Project 3 – Motion Control and Planning

Course: Unmanned Aerial Vehicles (UAVs)

M.Sc. in Aerospace Engineering

Professor: Bruno Guerreiro

June 2024

Group 3

José Duarte Dias Corvo, student number 67118

Rodrigo Miguel Santos Silva Pardela Veríssimo, student number 67133

Vasco Rafael da Ponte Luís Nunes, student number 67304

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Abstract

"This project aims at designing basic control loops and planning algorithms for the Crazyflie drone. In a first part, linear control loops will be explored, while in a second part, a nonlinear control loop for the full 3-D motion of the vehicle will be considered. The last part we will aim at implementing a simple planning strategy for avoiding a previously unknown no-fly-zone. In this lab project, the knowledge acquired in the first two projects will be important, both for having a good model of the drone, as well as to understand how the drone is localizing itself in the arena." – Bruno Guerreiro, 16th May 2024

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Index

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

List of Figures

1. Introduction

As cited by Captain Brian Tice from United States Air Force [1], Unmanned Aerial Vehicles (or UAVs) are powered vehicles that do not carry human operators. These vehicles started to be developed on the twentieth century for military use, but have evolved into other uses such as agriculture, entertainment, or product deliveries.

Nowadays it's still a technology under intense development, where the safety and security regarding the usage of these vehicles in our society is a big concern. Operating an unmanned aerial vehicle is common when a human is in charge of the commands, the use of fully autonomous drones in complex and challenging environments is still underdeveloped and under intense research by universities and the relevant industries that could benefit from this.

In this third and last project, the group was designated to consider the design of basic linear and nonlinear control loops, as well as the simple planning algorithms necessary to drive a micro aerial vehicle called Crazyflie 2.1.

Figure 1 – Crazyflie 2.1 by Bitcraze Store

A Micro Aerial Vehicle (or MAV) is a man-portable aerial vehicle which, by EASA's (European Union Aviation Safety Agency) definition, weighs less than 250 grams [2].

The coding resolution of this project is in the format of a GitHub repository after working on the software MATLAB R2021a.

2. Project 3 Parameters, Subjects and Goals

In the first part of this project, the group will design a simple linear controller for the position control of the Crazyflie 2.1 drone in 3D, considering null yaw angles.

Considering an ENU (East-North-Up) local tangent plane centred at the drone area of the CybAer laboratory in FCT-UNL (Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa), with the respective coordinates (Latitude 38.660319ºN, Longitude -9.204972ºW) as the inertial frame. It will be assumed that the drone is a rigid body in 3-D space with the relevant external forces acting on the centre of mass of the drone. In order to simplify, it will also be assumed that the body frame is located at the centre of mass of the drone, considering a constant mass ("$\boldsymbol{m} = 0.032\ kg$", as mentioned by Professor Bruno Guerreiro during a class on the 21st of march 2024 of "Unmanned Autonomous Vehicles") and also considering a constant matrix of the moment of inertia ("$\boldsymbol{J} = diag(J_x, J_y, J_z) = \begin{bmatrix} 1.395 \times 10^{-5} & 0 & 0 \\ 0 & 1.436 \times 10^{-5} & 0 \\ 0 & 0 & 2.173 \times 10^{-5} \end{bmatrix} {kg}/{m^2}$").

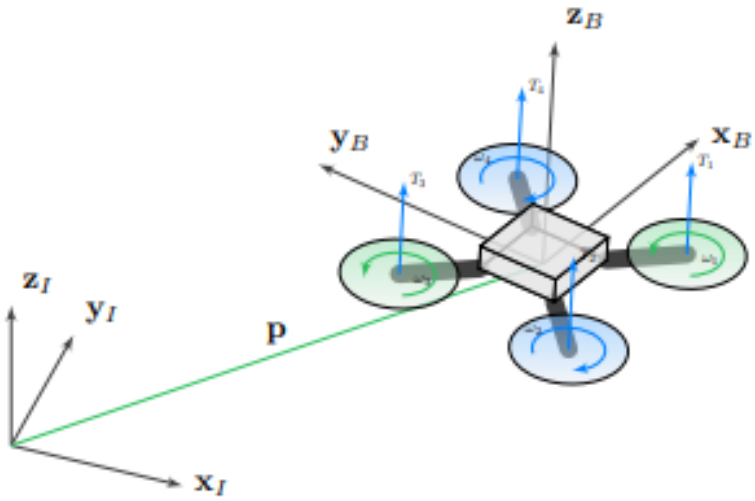

Figure 2 – Crazyflie 2.1 reference frames and configuration

In order to develop the project, the group will consider the following state-space variables of the drone:

- The drone's 3-D position (from the centre of mass "B" and origin of the body frame) relative to the inertial frame "I" ("$\boldsymbol{p} = {}^{I}p_B \in \mathbb{R}^3$");
- The velocity of the drone described on the body frame ("$\dot{\boldsymbol{p}} \in \mathbb{R}^3$");
- The attitude of the body frame regarding the inertial frame ("$\boldsymbol{R} \in SO(3)$"), parametrized by the $Z-Y-X$ Euler angles vector ("$\boldsymbol{\lambda} = [\phi \quad \theta \quad \psi]^T$");
- The relative angular velocity acting on the body frame ("$\boldsymbol{\omega} \in \mathbb{R}^3$").

Regarding the forces acting on the drone, it was considered the Earth Gravity Acceleration ("$\boldsymbol{g_{earth}} = 9.82\ [{m}/{s}]$") and Atmospheric Density ("$\boldsymbol{\rho_{earth}} = 1.217\ \left[{kg}/{m^3}\right]$") [3], the sum of the four forces of propulsion generated by the rotors ("$\boldsymbol{F_p} = F_{p_1} + F_{p_2} + F_{p_3} + F_{p_4}$")

and the sum of the four moment vectors of the propulsion generated by the rotors thrust ("$\boldsymbol{n_p} = n_{p_1} + n_{p_2} + n_{p_3} + n_{p_4}$"). The maximum thrust needed by the sum of the forces of propulsion is obtained by the formula "$T_{max} = (m_{drone} + m_{maximum\ load}) \times g_{earth} \Leftrightarrow$ $\Leftrightarrow T_{max} = (0.032\ kg + 0.015\ kg) \times 9.82\ ^{m}/_{S} = 0.46154\ N$".

The group, in order to obtain the most accurate simulation of the drone, verified the external dimensions of the Crazyflie 2.1.

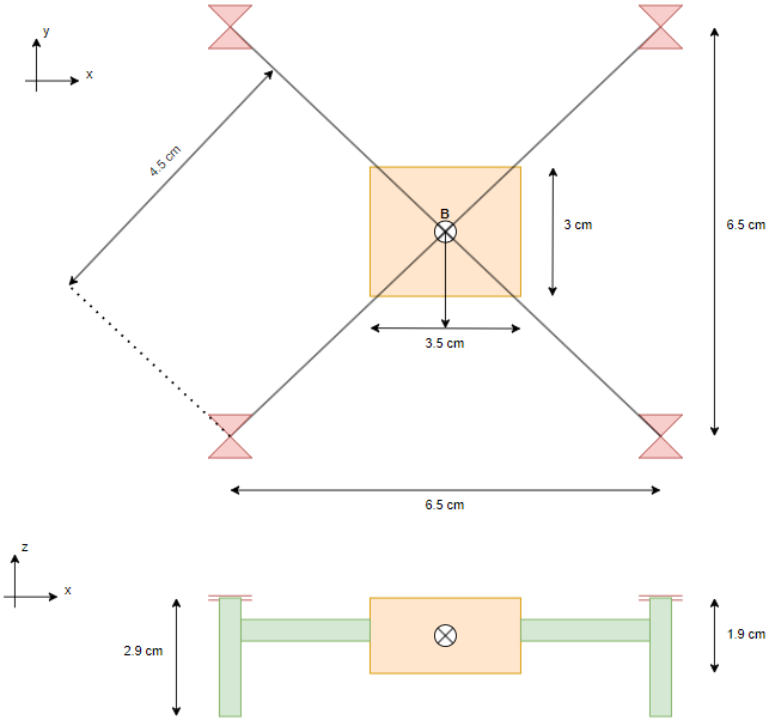

Figure 3 – Sketch displaying the dimensions of the Crazyflie 2.1 drone

Regarding the rotors, it can be assured that they have a vertical propulsion system (90º or $\frac{\pi}{2}\ rad$ regarding the axis angle xz).

Regarding the drag acting on the drone, the drag coefficient is considered to be "$c_d = 1$", which will simplify the calculations and is similar to the drag coefficient of the 2D shape of a rectangle. It's known that the total drag acting on the drone is obtained from the following function: "$Drag_{Body} = Drag_x + Drag_y + Drag_z$".

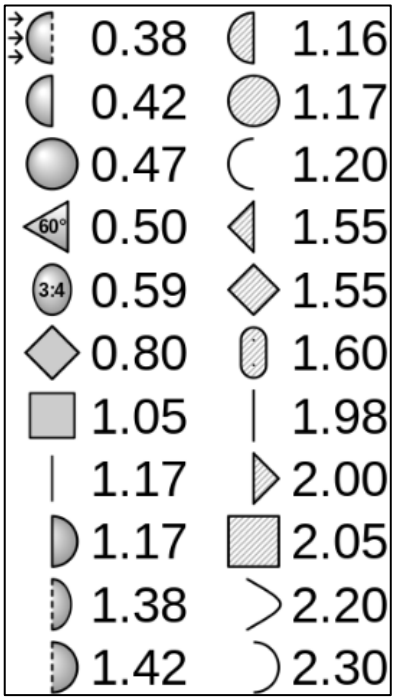

Figure 4 – Drag coefficient of different 2D shapes [4]

The goals imposed for this part of the project are the following:

- **Goal number 1.1:** Based on the results of Project 1, obtain a simplified nonlinear model of the Crazyflie drone that receives as input the vector "$u_\lambda \epsilon \mathbb{R}^4$" (defined as "$u_\lambda = [T \quad \lambda^T]^T$") with the total thrust "T" and the Euler angle vector "$\lambda \epsilon \mathbb{R}^3$" (composed of the roll angle "ϕ", the pitch angle "θ" and yaw angle "ψ").
- **Goal number 1.2:** For small roll and pitch angles (and assuming zero yaw), obtain an equivalent actuation in acceleration ("$u_a \epsilon \mathbb{R}^3$"), combining the effects of the total thrust, gravity and drone attitude.
- **Goal number 1.3:** Define the linear model for the linear motion of the drone, considering the control variable "u_a" and the state vector "$x = [p^T \quad {v_{inertial}}^T]^T$".
- **Goal number 1.4:** Design an LQR controller for this linear model and test it in simulation. Note that you should also test the controller considering the nonlinear model.
- **Goal number 1.5:** Consider now an error state vector defined as "$\tilde{x} = x - \bar{x}$", where we assume that the reference state "$\bar{x}$" is driven by the same dynamics as "x". Obtain the equivalent state space model.
- **Goal number 1.6:** Design an LQR controller for this error model and test it in simulation. Comment on the difference between this linear model and the previous one.

