UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Bachelor's thesis

Control of a multicopter based on differential flatness

August 2023

Author: Yara Abreu Infante

Tutor: Eduardo García Breijo

External cotutor: Jerome Jouffroy

Abstract

Conventional PID controllers can only keep a drone stable around its balance position, which means they are not effective in handling high pitch and roll angles due to strong disturbances. Also, these controllers do not allow aggressive maneuvers. To address this problem, this project focuses on the design of a trajectory control system for multicopters based on differential flatness theory, which considers all system nonlinearities. To implement the control system, the tools Matlab and Simulink are used. Finally, the results are evaluated through a simulation to verify the effectiveness of the system.

Key words: Trajectory control, Simulink, differential flatness.

Resumen

Los controladores PID convencionales solo pueden mantener un dron estable alrededor de su posición de equilibrio, lo que significa que no son efectivos para manejar ángulos de inclinación y balanceo elevados debido a fuertes perturbaciones. Además, estos controladores no permiten maniobras agresivas. Para abordar este problema, este proyecto se centra en el diseño de un sistema de control de trayectorias para multicópteros basado en la teoría de la planitud diferencial, que considera todas las no linealidades del sistema. Para implementar el sistema de control, se utilizan las herramientas Matlab y Simulink. Finalmente, se evalúan los resultados mediante una simulación para verificar la eficacia del sistema.

Palabras clave: Control de trayectorias, Simulink, planitud diferencial.

Table of content

Table of figures

Introduction

The realm of nonlinear control has witnessed remarkable progress recently, particularly in tackling the intricate issues surrounding trajectory generation and tracking within complex systems. At the heart of these endeavors lies a pivotal challenge - how to dynamically generate trajectories for differentially flat systems in real time. This demanding task involves devising a delicate balance between upholding stability and enhancing performance, all while accommodating potential computational delays. In the subsequent discourse, we introduce a pair of algorithms poised to address the real-time trajectory generation predicament for differentially flat systems, effectively navigating the intricate terrain of stability, performance, and system intricacies.

Differentially flat systems possess a unique characteristic: they afford the representation of their states and inputs through chosen outputs and their derivatives. This fundamental attribute underpins the very essence of trajectory generation, enabling the parameterization of state space and input trajectories employing the designated outputs as a basis. Remarkably, this achievement materializes even in the face of non-minimum phase and unstable zero dynamics scenarios.

This paper focus on the application of flatness theory to the domain of multicopter systems. Section 2 explains the historical development and evolution of this theoretical framework. Subsequently, in Section 3.1, we articulate the mathematical model characterizing the multicopter system. Proceeding to Section 3.2, we undertake the calculation of feedforward control pertinent to the system. Section 3.3 is dedicated to a the planning of the trajectory.

Following this theoretical groundwork, Section 4 presents the outcomes derived from this research, including graphical representations and simulations. In conclusion, the paper culminates in Section 5 with a summarize of findings, implications, and potential improvements for future work.

Background

Looking into the flatness theory, there are many components that can describe the behaviour of the system. Fields such as mathematics, physics, number system, and even algebraic geometry use flatness theory, however in this report, the main goal is to describe the relation of flatness theory to control theory and dynamical systems. In order to fully understand flatness, Jean Levine describes flatness like this: “To give an intuitive idea of differential flatness, a flat system is a system whose integral curves (curves that satisfy the system equations) can be mapped in a one-to-one way to ordinary curves (which need not satisfy any differential equation) in a suitable space, whose dimension is possibly different than the one of the original system state space.” [1] in other words it is the process of creating simpler solutions out of complex systems, in order to improve design strategies and to understand the system’s behaviour. The term flatness theory was proposed by Michel Fless in the 1990s as a way to describe a mathematical approach known as the Lie–Backlund method to study the equivalence and flatness of nonlinear systems. The authors describe it as: “We have proposed a differential geometric approach for investigating a new system equivalence, the Lie–Backlund equivalence, which can be realized by endogenous feedback, a special type of dynamic feedback. Such a Lie–Backlund equivalence is shown to be useful to reduce the dimension of a complex system and to study differentially or orbitally flat systems." [2].

The Proportional-Integral-Derivative (abbr PID) controller is a feedback control system commonly used in industrial control systems. The main idea is to control the error value of a system, and adjust it accordingly to fit the parameters set by the system. The most common example would be the cruise control set on a car. When the car increases the speed, the cruise control will lower it, and vice versa. In the context of drone applications it becomes increasingly more challenging as the natural elements can cause strong changes of movement that severely adjusts its trajectory, which is supported by Fliess: ”The strange industrial ubiquity of classic PID’s and the great difficulty for tuning them in complex situations is deduced, via an elementary sampling, from their connections with iPIDs.“ [3]. Creating an adjustable trajectory with a modern PID system, that follows the flatness theory will be the key to success in this project.

