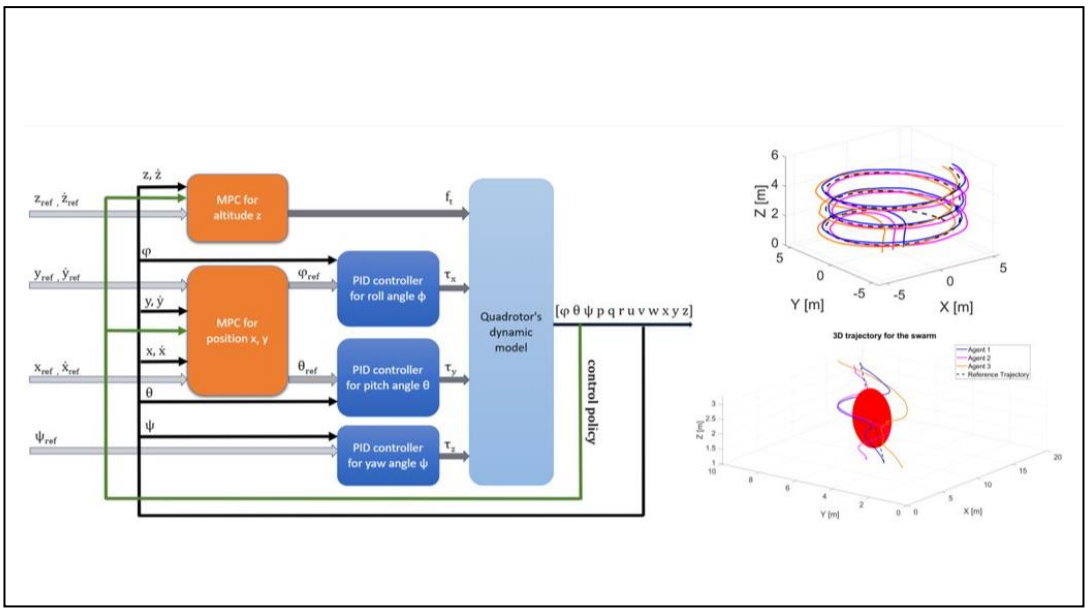
UNIVERSITY OF WEST ATTICA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

Development of a distributed model predictive control framework for autonomous unmanned aerial vehicle swarms



Student: Vavelidou Despoina - Panagiota
Registration Number: 18387084

Supervisor

Alex Alexandridis
Professor

ATHENS-EGALEO, October 2023

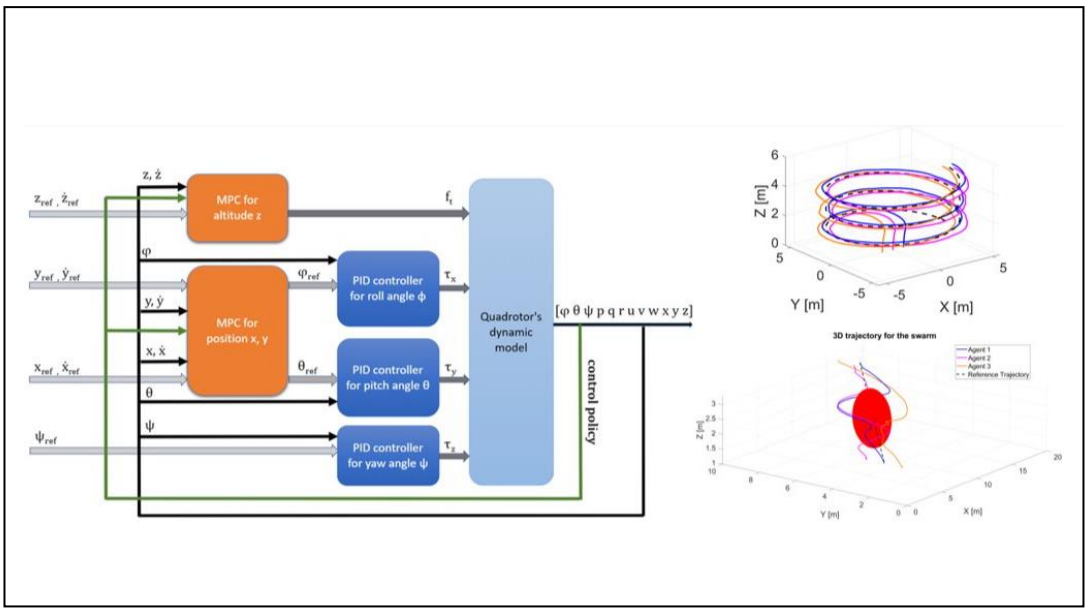
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Ανάπτυξη ενός πλαισίου κατανεμημένου προβλεπτικού ελέγχου για σμήνη αυτόνομων μη επανδρωμένων εναέριων οχημάτων