Building on the previous linear control algorithms and using concepts of Lyapunov theory:

- **Goal number 2.1:** Design a nonlinear controller with actuation in body accelerations ("$u_a \in \mathbb{R}^3$") and assuming zero yaw, that is able to achieve asymptotic stability in the Lyapunov sense (prove this result).

- **Goal number 2.2:** Test the controller developed in simulation.

Considering the controllers developed previously, these controllers will be tested in reality with the Crazyflie 2.1 drones. The implementation will be in the form of a Python function that has access to the current state and desired state, defining the desired acceleration vector in 3D for the drone inner loops to follow.

With this in mind:

- **Goal number 2.3:** Follow the professor instructions to use the arena PC and development environment to test the controllers developed in the arena, ensuring adequate saturations for the inputs.
- **Goal number 2.4:** Obtain the data logs for the experiment and compute the RMSE values of the 3D error between the real and desired position, according to the expression "$p_{RMSE} = \sqrt{\frac{1}{N}\sum_{k=0}^{N}\|p(k) - p_d(k)\|^2}$" as well as the velocity RMSE given by "$v_{RMSE} = \sqrt{\frac{1}{N}\sum_{k=0}^{N}\|v(k) - v_d(k)\|^2}$". The performance of the controller in terms of these values (it's preferrable to obtain low values of "p_{RMSE}" and "v_{RMSE}") will be considered.

Following these set of goals, the group will build on the previous controller and design the reference trajectories for the control loops. Defining the flying arena usable limits as "$\mathcal{P}_{fly-zone} = \begin{Bmatrix} -1.2 \le p_x \le 1.2 \\ p\epsilon\mathbb{R}^3 -2.1 \le p_y \le 2.1 \\ 0.0 \le p_z \le 2.0 \end{Bmatrix}$" and the no-fly-zones or obstacles as "$\mathcal{P}_{no-fly-zone_1} = \begin{Bmatrix} -0.6 \le p_x \le 0.6 \\ p\epsilon\mathbb{R}^3 -0.7 \le p_y \le 0.8 \\ 0.0 \le p_z \le 1.2 \end{Bmatrix}$" and "$\mathcal{P}_{no-fly-zone_2} = \begin{Bmatrix} -0.6 \le p_x \le 0.6 \\ p\epsilon\mathbb{R}^3 -0.8 \le p_y \le 0.7 \\ 0.6 \le p_z \le 2.0 \end{Bmatrix}$", and considering the possible initial positions defined by "$\mathcal{P}_{start} = \begin{Bmatrix} -0.3 \le p_x \le 0.3 \\ p\epsilon\mathbb{R}^3 -0.3 \le p_y \le 0.3 \\ p_z = 0.0 \end{Bmatrix}$" and two different sets of possible goal positions ("$\mathcal{P}_{goal_1} = \begin{Bmatrix} -0.2 \le p_x \le 0.2 \\ p\epsilon\mathbb{R}^3 \ 0.9 \le p_y \le 1.3 \\ 0.4 \le p_z \le 0.8 \end{Bmatrix}$" and "$\mathcal{P}_{goal_2} = \begin{Bmatrix} -0.2 \le p_x \le 0.2 \\ p\epsilon\mathbb{R}^3 -1.3 \le p_y \le -0.9 \\ 1.0 \le p_z \le 1.4 \end{Bmatrix}$"), the following set of goals were presented:

- **Goal number 3.1:** Implement a planning algorithm of your choice that is able to drive the drone from any initial position ("$p_{initial} \in \mathcal{P}_{start}$") to a first goal position ("$p_1 \in \mathcal{P}_{goal_1}$"), to a second goal position ("$p_2 \in \mathcal{P}_{goal_2}$") and back to the original starting point, ensuring the drone is always within "$\mathcal{P}_{fly-zone}$" while avoiding "$\mathcal{P}_{no-fly-zone_1}$" and "$\mathcal{P}_{no-fly-zone_2}$".
- **Goal number 3.2:** Test this planning algorithm with the controller developed in simulation, noting that "$p_{initial}$", "p_1" and "p_2" are randomly chosen from their respective sets, and describe the experimental tasks.

- **Goal number 3.3:** Test the planning algorithm and controller in the flying arena.

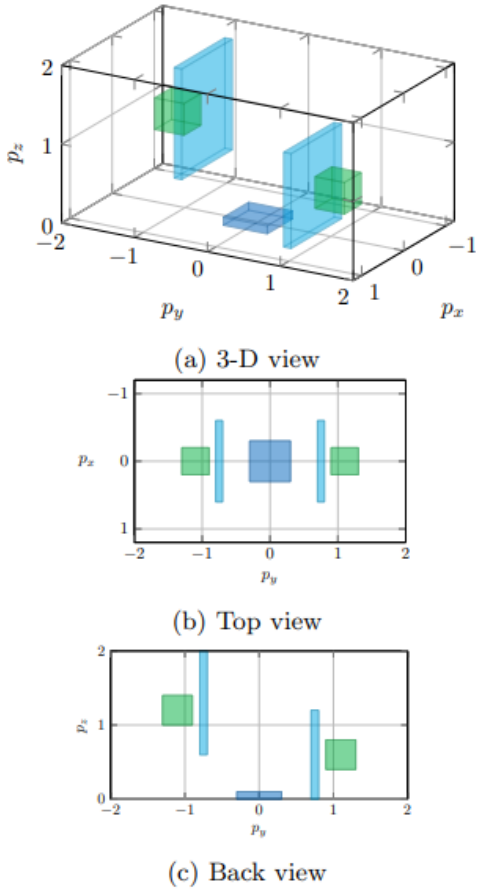

Figure 5 – Flight arena planning setup

3. Development of the goals imposed

In order to obtain the goals proposed by coding in MATLAB R2021a, the group based its research on the examples provided by Professor Bruno Guerreiro on the 2023/2024 version of FCT/UNL Moodle.

3.1. Goal number 1.1

Based on the results of Project 1, obtain a simplified nonlinear model of the Crazyflie drone that receives an input the vector "$u_\lambda \epsilon \mathbb{R}^4$" (defined as "$u_\lambda = [T \quad \lambda^T]^T$") with the total thrust "T" and the Euler angle vector "$\lambda \in \mathbb{R}^3$" (composed of the roll angle "ϕ", the pitch angle "θ" and yaw angle "ψ".

In order to obtain the nonlinear model pretended, the dynamics and parameters obtained in the first project of this course were used and adjusted to this project, with the following equations:

- $\dot{p} = v_{inertial}$;
- $\dot{v}_{inertial} = -g * z_I + \frac{1}{m} * (f_a + R(u_\lambda) * f_p)$.

As announced previously, "$u_\lambda = [T \quad \lambda^T]^T$" (with "$u_\lambda \in \mathbb{R}^4$"). The input on the drone is defined by the total thrust applied and the Euler angles (consisting of the yaw angle "ψ", the roll angle "ϕ" and the pitch angle "θ").

In order to develop this model, different parameters were defined, as demonstrated in the following graphics.

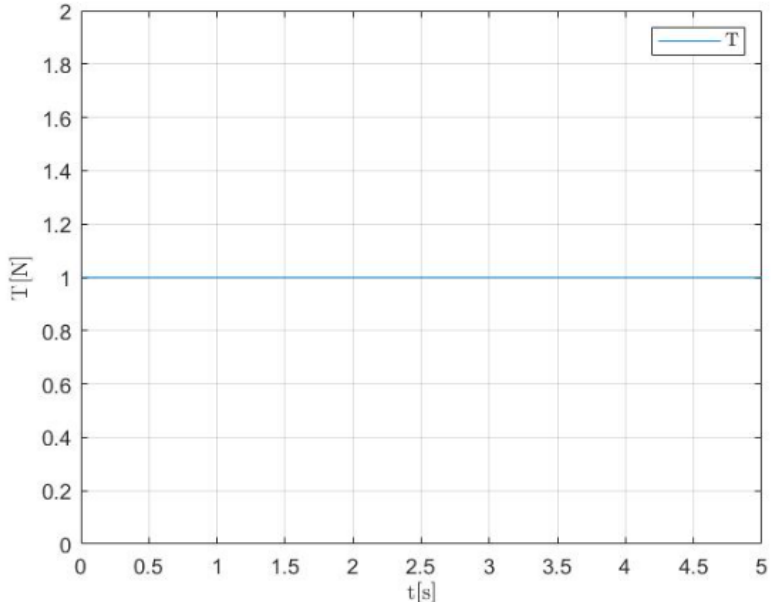

Figure 6 – Total thrust applied ("T $[N]$") over time ("t $[s]$") in the nonlinear model

It's possible to verify that the total thrust applied is constant over time with a given value of "$T = 1N$".