Another important system is the feedback control, which is important in control theory. It focuses on the continuous adjustment of a system's inputs or parameters based on the system's current state and the desired outcomes. This adjustment process involves comparing the system's actual performance to the preferred performance and then employing corrective actions to minimize error. Feedback loops play a crucial role in maintaining stability, enhancing accuracy, and achieving desired performance in dynamic systems. Hagenmeyer and Delaleau delve into the intricacies of integrating feedback control mechanisms with flatness-based control strategies, optimizing the maneuvering capabilities of multicopters. Given the innate complexity of multicopters as dynamic systems, the incorporation of feedback control mechanisms is important in upholding stability, precision, and flexibility. This exploration likely entails a comprehensive examination of how feedback control seamlessly integrates into the overarching control framework of multicopters, enabling instantaneous adaptations concerning variables like position, orientation, and altitude [4].

The trajectory of the drone will have to change course and allow for multiple natural elements to interact with it. The paper by Michiel van Nieuwstadt and Richard M. Murray, explores the idea of real time trajectory generation with delay, using the input that is received online. The paper presents algorithms for real-time trajectory generation in the context of a general control paradigm for nonlinear systems. The goal is to generate feasible trajectories that allow trade-offs between stability and performance [5]. Using some of the examples in this paper, it can be concluded that while it can be beneficial to gather inputs online, there are some different methods that needs to be used to overcome the challenges of the tradeoff of stability and performance.

Method

Modelling

In this section, differential flatness theory will be applied to the non-linear model of a quadrocopter, whose differential equations are shown below [6].

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{T}{m} \begin{bmatrix} \cos\psi \cdot \sin\theta \cdot \cos\phi + \sin\psi \cdot \sin\phi \\ \sin\psi \cdot \sin\theta \cdot \cos\phi - \cos\psi \cdot \sin\phi \\ \cos\theta \cdot \cos\phi \end{bmatrix} - g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{1}$$

The formula where $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$ represent the linear acceleration in the inertial frame; ϕ, θ and ψ are Euler's angles corresponding to the roll, pitch and yaw angle respectively. T corresponds with the thrust, m and g are the mass of the drone and the gravity acceleration with values of 0.320 kg and 9.81 m/s^2, respectively.

The rotation dynamics of the drone can be expressed with the following equations.

$$J_x\dot{\omega}_x + (J_z - J_y)\omega_z\omega_y = \tau_x \tag{2}$$

$$J_y\dot{\omega}_y + (J_x - J_z)\omega_x\omega_z = \tau_y \tag{3}$$

$$J_z\dot{\omega}_z + (J_y - J_x)\omega_x\omega_y = \tau_z \tag{4}$$

The relation between the angular velocities and the derivatives of the Euler angles is shown below.

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{5}$$

Feedforward control

Flatness theory states that a nonlinear system of the form $\dot{x} = f(x, u)$ will be controllable when two conditions are met:

- There exists a nonlinear function of F (flat output) describing the state x and its derivatives.
- There exists a nonlinear function of F describing the input u and its derivatives.

When selecting the flat outputs, it is important to consider the dimension of the input vector u, as both the inputs and outputs should have the same dimensions.

In the drone example, there are four inputs representing the Euler angles and thrust. Consequently, the vector F should also have a dimension of four. The flat outputs will include the x, y, and z position coordinates, along with the yaw angle. These outputs are chosen to represent the direction the drone follows in its trajectory.

$$F = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix} \tag{6}$$

After identifying the flat outputs, our next step involves representing each of the control inputs using a function of the vector F and its derivatives. By working with the equations in (1) and solving for each input, we arrive at the following expressions.

$$T = m\sqrt{\ddot{F}_1^2 + \ddot{F}_2^2 + (\ddot{F}_3^2 + g)^2} \tag{7}$$

$$\theta = \arctan\frac{\ddot{F}_1 \cos F_4 + \ddot{F}_2 \sin F_4}{\ddot{F}_3 + g} \tag{8}$$

$$\phi = \arctan\frac{\ddot{F}_1 \sin F_4 - \ddot{F}_2 \cos F_4}{\sqrt{(\ddot{F}_3^2 + g)^2 + (\ddot{F}_1 \cos F_4 + \ddot{F}_2 \sin F_4)^2}} \tag{9}$$

Thus, the vector of control inputs would be as follows:

$$u = \psi_u\left(F, \dot{F}, \ddot{F}\right) = \begin{bmatrix} T \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{10}$$

To check the correct functioning of the feedforward control, it is tested in Simulink by following a reference path that is detailed in section 3.3.