Φοιτήτρια: Δέσποινα – Παναγιώτα Βαβελίδου
ΑΜ: 18387084

Επιβλέπων Καθηγητής

Αλέξανδρος Αλεξανδρίδης
Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, Οκτώβριος 2023

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Αλεξανδρίδης Αλέξανδρος, Καθηγητής (Επιβλέπων)	Κανδρής Ξενοφών-Διονύσιος, Καθηγητής	Πυρομάλης Δημήτριος, Αναπληρωτής Καθηγητής
Digitally signed by Alexandros Alexandridis Date: 2023.10.16 13:26:14 +03'00' (Υπογραφή)	XENOFON DIONYSIOS KANDRIS Digitally signed by XENOFON DIONYSIOS KANDRIS Date: 2023.10.16 11:26:34 +03'00' (Υπογραφή)	Dimitrios Pyromalis Digitally signed by Dimitrios Pyromalis Date: 2023.10.16 13:14:04 +03'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη Δέσποινα – Παναγιώτα Βαβελίδου του Κωνσταντίνου, με αριθμό μητρώου 18387084 φοιτήτρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.
Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι 12 μήνες και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος καθηγητή.»

Η Δηλούσα
ΔΕΣΠΟΙΝΑ – ΠΑΝΑΓΙΩΤΑ ΒΑΒΕΛΙΔΟΥ

Digitally signed by Alexandros Alexandridis
Date: 2023.10.16 13:26:35 +03'00'

Acknowledgements

I would like to begin by expressing my utmost gratitude to Dr. Alex Alexandridis, whose mentorship has played a pivotal role in shaping my academic journey. Not only has he been an exceptional professor, whose profound passion for the subject ignited my own, but his guidance as my thesis supervisor has been transformative. I am forever thankful to him for entrusting me with the opportunity to explore such an intriguing topic, his willingness to invest time in discussions, provide feedback, offer support, and share his insightful approach that made this academic pursuit both enriching and rewarding. I consider myself truly fortunate to have had the privilege of learning from him, and I am eager to carry this knowledge into my future endeavors.

I also want to express my heartfelt appreciation to Teo Protoulis, a dedicated doctoral student whose contribution to my diploma thesis has been immeasurable. From day one, Teo has been an unwavering source of support, providing continuous help at any hour, offering his extensive knowledge and valuable feedback to enhance my understanding and further refine my thesis, giving constant encouragement, and consistently going above and beyond expectations. His active involvement in the entire process not only enriched my learning experience but also proved indispensable for the successful completion of my thesis. I am genuinely grateful for every aspect of his support and for his contagious enthusiasm for Automatic Control Systems, which inspired and motivated me throughout this journey.

Finally, I would like to thank my friends and my family, especially my sister Ioanna, for their never-ending support throughout my university years.