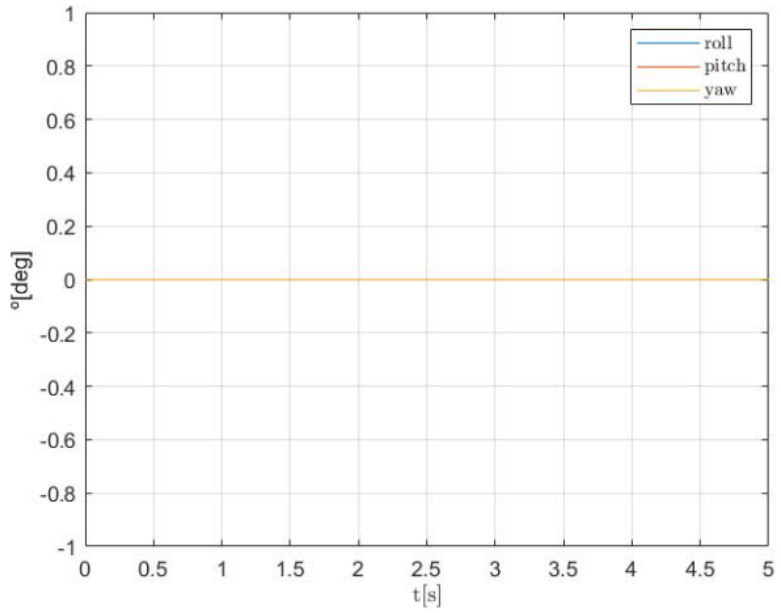

Figure 7 – Variation of the different Euler angles ("º $[deg]$") over Time ("t $[s]$") in the nonlinear model

By analysing the graphic for the different Euler angles (yaw angle "ψ", roll angle "ϕ" and pitch angle "θ"), it's observable that these angles remain consistently null over time (this happens intentionally).

With the thrust and Euler angles defined, it's possible to obtain graphics for the evolution of position and velocity over time.

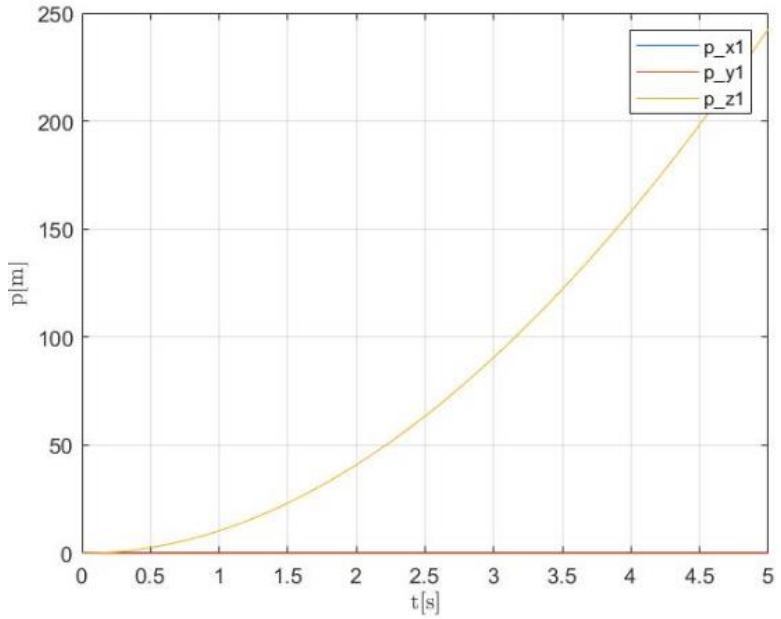

Figure 8 – Variation of the Position ("p $[m]$") over Time ("t $[s]$") in the different axes of the nonlinear model

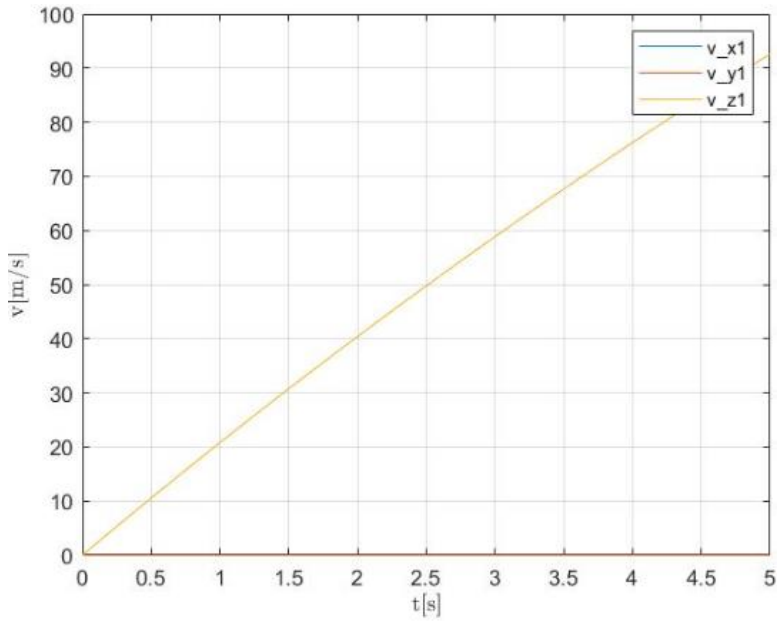

Figure 9 – Variation of the Velocity ("v $[m/s]$") over Time ("t $[s]$") in the different axes of the nonlinear model

While the values for the axes x and y are null for both the variations of position and velocity, the values for the axis z increase from null for both parameters, demonstrating an increase in altitude resulting from the thrust applied.

3.2. Goal number 1.2

For small roll and pitch angles (and assuming zero yaw), obtain an equivalent actuation in acceleration ("$u_a \in \mathbb{R}^3$"), combining the effects of the total thrust, gravity and drone attitude.

The parameter "u_a" is the inertial acceleration that it's possible to control through the control of the Euler angles (which can be controlled by the angular velocity).In order to obtain the acceleration vector ("$u_a \in \mathbb{R}^3$"), the equation "$\dot{v}_{inertial} = -g * z_I + \frac{1}{m} * (f_a + R(u_\lambda) * f_p)$" is used, obtaining the following formula:

- $u_a = \dot{v}_{inertial} - g * z_I + \frac{1}{m} * f_a$.

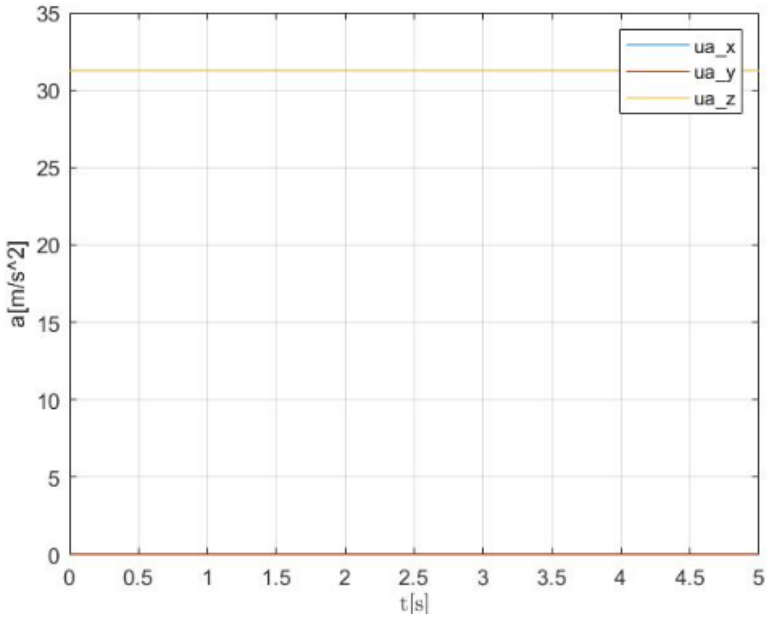

Figure 10 – Variation of Acceleration ("$a\ [m/s^2]$") over Time ("$t\ [s]$") in the different axes of the nonlinear model

As it could be predicted, the acceleration is constant and null in the axes x and y, while the acceleration is also constant but different from zero in the axis z, justifying the vertical movement.

3.3. Goal number 1.3

Define the linear model for the linear motion of the drone, considering the control variable "u_a" and the state vector "$x = [p^T \quad v_{inertial}{}^T]^T$".

In order to obtain the linear model for the linear motion of the drone, it's necessary to obtain the state matrices "A" and "B". These equations are defined by the equations "p" and "$v_{inertial}$" in "$x = [p^T \quad v_{inertial}{}^T]^T$".

The matrices "A" and "B" are calculated with the derivation of the state equations by the outputs:

- $A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & Drag_x & 0 & 0 \\ 0 & 0 & 0 & 0 & Drag_y & 0 \\ 0 & 0 & 0 & 0 & 0 & Drag_z \end{bmatrix}$;
- $B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

With these matrices, it's possible to compare the nonlinear model with the linear model regarding position and velocity.