Trajectory generation

When planning a trajectory, it is essential to not only consider the way-points but also the transitions between them. A trajectory is considered smooth when these transitions are continuous, avoiding abrupt changes in position, velocity and even acceleration. This smoothness is crucial in fields such as robotics, where aggressive manoeuvres can degrade system performance and lead to mechanical or electronic wear and tear.

The cubic spline is one of the most powerful and well-known tools for generating smooth curves. It is a data interpolation technique that generates different third-degree polynomial segments that meet at user-specified points, also known as nodes, to provide a smooth and continuous transition between them. The advantage of cubic splines is that they can be derived twice, guaranteeing smoothness in position values and speed. However, this technique does not allow to specify the desired speed values, which is possible with a higher degree polynomial. However, increasing the degree of the polynomial can lead to an increase in the curve oscillation. Therefore, it is essential to consider the needs of the application in order to choose the most appropriate degree.

Each segment of the curve is represented as follows [7]:

$$s_j(t) = a_j + b_j t + c_j t^2 + d_j t^3 \tag{11}$$

where $j = 1,2,..., n$-1, being n the number of segments, and the parameters a, b, c and d, the coefficients of the polynomial.

Knowing that (11) represents the position values, deriving it gives the velocity and acceleration equations.

$$s'_j(t) = b_j + 2c_j t + 3d_j t^2 \tag{12}$$

$$s''_j(t) = 2c_j + 6d_j t \tag{13}$$

The trajectory designed in this project consists of eight points, which is a seven segments curve. This means that the value of 28 coefficients must be found. To do so, a matrix equation system with the following form must be solved:

$$A \cdot X = b \tag{14}$$

where A is an invertible square matrix of dimension 28.

Since the positions for each instant are known, the first equations will be obtained using (11).

$$s_j(t_{j-1}) = a_j + b_j t_{j-1} + c_j t_{j-1}^2 + d_j t_{j-1}^3 = f(t_{j-1}) \quad (15)$$

$$s_j(t_j) = a_j + b_j t_j + c_j t_j^2 + d_j t_j^3 = f(t_j) \quad (16)$$

From (15) and (16), 14 equations are obtained, where $f(t_{j-1})$ and $f(t_j)$ represent the positions at instants $t_{j-1}$and t_j.

It is then necessary to ensure that the velocity at the nodes is the same for the adjacent segments. To do that, (12) is used as follows.

$$s'_j(t_j) = s'_{j+1}(t_j) \quad (17)$$

which can also be expressed as

$$s'_j(t_j) - s'_{j+1}(t_j) = 0 \quad (18)$$

Only six equations are obtained for this case since the initial and final instants are not used.

For the acceleration, the same process is repeated using the following equation.

$$s''_j(t_j) - s''_{j+1}(t_j) = 0 \quad (19)$$

So far, 26 equations have been obtained, so to get the two left, natural boundary conditions are defined by making the acceleration equal to 0 at the beginning and end of the trajectory.

$$s''_j(t_j) = 0 \quad (20)$$

$$s''_j(t_n) = 0 \quad (21)$$

Solving (14) for X would give the coefficients of the polynomials.

Results

Reference trajectory

The desired trajectory consists of 8 points, which led to the definition in MATLAB of three position vectors for each coordinate: *pos_x* = [1, 2, 3, 4, 5, 5, 4, 3], *pos_y* = [1, 2, 2, 4, 5, 6, 7, 7], and *pos_z* = [1, 1, 1, 1, 2, 4, 2, 2]. The time instances corresponding to each point were established using the vector *T* = [0, 2, 3, 4, 5, 7, 9, 10].

By employing these parameters within the 'coeffcalculation()' function, we successfully derived the coefficients required for generating curves. Utilizing a Matlab function block within Simulink (see figure 1), we were able to visualize the resulting position, velocity, and acceleration curves (figures 2, 3 and 4).