Abstract

Unmanned Aerial Vehicle (UAV) technology has witnessed remarkable advancements, permeating various industries. Quadcopters, a specialized type of UAV with four rotors, are at the forefront of these innovations with recent trends and future projections indicating that the utilization of multiple quadcopters, forming what it is referred to as a swarm of quadcopters, is poised for substantial growth. However, the inherently nonlinear and highly complex behavior of quadcopters introduces serious challenges in terms of efficiently controlling them, necessitating the need of developing advanced control methods and intricate frameworks, particularly when addressing the collective behavior of a swarm. In response to these challenges, this diploma thesis introduces a comprehensive Distributed Model Predictive Control (DMPC) framework, designed to enable precise trajectory tracking for a swarm of quadcopters. Building upon the initial introduction of a Model Predictive Control (MPC) scheme for addressing the trajectory tracking task of a single quadcopter, which employs MPCs and Proportional-Integral-Derivative (PID) controllers in a cascaded design, the proposed framework seamlessly scales its capabilities. Through the integration of an inter-agent communication strategy, it achieves the transition to effective, autonomous and distributed control over quadcopter swarms, even in complex scenarios such as trajectory tracking with collision avoidance and obstacle evasion. The distributed nature of the proposed approach eliminates the necessity for a central controller, thus enhancing the system's resilience in the event of individual failures, or communication disruptions, and offers easy adaptation to various swarm sizes. Additionally, the proposed framework diverges from conventional methods by liberating the agents of the swarm from following predetermined trajectories and predefined formation strategies. Instead, it empowers agents to dynamically tailor their behavior, enabling them to closely follow real-time desired trajectories with responsive agility to unpredicted scenarios. To evaluate the efficiency and practicality of the proposed framework, simulations have been conducted for trajectory tracking, including scenarios involving suddenly appearing obstacles. The results endorse its suitability for quadcopter swarm control and illuminate its potential application across a broad spectrum of domains, where the swarm's precision, adaptability and robustness are paramount.

Keywords

Agent swarms, cascaded design, collision avoidance, Distributed Model Predictive Control (DMPC), inter-agent communication, Model Predictive Control (MPC), obstacle avoidance, Proportional-Integral-Derivative (PID) controller, quadcopter, trajectory tracking, Unmanned Aerial Vehicles (UAVs).

Περίληψη

Η τεχνολογία των Μη Επανδρωμένων Εναέριων Οχημάτων (UAV) έχει γνωρίσει αξιοσημείωτες εξελίξεις, διεισδύοντας σε διάφορους κλάδους. Τα τετρακόπτερα, ένα εξειδικευμένο είδος UAV με τέσσερις ρότορες, βρίσκονται στο προσκήνιο αυτών των καινοτομιών, με τις πρόσφατες τάσεις και τις μελλοντικές προβλέψεις να δείχνουν ότι η χρήση πολλαπλών τετρακόπτερων, που σχηματίζουν αυτό που αναφέρεται ως σμήνος τετρακόπτερων, βρίσκεται σε πορεία σημαντικής ανάπτυξης. Ωστόσο, η εγγενώς μη γραμμική κι ιδιαίτερα πολύπλοκη συμπεριφορά των τετρακόπτερων εισάγει σοβαρές προκλήσεις σχετικά με τον αποτελεσματικό έλεγχό τους, γεγονός που καθιστά αναγκαία την ανάπτυξη προηγμένων μεθόδων ελέγχου και πολύπλοκων πλαισίων, ιδίως όταν πρόκειται για τη συλλογική συμπεριφορά ενός σμήνους. Ως απάντηση σε αυτές τις προκλήσεις, η παρούσα διπλωματική εργασία εισάγει ένα ολοκληρωμένο πλαίσιο Κατανεμημένου Προβλεπτικού Ελέγχου Μοντέλου (DMPC), σχεδιασμένο για να επιτρέπει την επακριβή παρακολούθηση τροχιάς για ένα σμήνος τετρακόπτερων. Στηριζόμενο στην αρχική εισαγωγή ενός συστήματος Προβλεπτικού Ελέγχου Μοντέλου (MPC) για την αντιμετώπιση του ζητήματος της παρακολούθησης τροχιάς ενός τετρακοπτέρου, το οποίο χρησιμοποιεί MPCs και ελεγκτές Αναλογικού-Ολοκληρωτικού-Διαφορικού μέρους (PID) σε αλυσιδωτή σχεδίαση, το προτεινόμενο πλαίσιο κλιμακώνει απρόσκοπτα τις δυνατότητές του. Μέσω της ενσωμάτωσης μιας στρατηγικής για την επικοινωνία μεταξύ των πρακτόρων του σμήνους, επιτυγχάνεται η μετάβαση σε αποτελεσματικό, αυτόνομο και κατανεμημένο έλεγχο σμήνους τετρακόπτερων, ακόμα και σε περίπλοκα σενάρια όπως η παρακολούθηση τροχιάς με αποφυγή σύγκρουσης και εμποδίων. Ο κατανεμημένος χαρακτήρας της προτεινόμενης προσέγγισης εξαλείφει την ανάγκη για έναν κεντρικό ελεγκτή, ενισχύοντας έτσι την ανθεκτικότητα του συστήματος σε περίπτωση μεμονωμένων αστοχιών ή διαταραχής της επικοινωνίας και προσφέροντας εύκολη προσαρμογή σε διάφορα μεγέθη σμήνους. Επιπλέον, το προτεινόμενο πλαίσιο αποκλίνει από τις συμβατικές μεθόδους απελευθερώνοντας τους πράκτορες του σμήνους από το να ακολουθούν προκαθορισμένες τροχιές και προσχεδιασμένες στρατηγικές σχηματισμού. Αντ' αυτού, δίνει την δυνατότητα στους πράκτορες να προσαρμόζουν δυναμικά την συμπεριφορά τους, επιτρέποντάς τους να ακολουθούν στενά τις επιθυμητές τροχιές σε πραγματικό χρόνο με ευέλικτη ανταπόκριση σε απρόβλεπτα σενάρια. Για την αξιολόγηση της αποτελεσματικότητας και της πρακτικότητας του προτεινόμενου πλαισίου, πραγματοποιήθηκαν προσομοιώσεις για την παρακολούθηση τροχιάς, συμπεριλαμβανομένων σεναρίων που περιλαμβάνουν ξαφνικά εμφανιζόμενα εμπόδια. Τα αποτελέσματα επικυρώνουν την καταλληλότητά του για τον έλεγχο σμήνους τετρακόπτερων και φωτίζουν την δυνατότητα εφαρμογής του σε ένα ευρύ φάσμα τομέων, όπου η ακρίβεια, η προσαρμοστικότητα κι η ανθεκτικότητα του σμήνους είναι υψίστης σημασίας.