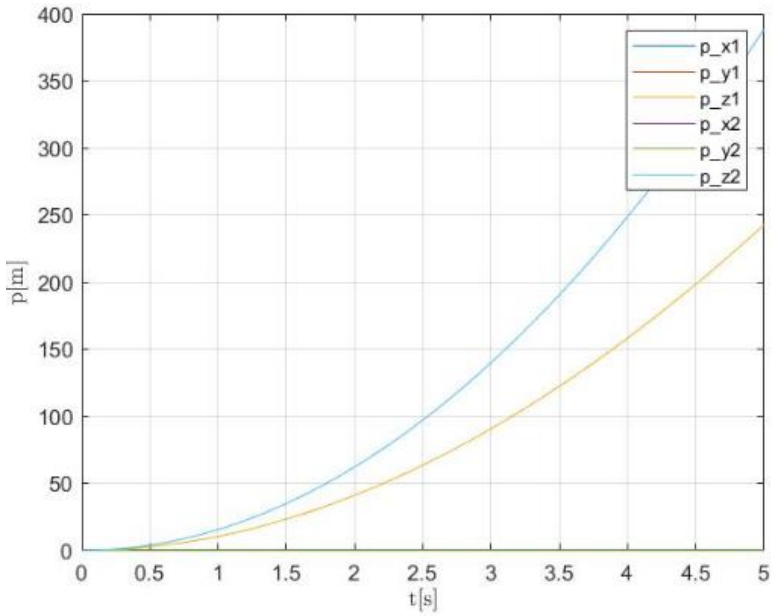

Figure 11 – Comparison of the variation of Position ("$p\ [m]$") over Time ("$t\ [s]$") between the Linear Model ("p_n1") and the nonlinear model ("p_n2")

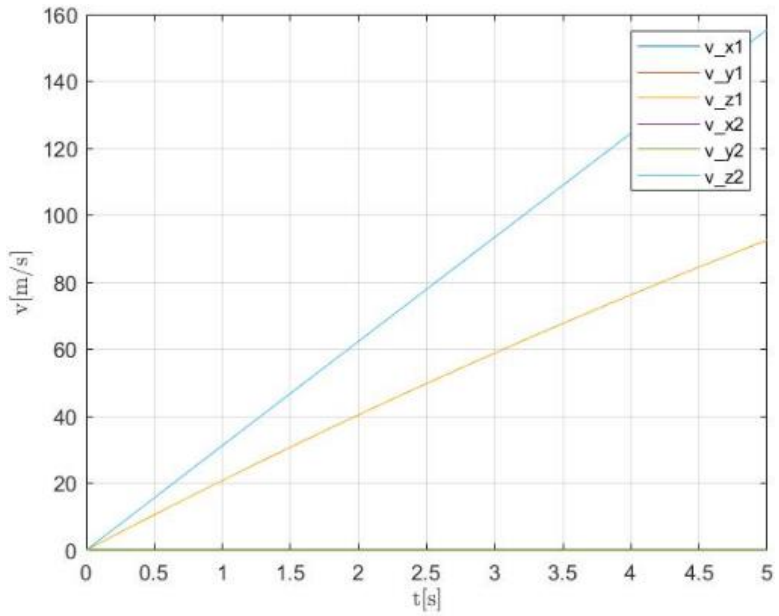

Figure 12 – Comparison of the variation of Velocity ("$v\ [m/s]$") over Time ("$t\ [s]$") between the Linear Model ("p_n1") and the nonlinear model ("p_n2")

Similarly to the nonlinear model, there's no variation in the values for the axes x and y for both values of position and velocity, remaining null.

Since one of the states considered is the inertial velocity, when linearizing the nonlinear model and given the independence of the gravitational acceleration from the states and inputs, it is not taken into account in the linear model (in the matrices "A" and "B" respectively), which justifies the greater acceleration in the axis z in the upward movement of the drone when compared to the nonlinear model (which takes this nonlinearity into account (in the simulation takes the gravitational acceleration decelerating the movement into account and there are more unknown nonlinearities to take into account in a real case).

3.4. Goal number 1.4

Design an LQR controller for this linear model and test it in simulation. Note that you should also test the controller considering the nonlinear model.

In order to obtain an LQR controller (Linear-quadratic regulator) for the linear model defined earlier, the matrices "Q" and "R" need to be defined (in order to define the weights given to the errors accepted). These matrices will have the following configurations:

- $Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.00000001 \end{bmatrix}$;
- $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$.

It's possible to observe in "Q" that the weight given for positions is higher than the weight given for velocities (positions have a weight diagonal of $[10, 10, 10]$ while velocities have a weight diagonal of $[0.0001, 0.0001, 0.00000001]$). In order to obtain movement in the pretended direction, it's necessary that the weight provided to the positions is higher to the weight for the velocities, due to the fact that the controller must focus on the positions in order to avoid the controller's defocus and maintain an almost null, staying virtually static.

In the diagonal regarding the velocities of the matrix "Q", the weights are bigger in the axes x and y due to the fact that this experiment was made considering a hypothetical context of transportation, where the horizontal surface would eventually support the weight transported.

Regarding the matrix "R", the first and second weight values refer to the inputs in the axes x and y, while the third weight value refers to the inputs in z. This means that the acceleration in x and y is higher that the acceleration in z, justified by the same logic of the matrix "Q".

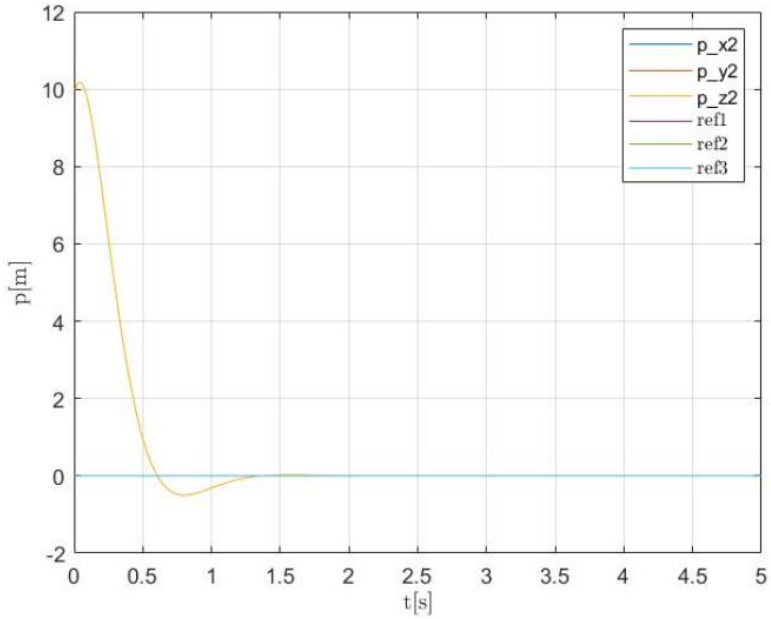

Figure 13 – Variation of Position ("$p\ [m]$") over Time ("$t\ [s]$") in the different axes of the linear "LQR" controller

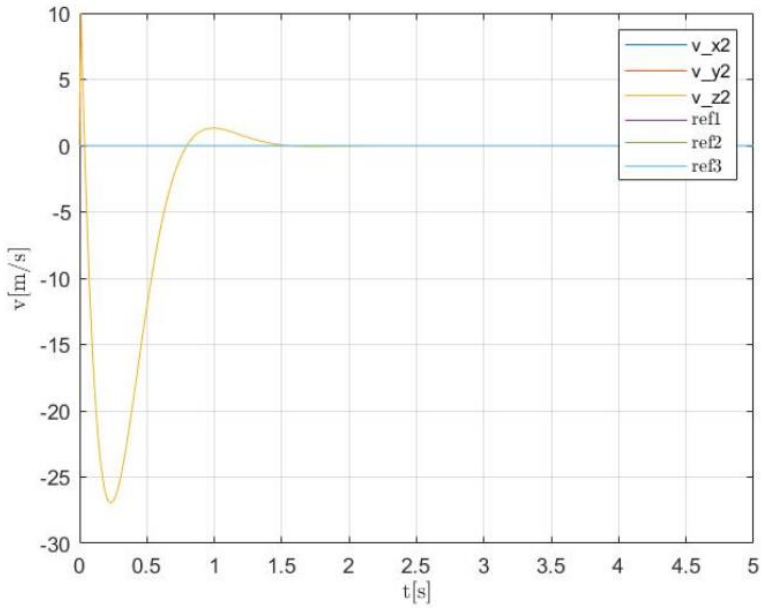

Figure 14 – Variation of Velocity ("$v\ [m/s]$") over Time ("$t\ [s]$") on the different axes of the linear "LQR" controller

As it can be observed in both graphics of the linear LQR controller, the drone is able to follow the reference as expected. In both graphics the states will tend to zero, since this controller has the specificity to tend all its inputs to zero.

Figure 15 – Inputs and Outputs of the "LQR" controller

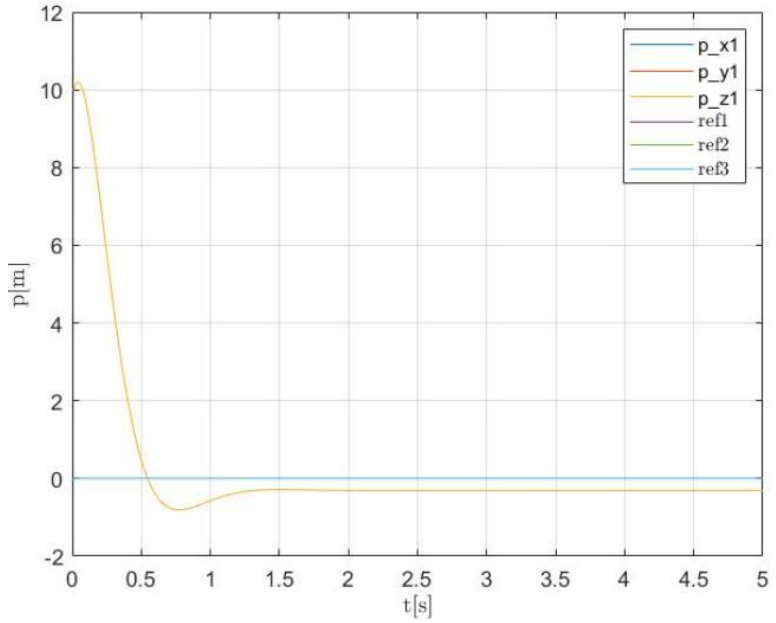

Figure 16 - Variation of Position ("$p\ [m]$") over Time ("$t\ [s]$") on the different axes of the nonlinear "LQR" controller