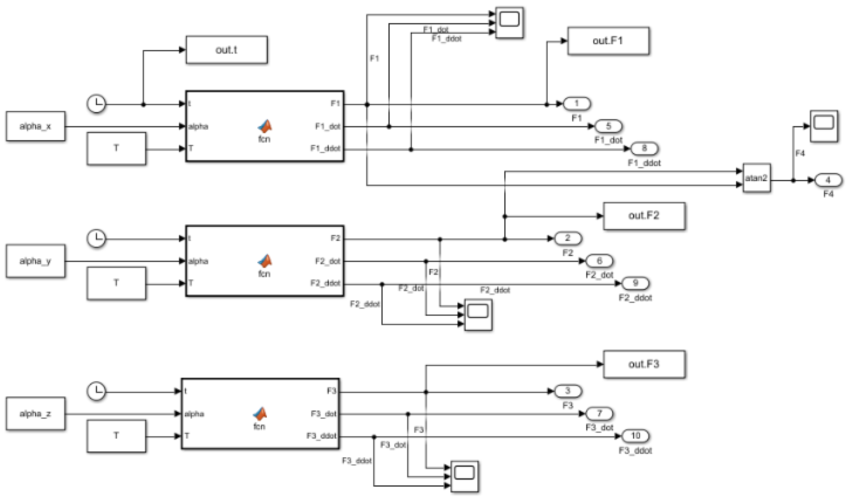

Figure 1. *Trajectory generation block in Simulink*

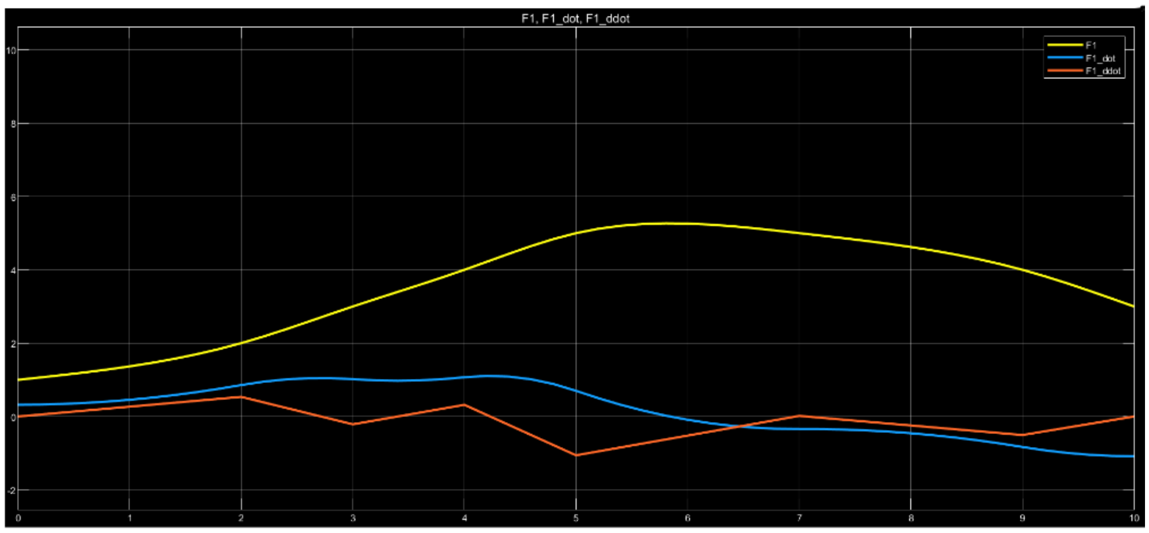

Figure 2. *Desired position, velocity and acceleration of x coordinate*

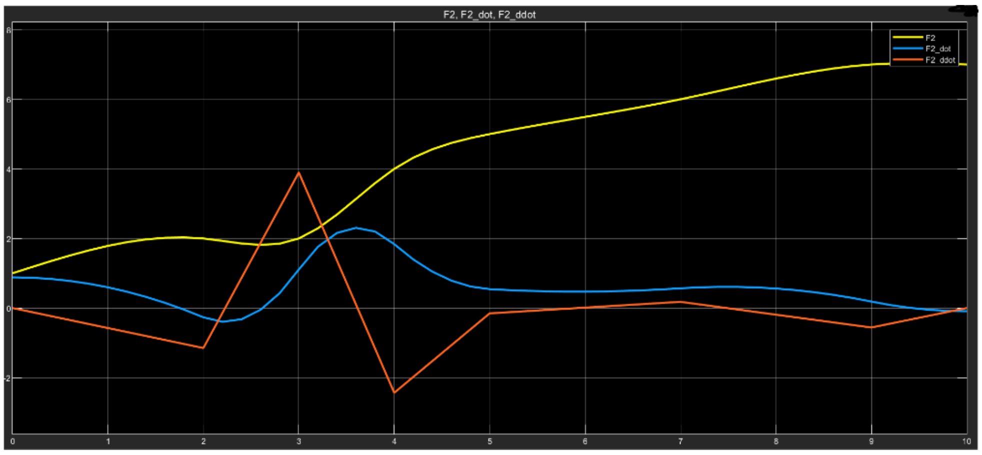

Figure 3. *Desired position, velocity and acceleration of y coordinate*

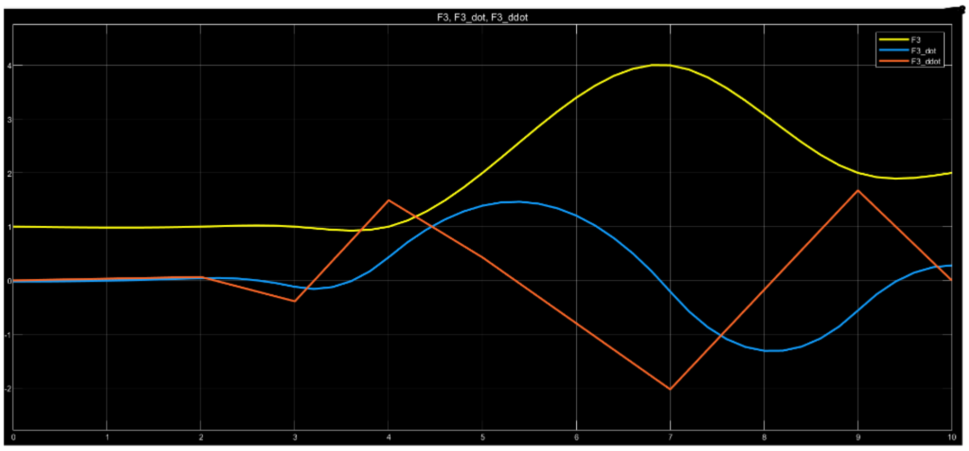

Figure 4. *Desired position, velocity and acceleration of z coordinate*

The yaw angle is intricately tied to the positional values of the x and y coordinates, representing the vehicle's forward direction. To obtain its temporal values, we computed the arctangent of the y-position divided by the x-position, as depicted in figure 1. This process yielded the yaw angle curve illustrated in figure 5, expressed in radians.