Λέξεις – κλειδιά

Αλυσιδωτή σχεδίαση, Αναλογικός-Ολοκληρωτικός-Διαφορικός ελεγκτής, αποφυγή εμποδίου, αποφυγή σύγκρουσης, επικοινωνία μεταξύ πρακτόρων, Κατανεμημένος Προβλεπτικός Έλεγχος Μοντέλου, μη επανδρωμένα εναέρια οχήματα, παρακολούθηση τροχιάς, Προβλεπτικός Έλεγχος Μοντέλου, πράκτορες του σμήνους, τετρακόπτερο.

Table of Contents

List of Tables

List of Figures

INTRODUCTION

The realm of UAVs has witnessed remarkable progress in recent years, owing to their industrial and commercial applications, as well as their use in mobile edge computing, cellular communication, smart healthcare, remote sensing or relief operations in disaster environments. Their successful use in different scenarios has presented the promising avenue for UAVs to achieve even more complex missions and revolutionize modern technology and industry [1].

A special type of UAV, belonging to the broader category of aerial vehicles known as multicopters or multirotors, is the quadcopter. Quadcopters are equipped with four independently controlled rotors, organized into two diametrically opposing groups. Essentially, the way of controlling the position and orientation of a quadrotor is through the variation of the speed of each rotor, and, thus, the variation of the produced forces and moments along the three axes of the 3D space. Overall, the quadrotor is a highly nonlinear underactuated system, with six degrees of freedom and only four inputs, presenting intricate dynamics, a sensitive behavior and intertwined terms [2]. The synergy of internal complexities and external factors, such as wind disturbances or environmental uncertainties, renders accurate trajectory tracking a challenging task for the quadcopter, which demands sophisticated control algorithms and rapid response capabilities.

Despite their popularity and due to the challenges mainly associated with the trajectory tracking task, there is a wide range of applications that are challenging to achieve with a single quadcopter, thus necessitating the need of a swarm of drones that use collective behavior [3]. Swarm of quadcopters has been one of the most recent and prominent examples of swarm robotics, finding success in dealing with precision, reliability and swiftness with complicated tasks, such as search-and-rescue missions, precision agriculture or environmental monitoring [4]. This comes with the need for advanced control strategies that will enable the effective operation of quadcopter swarms in dynamic and unconstructed environments.