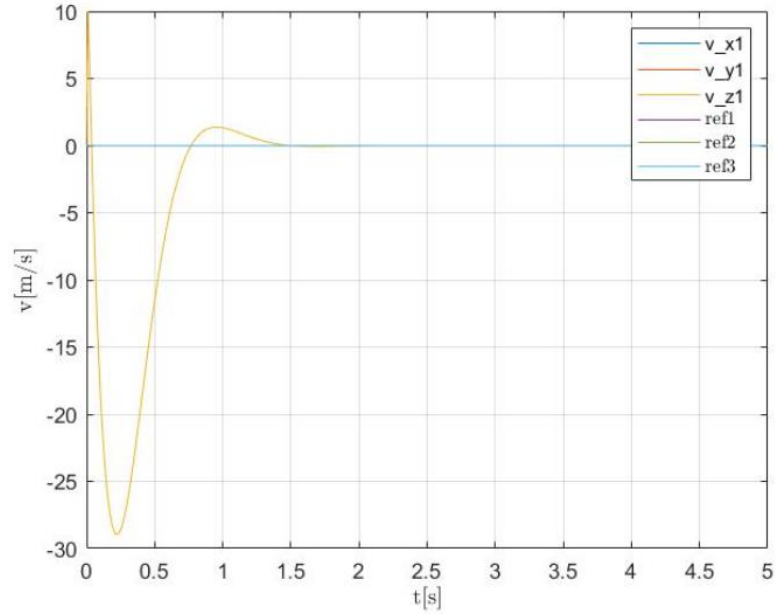

Figure 17 – Variation of Velocity ("$v\ [m/s]$") over Time ("$t\ [s]$") in the different axes of the nonlinear "LQR" controller

Applying the LQR controller in the nonlinear model, it's possible to observe that the results are not the same as the ones in the linear model. While the LQR controller obtained in the linear model used to tend do zero due to the fact that the gravitational acceleration was not considered during the linearization and obtaining the "LQR" controller, this does not happen in the nonlinear model (which is considered the gravitational acceleration). The reference (both in the position and in the velocity) is at zero, and the initial position in the axis z is at approximately $10\ [m]$, which will dictate a descending movement from the start

provoked by the controller. Applying the linear controller (that does not consider the gravitational acceleration) into the nonlinear model will provoke a descend with a higher rate in the axis z (after considering the gravitational acceleration).

In addition, the position in the axis z does not tend towards zero as it would be desired since the position and error to which the drone has converged has a reaction from the drone and its controller with an input of inertial acceleration equivalent to the gravitational acceleration, which the controller is not aware of (given that error and the linear model, the controller will act in a way that the input in question is enough to reach the reference).

Since there's this problem in the nonlinear model, there will also be nonlinearities and unknown elements in the real drone, meaning that this problem becomes recurrent and there's going to appear a need for solutions to minimize errors.

3.5. Goal number 1.5

Consider now an error state vector defined as "$\tilde{x} = x - \bar{x}$", where we assume that the reference state "$\bar{x}$" is driven by the same dynamics as "x". Obtain the equivalent state space model.

In order to obtain the state error vector, it's necessary the development of the error dynamics "e_p" and "e_v", obtaining the derivative function "$x = [e_p \quad e_v]^T$", where "$e_p = p - p_d$" and "$e_v = v - v_d$".

By deriving the states, the following functions are obtained:

- $\dot{e}_p = \dot{p} - \dot{p}_d = v - v_d = e_v$;
- $\dot{e}_v = \dot{v} - \dot{v}_d = -g * z_I + \frac{1}{m} * f_a + u_a - \dot{v}_d$.

Defining "$u_a = g * z_I - \frac{1}{m} * f_a + \dot{v}_d + u$" as the control law, the function "$\dot{e}_v = \dot{v} - \dot{v}_d = -g * z_I + \frac{1}{m} * f_a + \left(g * z_I - \frac{1}{m} * f_a + \dot{v}_d + u\right) - \dot{v}_d = u$" is established. Therefore, the functions developed are the following:

- $\dot{e}_p = e_v$;
- $\dot{e}_v = u$.

With the new state developed, the new matrices "A" and "B" for the error state are the following:

- $A_e = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$,
- $B_e = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

3.6. Goal number 1.6

Design an LQR controller for this error model and test it in simulation. Comment on the difference between this linear model and the previous one.

Starting from the errors model defined before, the new "LQR" is defined with the matrices "Q" and "R" similar to the ones from before, receiving errors as inputs in order to obtain variation regarding the input desired.

Figure 18 – Inputs and Outputs of the "$LQR(k)$" controller

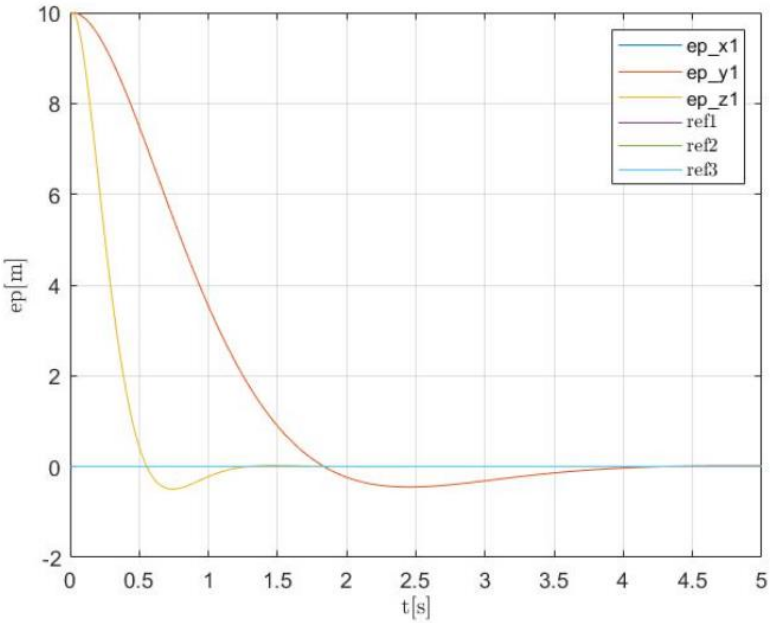

Figure 19 – Variation of the Position Error ("$e_p\ [m]$") over Time ("$t\ [s]$") for the different axes of the nonlinear "$LQR(k)$" model

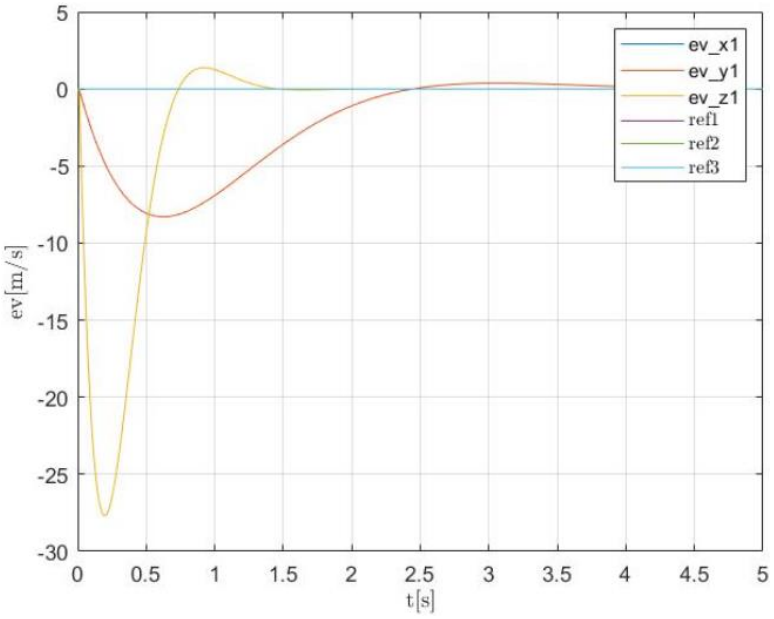

Figure 20 – Variation of the Velocity Error ("$e_v\ [m]$") over Time ("$t\ [s]$") for the different axes of the nonlinear "$LQR(k)$" model

Knowing that the graphics for the nonlinear LQR model regarding "e_p" and "e_v" are similar to the respective graphics for the linear LQR model, it's possible to observe that the errors tend to zero over time and that they are different from zero at the beginning of the controller, allowing to the define references different to null for the initial states considered (position and inertial velocity).

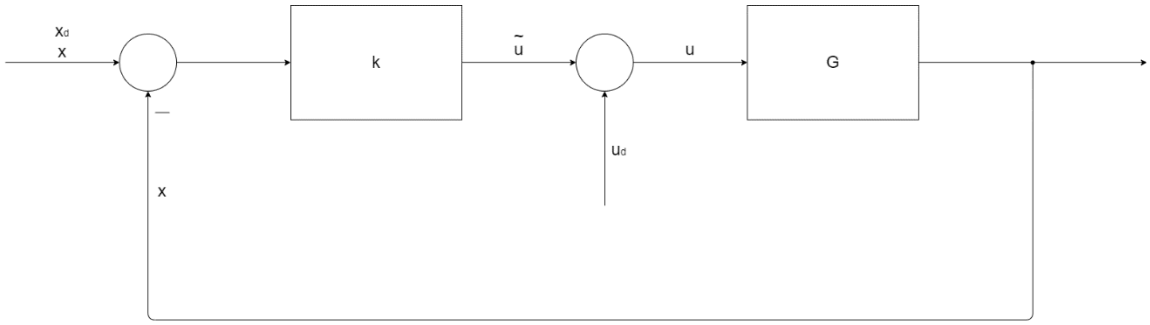

Figure 21 – "$LQR(k)$" controller

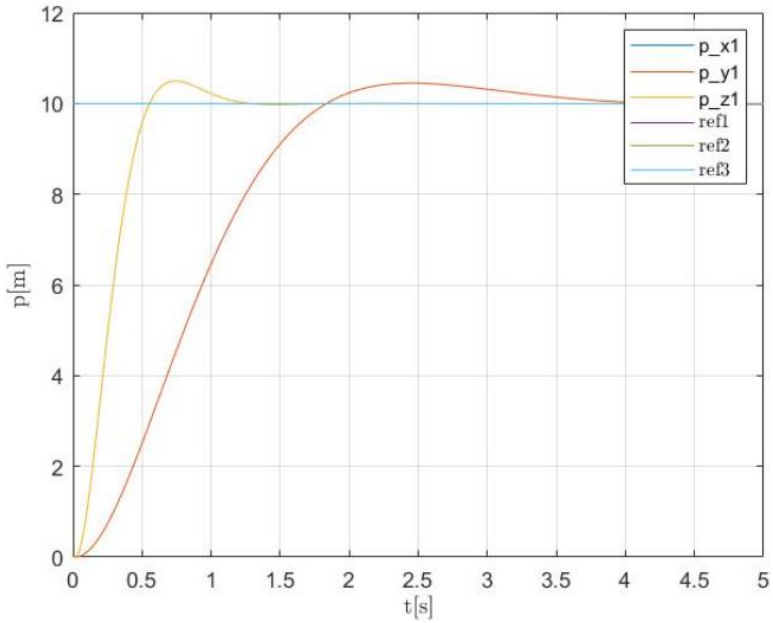

Figure 22 – Variation of Position ("$p\ [m]$") over Time ("$t\ [s]$") for the different axes of the nonlinear "$LQR(k)$" model