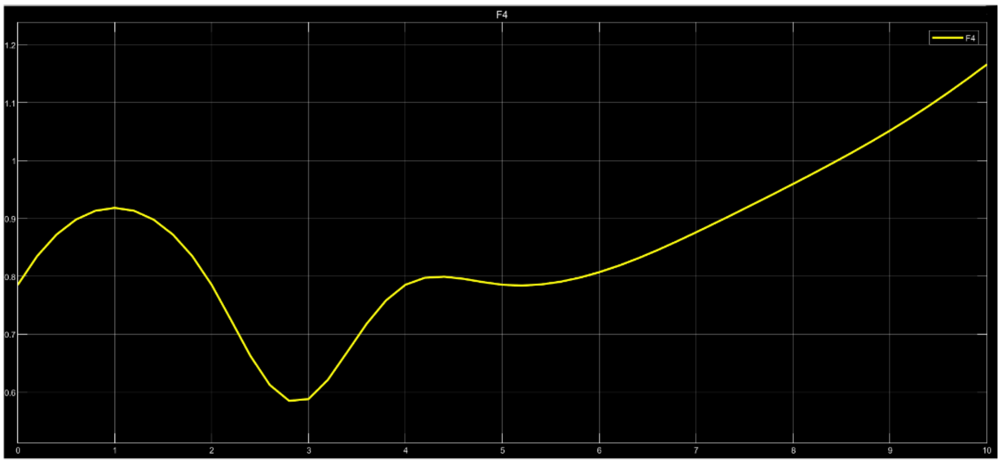

Figure 5. *Desired variation of the yaw angle*

Feedforward control

After acquiring the reference signals associated with the flat outputs, a subsystem was developed in Simulink. Within this subsystem (figure 6), control inputs were derived by employing mathematical blocks based on the equations 7, 8 and 9.

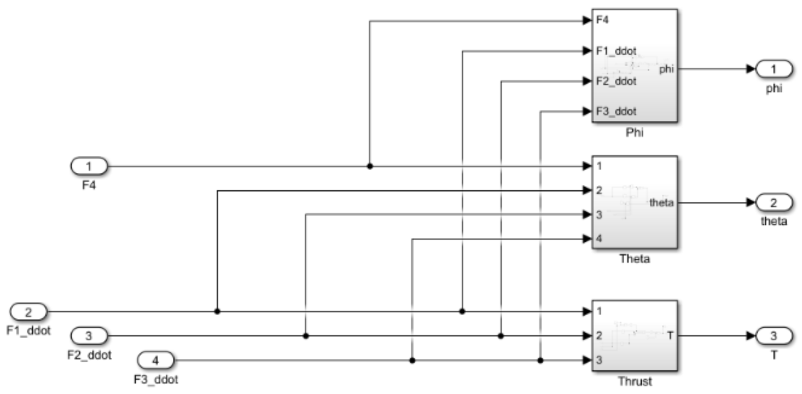

Figure 6. *Simulink block for the control inputs*

Following the generation of control inputs, a feedforward control was executed based on the drone model defined in (1). This led to the calculation of accelerations in the x, y, and z directions. For acquiring velocities and positions, simple integration of these accelerations is good enough.