Managing the control of a single quadcopter, let alone orchestrating the coordination of a swarm of quadcopters with multiple agents, presents a multifaceted challenge. When confronted with the intricacies of managing a swarm, it becomes imperative to address not only the inherent complexities involved in controlling individual quadcopters, but also novel challenges. Among these, paramount is the necessity for seamless and efficient communication among swarm agents, a critical factor in averting collisions, both among the agents themselves and with potential obstacles.

This intricate interplay of factors has spurred extensive research endeavors for the development of efficient control methodologies and techniques. More specifically, in literature, linear Proportional-Integral-Derivative (PID) control is one of the most common control techniques for an individual

quadcopter, due to its simplicity [5]. Nevertheless, PID control seems to fall short when dealing with more complicated tasks or with swarms, and it is mainly considered to perform in combination with other techniques. Different control methods that have been proposed for swarm control include Linear Quadratic Regulator (LQR) [6] , backstepping [7], sliding mode [8] and neural networks [9]. However, one of the most prevalent and efficient control techniques for swarm control is Model Predictive Control (MPC) [10][11][12][13], which is an advanced control configuration that utilizes a mathematical model of the system in order to optimize the produced control inputs over a finite time horizon, while also considering and respecting the dynamic constraints imposed. Depending on the application of the swarm, multiple MPC algorithms and frameworks have been developed, such as Distributed MPC or Centralized MPC, along with various strategies for formation control, especially for the trajectory tracking task, which is a fundamental aspect of swarm robotics applications.

In this diploma thesis, a Distributed Model Predictive Control (DMPC) framework is developed for trajectory tracking with obstacle evasion. Expanding on the implementation of Model Predictive Control (MPC) for an individual quadcopter [14] - a design which employs MPCs for the position regulation and Proportional-Integral-Derivative (PID) controllers for the angles' regulation in a cascaded design - this framework is used to orchestrate the collective behavior of a swarm of quadcopters. These are controlled autonomously from individual MPCs in a distributed way, while their communication was achieved through inter-agent sharing of information regarding their predicted positions in the three-dimensional space.

The proposed framework for controlling a quadcopter swarm eliminates the need for a central controller, thus resulting in a more robust and resilient system, in case of individual failure or communication disruption. Moreover, as the swarm size increases, the control framework can accommodate additional agents, without significant modifications, making it suitable for different applications where the number of agents can widely vary. Additionally, since each quadrotor can independently plan its trajectory, while considering the intentions of the other agents, enhanced adaptability is allowed, which makes for precise collision avoidance, obstacle evasion and a safe operation. This framework comes with additional advantages, as well, including reduced bandwidth and energy requirements, improved privacy-security as well as a significantly simpler optimization problem that needs to be solved by each agent in comparison to the centralized framework.

Furthermore, the proposed framework is characterized by the absence of predetermined trajectories for each agent, which deviates from conventional methods adopted by many other researchers in the field [15][16][17]. Instead, each agent dynamically adjusts its behavior to closely follow the real-time desired trajectory, allowing for more agile responses to changing conditions and unforeseen obstacles. This improved adaptability of the swarm in unpredictable scenarios is further

facilitated by the absence of a formation control strategy. The quadrotors are able to autonomously and dynamically determine their positions and find their formations online, making them better equipped to respond dynamically to changes in the trajectory.

The proposed framework underwent evaluation across different simulated scenarios, demonstrating its efficiency not only in complex trajectory tracking, but in scenarios involving the sudden appearance of obstacles, as well.

The structure of the presented diploma thesis is organized as follows:

Chapter 1. provides a comprehensive understanding of the quadcopter, delving into its motion characteristics and the mathematical model that describes its kinematics and dynamics. In this way, it sets the stage for an in-depth investigation of the control strategies.

Chapter 2. explores the application of PID control for a single quadcopter, aiming to gain a deeper understanding of its dynamic behavior. It also highlights areas where PID control may not be ideally suitable, thus paving the way for the exploration of a more advanced control strategy.

Chapter 3. is dedicated to designing an MPC control system for a single quadcopter, which also incorporates aspects of PID control, and aims for precise trajectory tracking. Its evaluation not only serves as a benchmark for the system's capabilities, but also bolsters the argument for its scalability to swarm applications.