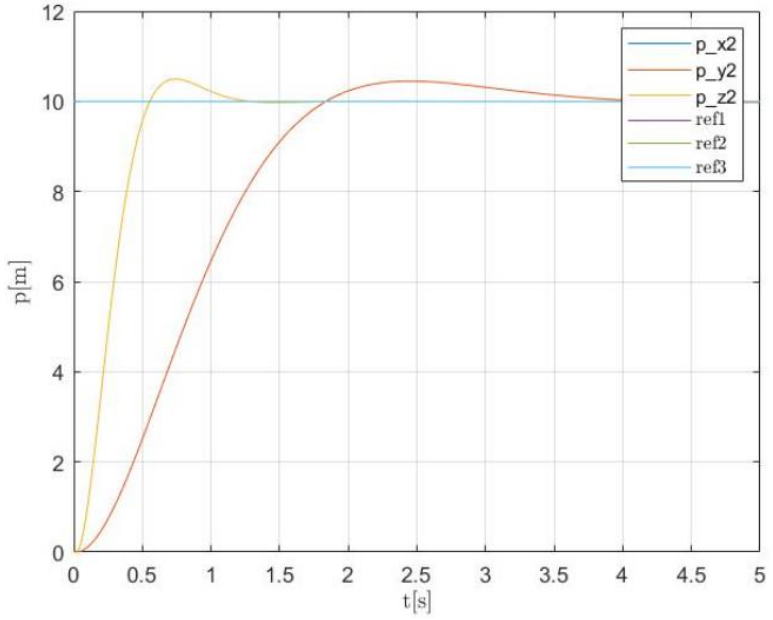

Figure 23 – Variation of Position ("$p\ [m]$") over Time ("$t\ [s]$") for the different axes of the linear "$LQR(k)$" model

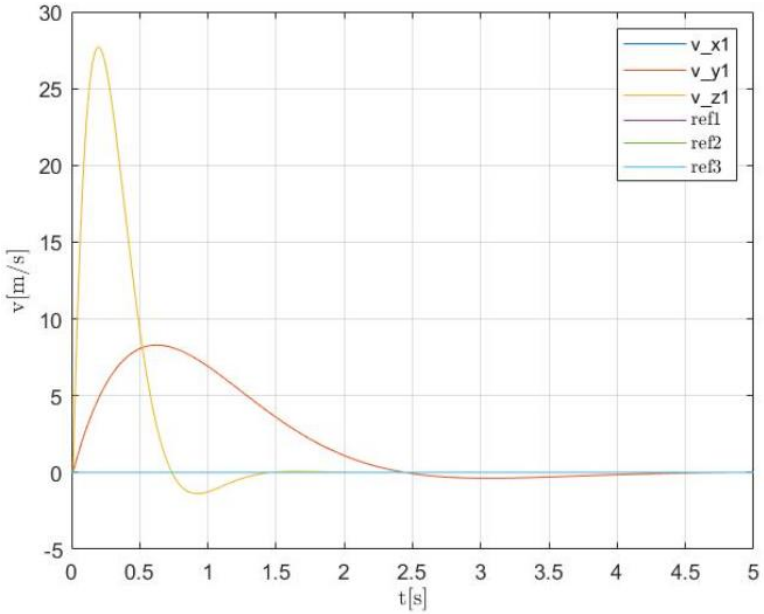

Figure 24 – Variation of Velocity ("$v\ [m/s]$") over Time ("$t\ [s]$") for the different axes of the nonlinear "$LQR(k)$" model

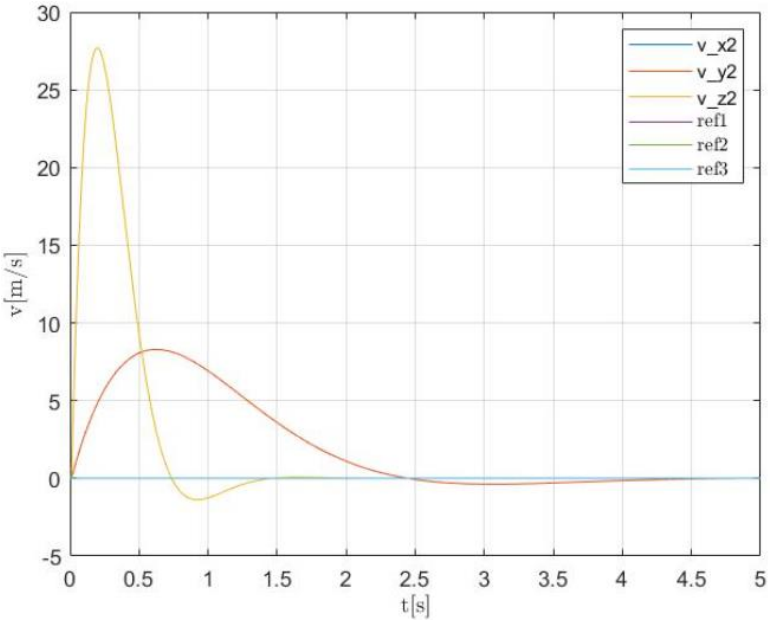

Figure 25 – Variation of Velocity ("$v\ [m/s]$") over Time ("$t\ [s]$") for the different axes of the linear "$LQR(k)$" model

When comparing the graphics of the variations of position and the variations of velocity for both nonlinear and linear "$LQR(k)$" models, it's possible to observe small to no discrepancies. Initially, there were bigger differences due to the fact that the models and controllers were not ready to deal with the nonlinearities caused by the gravitational acceleration, but these differences were nullified with the addition of a desired "u_d", since the inputs on the model begin to be controlled by the outputs of the controller "$\tilde{u}$" and the desired manual input (which will nullify the actions of the gravitational acceleration).

3.7. Goal number 2.1

Design a nonlinear controller with actuation in body accelerations ("$u_a \in \mathbb{R}^3$") and assuming zero yaw, that is able to achieve asymptotic stability in the Lyapunov sense (prove this result).

In order to achieve asymptotic stability in the Lyapunov sense, it's necessary the use of the state error vector obtained previously:

- $\dot{e}_p = e_v$;
- $\dot{e}_v = u$.

Using this vector, the stability is achieved through the Lyapunov theorem if:

- "$V(0) = 0$", "$V(x) > 0, \forall x \in D_{/\{0\}}$" and "$\dot{V}(x) \leq 0, \forall x \in D$" for a stable controller;
- "$V(0) = 0$", "$V(x) > 0, \forall x \in D_{/\{0\}}$" and "$\dot{V}(x) < 0, \forall x \in D$" for an asymptotic stable controller.

To test the state stability, a control law was applied that allows the adjustment of the proportional gains and derivatives ("$u = -k_p * e_p - k_d * e_v$"), resulting in the following functions:

- $\dot{e}_p = e_v$;
- $\dot{e}_v = -k_p * e_p - k_d * e_v$.

Knowing that the Lyapunov function is given by "$V = \frac{1}{2}{e_p}^T e_p + \frac{1}{2}{e_v}^T e_v = \frac{1}{2}\|e_p\|^2 + \frac{1}{2}\|e_v\|^2$" and the respective derivative given by "$\dot{V} = {e_p}^T \dot{e}_p + {e_v}^T \dot{e}_v = {e_p}^T \dot{e}_p + {e_p}^T u$" (since "$\dot{e}_v = u$"), and knowing that "$u = -e_p - k_d e_v$", the function obtained is "$\dot{V} = -k_d {e_v}^T e_v = -k_v \|e_v\|^2 \leq 0$". Considering the state "$x = \begin{bmatrix} {e_p}^T & {e_v}^T \end{bmatrix}^T$", the set of solutions "$S = \{x \epsilon \mathbb{R}^6 : \dot{V} = 0\}$" implies that "$e_v = 0$", therefore "$\dot{e}_v = -k_p * e_p - k_d * e_v = 0$" and "$e_p = 0$". Due to the fact that "$e_p \in R^3$" and "$e_v \in R^3$", it's possible to confirm that the state is asymptotically stable.

Continuing from the state, a new nonlinear controller with two modes is created, with these modes being "step-movement" and "circular flight", with the gains "k_p" and "k_d" being defined by the controller, with the following values:

- $k_p = 0.39$;
- $k_d = 0.29$.

3.8. Goal number 2.2

Test the controller developed in simulation.