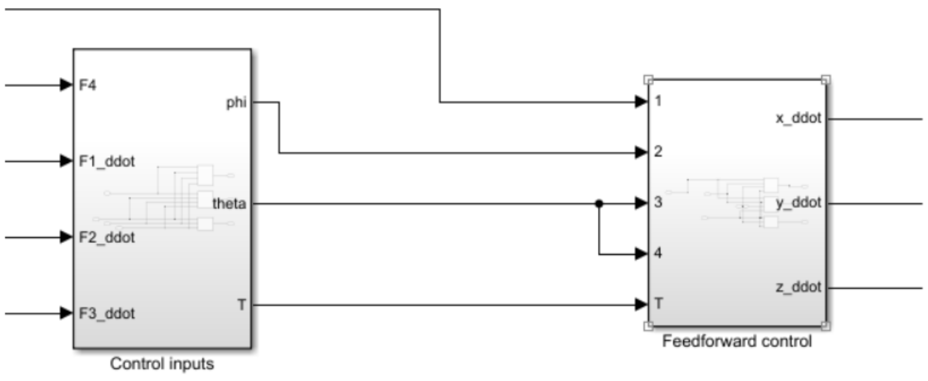

Figure 7. *Simulink block for feedforward control*

To validate the efficacy of the feedforward control, a comparison was drawn between the reference signal graphs and those obtained post-control. The graphical representations (figures 8, 9 and 10) demonstrated an identical match, confirming the functionality of the control. Notably, this system is ideal – devoid of external influences like air friction or wind force – resulting in the absence of discrepancies between the obtained graphs.

However, in an experimental application of the control system to an actual drone, incorporating feedback control would be indispensable to rectify any deviations caused by real-world factors.

