontSize',fontsize)
302 %Configura  es
303 xlabel('Eixo Real','FontSize',fontsize)
304 ylabel('Eixo Imagin rio','FontSize',fontsize)
305 grid on
306 hm = findobj(gca, 'Type', 'Line');          % Handle To 'Line' Objects
307 hm(2).MarkerSize = 20;                      %    Zero     Marker
308 hm(3).MarkerSize = 20;                      %    Pole     Marker
309 hm(2).LineWidth = 5;                      %    Zero     Marker
310 hm(3).LineWidth = 5;                      %    Pole     Marker
311 %% PSI
312 hold on
313 pzplot(SYSpsi,'b')
314 hm = findobj(gca, 'Type', 'Line');          % Handle To 'Line' Objects
315 hm(2).MarkerSize = 20;                      %    Zero     Marker
316 hm(3).MarkerSize = 20;                      %    Pole     Marker
317 hm(2).LineWidth = 5;                      %    Zero     Marker
318 hm(3).LineWidth = 5;                      %    Pole     Marker
319
320 %% FI
321 hold on
322 pzplot(SYSfi,'k')
323 hm = findobj(gca, 'Type', 'Line');          % Handle To 'Line' Objects
324 hm(2).MarkerSize = 20;                      %    Zero     Marker
325 hm(3).MarkerSize = 20;                      %    Pole     Marker
326 hm(2).LineWidth = 5;                      %    Zero     Marker
327 hm(3).LineWidth = 5;                      %    Pole     Marker
328
329 %% THETA
330 hold on
331 pzplot(SYStheta,'m')
332 hm = findobj(gca, 'Type', 'Line');          % Handle To 'Line' Objects
333 hm(2).MarkerSize = 20;                      %    Zero     Marker
334 hm(3).MarkerSize = 20;                      %    Pole     Marker
335 hm(2).LineWidth = 5;                      %    Zero     Marker
336 hm(3).LineWidth = 5;                      %    Pole     Marker
337
338 legend('Subsistema z','Subsistema \psi','Subsistema \phi','Subsistema \
    theta')
339
```

```
340
341 %% ================================================================
342 %%  ================== RESULTADOS DA SIMULA  O ==================
343 %%  ================================================================
344 %% ===============> Plot da Trajet ria em 3D <====================
345 if tipo == 1
346     %REFER NCIA CIRCULAR
347     figure
348     t = tout;
349     xref = sin(0.8*t);
350     yref = cos(0.8*t);
351     zref = 1.2*ones([1 length(t)]);
352     psiref = 0;
353     plot3(xref,yref,zref,'r','LineWidth',2);
354     title('Trajet ria Circular','FontSize',fontsize)
355
356     xref = sin(0.8*t-0.65);
357     yref = cos(0.8*t-0.65);
358 else
359     %REFER NCIA OITO INCLINADO
360     figure
361     t = tout;
362     xref = 0.5*sin(0.8*t);
363     yref = sin(0.4*t);
364     zref = 1.2 + 0.5*sin(0.4*t);
365     psiref = -((pi/6)*sin(0.4*t));
366     plot3(xref,yref,zref,'r','LineWidth',2);
367     title('Trajet ria em Formato de Oito Inclinado','FontSize',fontsize
    )
368
369     xref = 0.5*sin(0.8*t-0.65);
370     yref = sin(0.4*t-0.33);
371     zref = 1.2 + 0.5*sin(0.4*t-0.4);
372     psiref = -((pi/6)*sin(0.4*t+0.9));
373 end
374
375 %Simula  o
376 hold on
377 plot3(x,y,z,'LineWidth',2);
378 xlabel('Eixo x','FontSize',fontsize)
379 ylabel('Eixo y','FontSize',fontsize)
380 zlabel('Eixo z','FontSize',fontsize)
381 lgd = legend('Trajet ria Refer ncia','Trajet ria Realizada pelo Drone
    ')
382 lgd.FontSize = fontsize2;
383 grid on
384 axis([-1.5 1.5 -1.5 1.5 0 2.2]) %Ajuste do Gr fico
```

```
385 % ================================================================
386 %% TEMPOR RIO
387 %return
388 %close all
389
390
391 %% Plot dos Sinais de Saida e Sinais de Controle Individuais
392 %% X
393 figure
394 plot(tout,x,'b','LineWidth',2)
395 hold on
396 plot(t,xref,'r','LineWidth',2)
397 lgd = legend('X da simula  o','X de refer ncia')
398 lgd.FontSize = fontsize2;
399 title('Sa da X','FontSize',fontsize)
400 grid on
401 axis([0 20 -3 3])
402 xlabel('Tempo (s)','FontSize',fontsize)
403 ylabel('Posi  o x (m)','FontSize',fontsize)
404
405 figure
406 plot(tout,u_theta,'LineWidth',2)
407 title('Sinal de Controle - U_{\theta}','FontSize',fontsize)
408 grid on
409 %% Y
410 figure
411 plot(tout,y,'b','LineWidth',2)
412 hold on
413 plot(t,yref,'r','LineWidth',2)
414 lgd = legend('Y da simula  o','Y de refer ncia','FontSize',fontsize)
415 lgd.FontSize = fontsize2;
416 title('Sa da Y','FontSize',fontsize)
417 grid on
418 axis([0 20 -1.5 1.5])
419 xlabel('Tempo (s)','FontSize',fontsize)
420 ylabel('Posi  o y (m)','FontSize',fontsize)
421
422 figure
423 plot(tout,u_fi,'LineWidth',2)
424 title('Sinal de Controle - U_{\phi}','FontSize',fontsize)
425 grid on
426 %% Z
427 figure
428 plot(tout,z,'b','LineWidth',2)
429 hold on
430 plot(t,zref,'r','LineWidth',2)
431 lgd = legend('Z da simula  o','Z de refer ncia','FontSize',fontsize)
```

```
lgd.FontSize = fontsize2;
title('Sa da Z','FontSize',fontsize)
grid on
xlabel('Tempo (s)','FontSize',fontsize)
ylabel('Posi  o z (m)','FontSize',fontsize)
axis([0 10 0 1.5]) %circulo
axis([0 30 0 1.8]) %oito

figure
plot(tout,uz,'LineWidth',2)
title('Sinal de Controle - Uz','FontSize',fontsize)
grid on

%% PSI
figure
plot(tout,e_psi,'b','LineWidth',2)
hold on
plot(t,psiref,'r','LineWidth',2)
lgd = legend('\psi da simula  o','\psi de refer ncia')
lgd.FontSize = fontsize2;
title('Sa da \psi','FontSize',fontsize)
grid on
xlabel('Tempo (s)','FontSize',fontsize)
ylabel(' ngulo  \psi (rad)','FontSize',fontsize)
axis([0 30 -0.6 0.6])

figure
plot(tout,u_psi,'LineWidth',2)
title('Sinal de Controle - U_{\psi}','FontSize',fontsize)
grid on
```

Listing C.4 – Projeto do Controlador LQR Robusto Completo