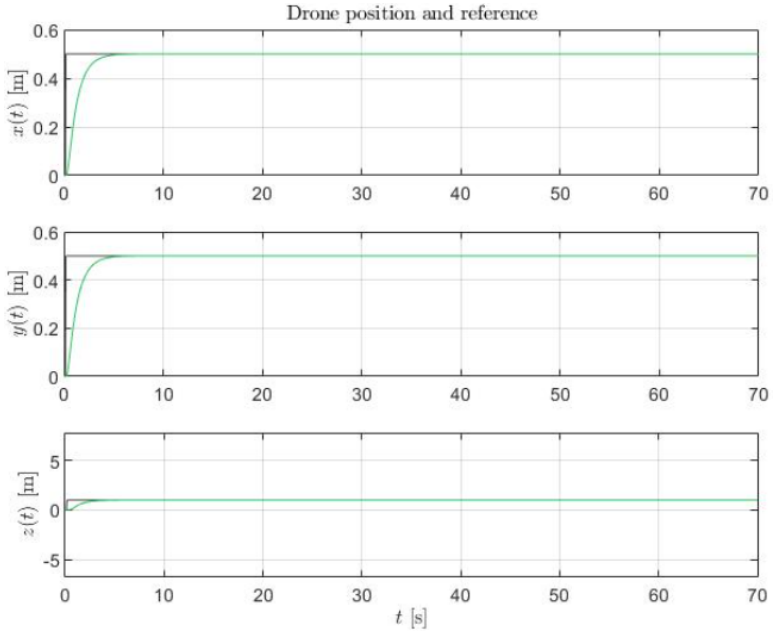

Figure 26 – Variation of the Drone Position over Time ("$t\ [s]$") for the different axes ("$x\ [m]$", "$y\ [m]$" and "$z\ [m]$") of the nonlinear model in the mode "Step-movement"

In the figure above, it's possible to observe that all positions tend to the reference quickly, therefore showing a good response to the controller.

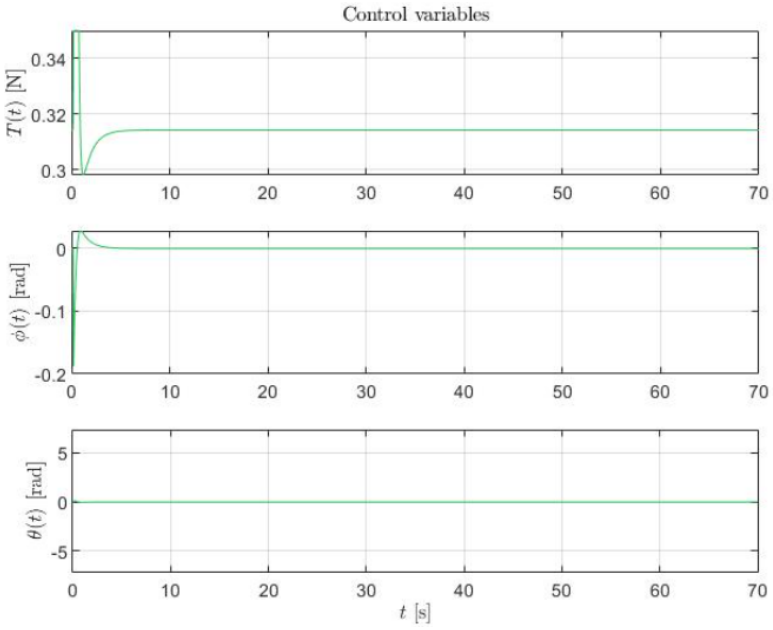

Figure 27 – Variation of the different Control variables (Thrust "$T\ [N]$", Roll Angle "$\phi\ [rad]$" and Pitch Angle "$\theta\ [rad]$") over Time ("$t\ [s]$") of the nonlinear model in the mode "Step-movement"

The outputs obtained from the controller show no variation in the pitch angle "θ" and a small variation in the roll angle "ϕ" (with both angles tending to zero in the end). The small variation in the roll angle is caused by the incapacity of the controller to follow the reference constantly (as showed in the following figure). The thrusts present an initial thrust, descending abruptly until the position of the drone stabilizes and afterwards the thrust mains constant to maintain the position. The graph for the yaw angle "ψ" is not presented, assuming that the value is null consistently.

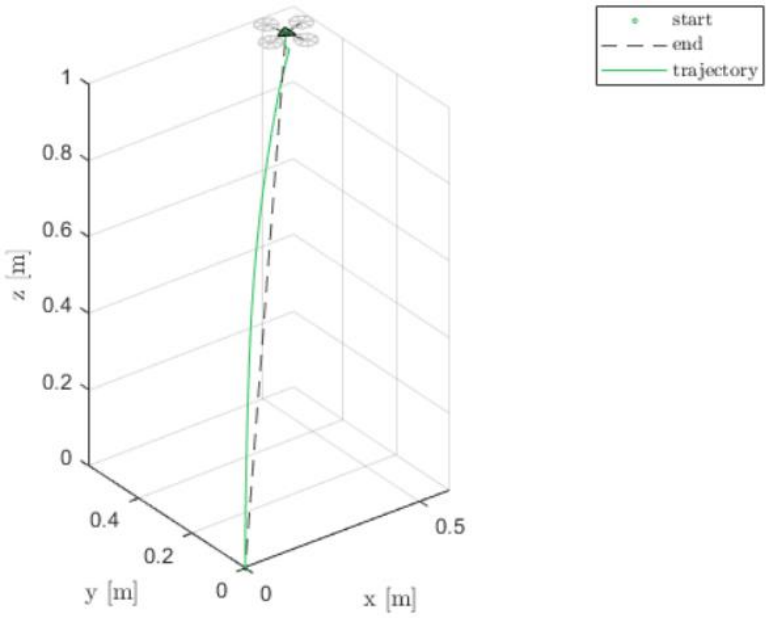

Figure 28 – Trajectory of the drone Crazyflie 2.1 in the mode "Step-movement"

The figure above shows how the controller follows the reference in order to achieve the point desired, achieving a near-perfect trajectory in order to reach the goal (even though the traced line seems to not be followed, since the reference is given as a step with variation almost instantaneously, the controller was developed taking into account over-speed in order to achieve the position wanted, resulting in a RMSE lower to simulations were the graphic seemed to be closer to the ideal traced line).

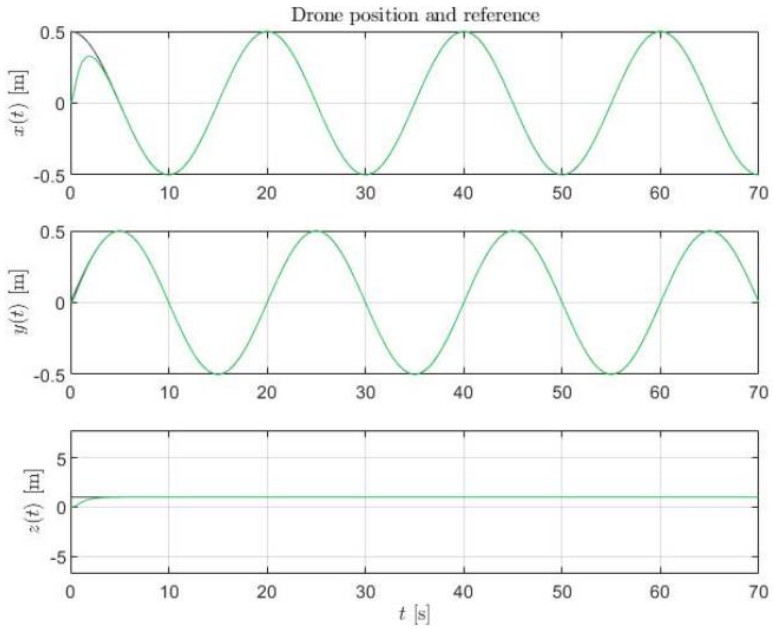

Figure 29 – Variation of the Drone Position over Time ("t $[s]$") for the different axes ("x $[m]$", "y $[m]$" and "z $[m]$") of the nonlinear model in the mode "Circular Flight"

In order to obtain the circular movement of the drone, the graphs show bigger variations of the values of the positions. Knowing that the movement is circular, it's expected that the movements in the axes x and y show a variation similar to comparison between the functions of cosine and sine. Meanwhile, the axis z shows a sudden increase at the beginning, before staying constant over the rest of the time of the simulation, showing that the drone climbs to the desired altitude and maintains the level.

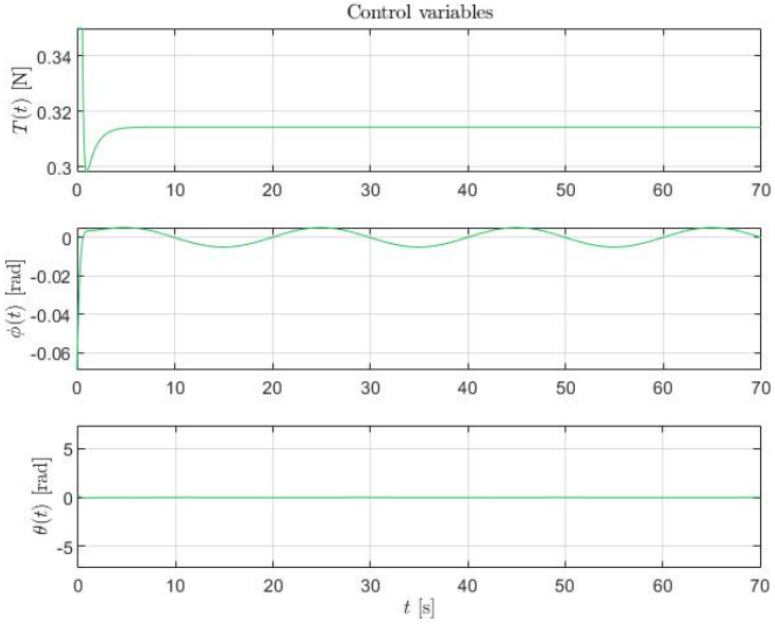

Figure 30 – Variation of the different Control variables (Thrust "T $[N]$", Roll Angle "ϕ $[rad]$" and Pitch Angle "θ $[rad]$") over Time ("t $[s]$") of the nonlinear model in the mode "Circular Flight"

Similar to “Step-movement”, the thrust will begin with a high value, decreasing until it reaches a stable altitude and increasing slightly to maintain the level. Similarly, the pitch angle will remain null throughout the simulation. Nevertheless, the roll angle will present variations justifying the circular movement of the drone.