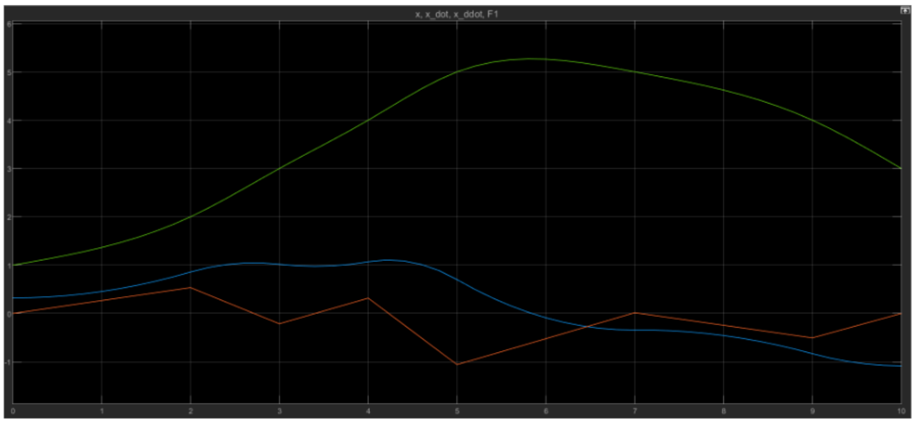

Figure 8. *Position, velocity and acceleration of x*

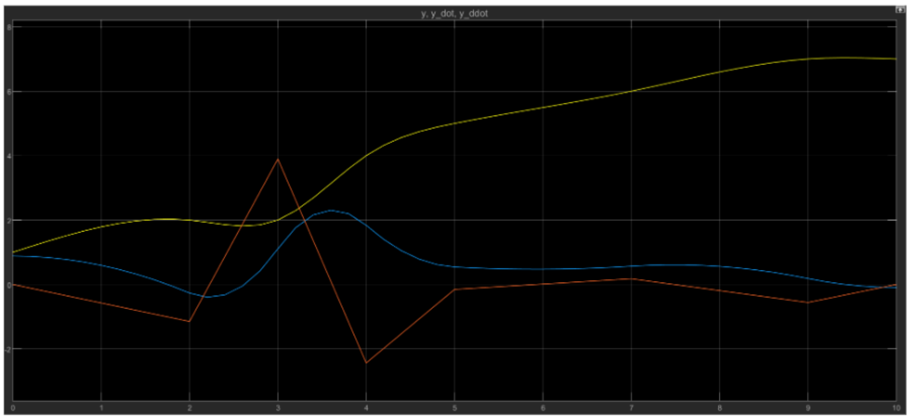

Figure 9. *Position, velocity and acceleration of y*

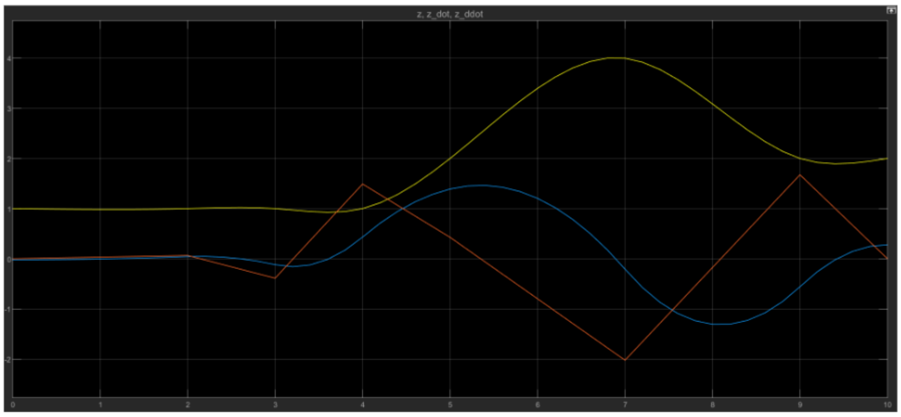

Figure 10. *Position, velocity and acceleration of z*

Animation

Finally, a 3D graph was made showing the desired trajectory and the trajectory obtained.

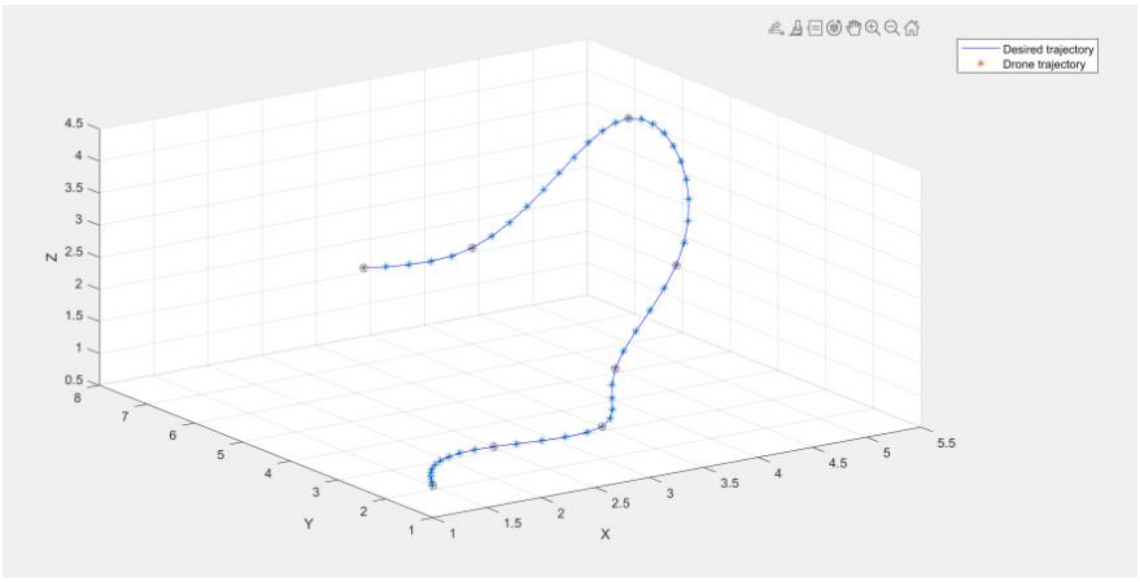

Figure 11. *Trajectory in three dimensions*

As can be seen in Figure 11, and confirming the graphs shown above, the final trajectory of the drone follows perfectly the desired path.

When planning the flight of the drone, it must be ensured that the turning angles are not aggressive. Therefore, to check that the drone flight is smooth, an animation was made using Matlab, where it is possible to observe not only the translational movement of the drone, but also its rotation.

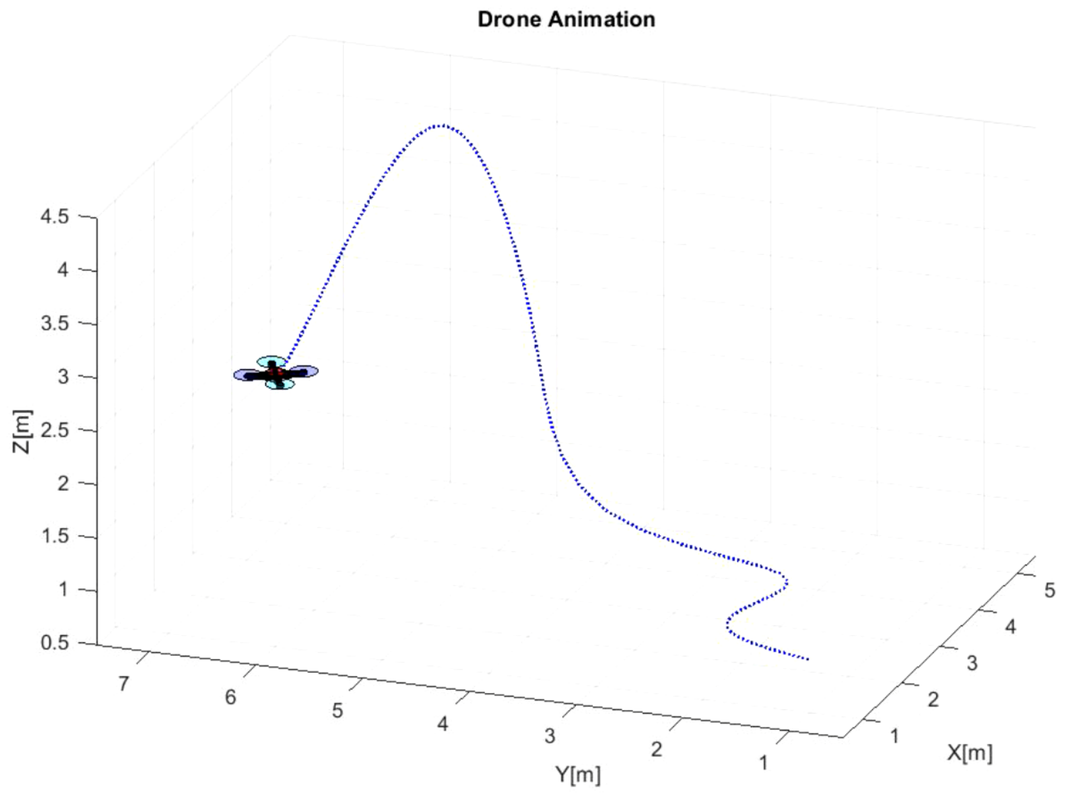

Figure 12. *Drone animation*

From the animation it was possible to check the complete flight of the drone as well as its turning angles.

Conclusion

In conclusion, the field of nonlinear control is advancing, particularly in the realm of making complex systems follow desired paths in real-time. This study focuses on a method that takes advantage of the unique characteristics of these systems, allowing us to describe and control them effectively by selecting specific measurements as reference points. To validate this approach, a simulation and animation were made where we could see the accuracy with which the drone followed the desired path.

However, the conditions in which the drone was tested were ideal, since it is a simulation where the effect of external aspects such as air friction or sudden disturbances is not taken into account. For this reason, the implementation of feedforward control alone was sufficient for this case. If the drone were going to be tested in the real world, it would be convenient to also design a feedback control, which corrects the trajectory errors caused by these external disturbances or noise caused by the system itself.

Thus, combining traditional feedback control with flatness-based methods enhances the agility and precision of multicopter maneuvers.

References

[1] Jean Lévine. *Analysis and Control of Nonlinear Systems A Flatness-based Approach*. May 2009.

[2] M. Fliess et al. “A Lie-Backlund approach to equivalence and flatness of nonlinear systems”. In*: IEEE Transactions on Automatic Control* 44.5 (1999), pp. 922-937.

[3] Michel Fliess and Cédric Join. “Model-free control”. In: *International Journal of Control* 86 (May 2013).

[4] Veit Hagenmeyer and Emmanuel Delaleau. “Exact feedforward linearization based on differential flatness”. In: 76 (2003), pp. 537-556.

[5] Michiel van Nieuwstadt and Richard M. Murray. “REAL TIME; TRAJECTORY GENERATION FOR DIFFERENTIALLY FLAT SYSTEMS”. In: 29 (1996), pp. 2301-2306.

[6] Armando Sanca, Pablo Alsina, and Jes Cerqueira. “Dynamic Modelling of a Quadrotor Aerial Vehicle with nonlinear Inputs”. In: *Latin American Robotics Symposium and Intelligent Robotics Meeting* 0 (Oct. 2008), pp. 143-148.

[7] A Jaramillo-Botero, JF Correa, and IJ Osorio. “Trajectory planning in ROBOMOSP”. In: *Robotics and Automation Group* (2004).