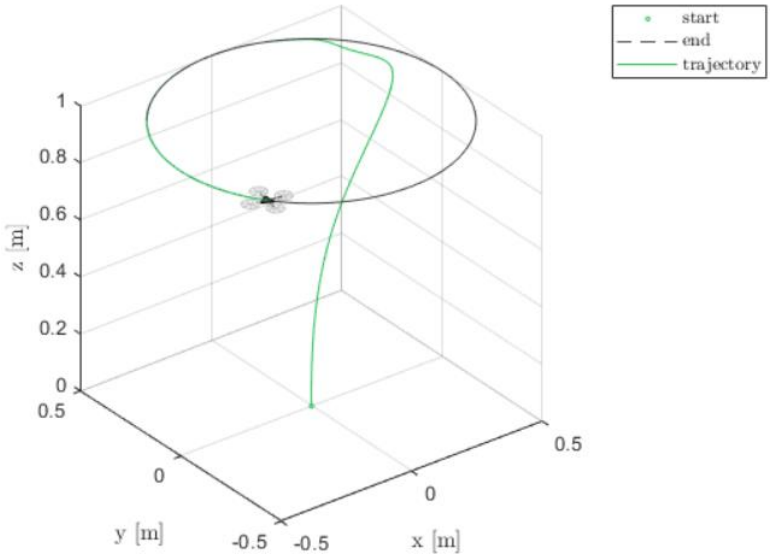

Figure 31 – Trajectory of the drone Crazyflie 2.1 in the mode “Circular Flight”

Knowing that the reference trajectory of the drone is stable in a hovering position, the model is able to follow the trajectory with a higher level of accuracy than the “Step-movement” model.

3.9. Goal number 2.3

Follow the professor instructions to use the arena PC and development environment to test the controllers developed in the arena, ensuring adequate saturations for the inputs.

Unfortunately, the group was unable to realize this task in time.

3.10. Goal number 2.4

Obtain the data logs for the experiment and compute the RMSE values of the 3D error between the real and desired position, according to the expression "$p_{RMSE} = \sqrt{\frac{1}{N}\sum_{k=0}^{N}\|p(k) - p_d(k)\|^2}$" as well as the velocity RMSE given by "$v_{RMSE} = \sqrt{\frac{1}{N}\sum_{k=0}^{N}\|v(k) - v_d(k)\|^2}$. The performance of the controller in terms of these values (it's preferrable to obtain low values of "p_{RMSE}" and "v_{RMSE}") will be considered.

Since the data could not be obtained from simulations in the arena, the results regarding RMSE (Root-Mean Square Error) for both the position and velocity will be obtained from the reference values and the respective simulations in software. Therefore, the following values were obtained:

- $p_{RMSE_{step-movement}} = 0.1151$;
- $v_{RMSE_{step-movement}} = 0.1437$;
- $p_{RMSE_{circular\ flight}} = 0.1393$;
- $p_{RMSE_{circular\ flight}} = 0.1471$.

3.11. Goal number 3.1

Implement a planning algorithm of your choice that is able to drive the drone from any initial position ("$p_{initial} \in \mathcal{P}_{start}$") to a first goal position ("$p_1 \in \mathcal{P}_{goal_1}$"), to a second goal position ("$p_2 \in \mathcal{P}_{goal_2}$") and back to the original starting point, ensuring the drone is always within "$\mathcal{P}_{fly-zone}$" while avoiding "$\mathcal{P}_{no-fly-zone_1}$" and "$\mathcal{P}_{no-fly-zone_2}$".

In order to implement a planning algorithm (in this case a sample-based algorithm, removed from a GitHub database repository provided by Professor Bruno Guerreiro called "$RRT3D$"). From this code made available, the obstacles environment was developed in order to find a way through the arena.

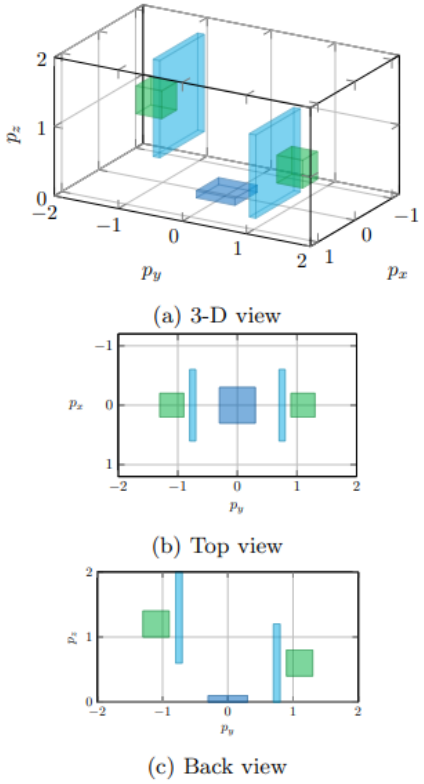

Figure 32 – Flight arena planning setup

The flight path desired is the following:

1. From the initial position of coordinates $(0, 0, 0)$ to the first goal position $(0, 1.1, 0.6)$ while avoiding the obstacles;
2. From the first goal position $(0, 1.1, 0.6)$ to the second goal position first goal position $(0, -1.1, 1.2)$;
3. From the second goal position $(0, -1.1, 1.2)$ to the initial position $(0, 0, 0)$.

Developing the software "$RRT3D$", the following figures were obtained as the result of the simulation.

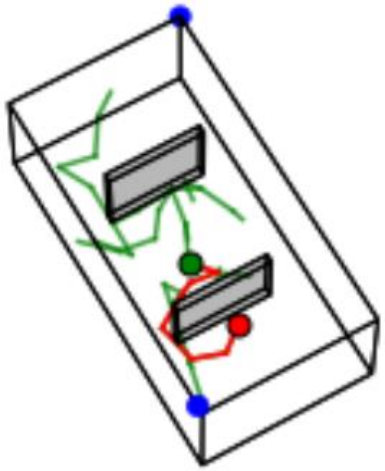

Figure 33 – First part of the path developed by the software "RRT3D"

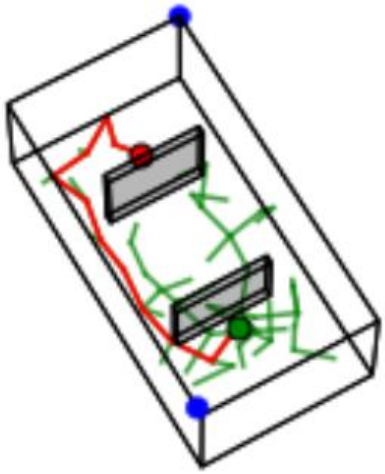

Figure 34 – Second part of the path developed by the software "RRT3D"

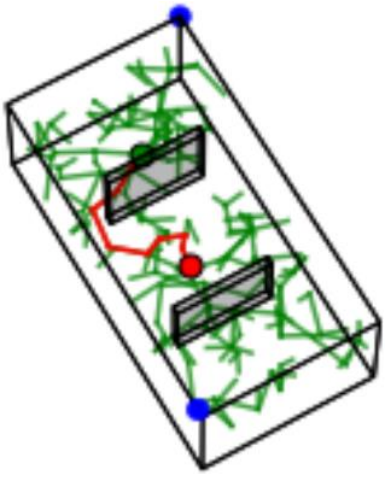

Figure 35 – Third part of the path developed by the software "RRT3D"

The software reached the goals it was set up for, visualized with the red paths.

3.12. Goal number 3.2

Test this algorithm with the controller developed in simulation, noting that "$p_{initial}$", "p_1" and "p_2" are randomly chosen from their respective sets, and describe the experimental tasks.

Unfortunately, the group was unable to realize this task in time.

3.13. Goal number 3.3

Test the planning algorithm and controller in the flying arena.

Unfortunately, the group was unable to realize this task in time.

4. Conclusions and Other Commentaries

During the development of this project, the group got to understand complexity of the systems control of unmanned autonomous vehicles and experience the kind of work that goes into every project in the aerospace area. Even though this work is a very simplified project in comparison to the work of the corporations worldwide, the challenge of this project allowed the students to learn and practice about the control models and modelling techniques, linear and nonlinear control, LQR controllers, how to deal with nonlinearities and how to search the ideal algorithm to find the best path that a drone should follow in order to reach a goal position without running into obstacles.

The results obtained were as expected, with the linearized models acting in accordance with what was expected and representing what the group believes that should be the dynamics of the drone. The controllers obtained were able to guarantee the drone to follow the wanted reference and, with the error state model and the nonlinear model in order to minimize its error against the wanted reference. The nonlinear model shows a fast response to the errors of position and find an adequate path for the Crazyflie 2.1 drone to reach the wanted positions without colliding in the simulation walls.

With the project coming to an end, it’s important to refer that the three parts of the project enhanced the skills of the group in order to produce better engineers for the future work environments, both in aerospace engineering and in other markets with similar needs, and both in Portugal or anywhere else in the world.

5. Webgraphy

[1] – Captain Brian P. Tice, "Unmanned Aerial Vehicles" – https://web.archive.org/web/20090724015052/http://www.airpower.maxwell.af.mil/airchronicles/apj/apj91/spr91/4spr91.htm

[2] – EASA (European Union Aviation Safety Agency) website, "Open Category – Low Risk – Civil Drones" where we checked the classification of a Micro Aerial Vehicle – https://www.easa.europa.eu/en/domains/civil-drones-rpas/open-category-civil-drones

[3] – NASA's Earth Fact Sheet where we obtained the Bulk Parameter of Mean Surface Gravity ("$\boldsymbol{g_{earth}} = 9.82\ [^{m}/_{s}]$") and the Terrestrial Atmosphere Surface Density ("$\boldsymbol{\rho_{earth}} = 1.217\ \left[^{kg}/_{m^3}\right]$") – https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

[4] – Sighard F. Horner, "Fluid-Dynamic Drag" where we checked the different values of drag coefficient for various 2D shapes in order to attribute "$c_d = 1$" to the Crazyflie 2.1 code – https://archive.org/details/FluidDynamicDragHoerner1965/mode/2up

6. Attachments

Github repository by José Corvo:

- Link for the Main Folder – https://github.com/Jose-Corvo-Lv99/UAV-s_Grupo3_FCT_2023-24