Chapter 4. focuses on the implementation of the MPC control system within the context of a swarm of quadcopters, culminating in the development of a Distributed MPC framework. This framework, which is evaluated through different simulations, ensures that the agents are able to follow the trajectory path while avoiding collision with each other and navigating around obstacles.

Chapter 5. is dedicated to the conclusions of the diploma thesis. It encompasses a meticulous summary, a detailed analysis of the results, a comprehensive examination of the challenges encountered, in-depth discussions, and the exploration of potential avenues for further research and expansion.

CHAPTER 1: The dynamic behavior of the quadcopter

In this chapter, a comprehensive overview of the quadrotor is provided, including its basic characteristics, the reference system used, the principles of its space motion, and the mathematical model implemented. Finally, an open-loop simulation is performed to show the model's inherent behavior and the challenges presented. Overall, this chapter serves as a foundation for the subsequent chapters.

1.1 The Quadrotor

A quadrotor, also known as a quadcopter or a quadrotor helicopter, is a type of UAV (unmanned aerial vehicle) that has become increasingly popular in recent years, due to its versatility, maneuverability, and relatively low cost. Unlike traditional helicopters which have one or two large rotors, quadrotors have four independently controlled rotors on the extremities of a rigid frame and an electronic board in the middle. This configuration serves as an advantage, making them useful for a wide range of applications that span many different fields. [2]

The field of quadrotor technology is advancing rapidly and is only expected to continue to grow in the coming years. Some of its most promising applications are in search and rescue, especially for inaccessible or unsafe environments, agriculture, construction, building exploration, mapping, personal use, aerial photography or videography and entertainment. As new technologies continue to be integrated into the design and operation of quadrotors, even more possibilities are emerging, including providing innovative solutions for urban air mobility, environmental monitoring, disaster response or space exploration. [18]

The design of a quadrotor usually follows a cross (x) or a plus (+) configuration with opposing motors rotating in the same direction, as illustrated in Figure **1**. By dividing the motors into two groups with two diametrically opposing motors in each one, the observation is made simpler: for the plus configuration, the front propellers Ω_N and Ω_S rotate clockwise (CW), while rear propellers Ω_W and Ω_E rotate counterclockwise (CCW). On the other hand, for the cross configuration, the propellers are divided into the ones rotating CW (Ω_{NW} and Ω_{SE}) and the ones rotating CCW (Ω_{NE} and Ω_{SW}). This division cancels out the unwanted yawing moment, as spinning all rotors in the same direction would cause the quadrotor to constantly rotate around its z-axis.

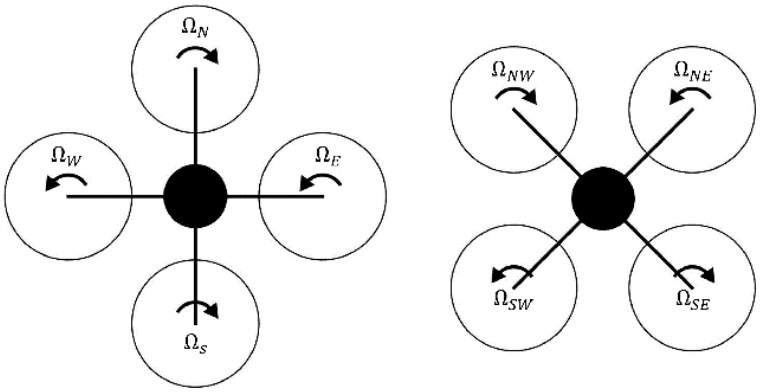


Figure 1. Flight configuration types of the quadrotor

1.2 The Reference System

The reference system provides a standardized frame of reference that allows accurate and consistent design and measurements, as well as effective communication and coordination between multiple vehicles. In the case of the quadrotor, to describe its general position, various body frame reference systems can be used. Some of the most popular ones, used in aerial navigation and control systems design the ABC, the East-North-Up (ENU) or the North-East-Down (NED), which is described below. The selection of a reference system in control design is a contractual agreement between those involved in the design process.

The NED reference system stands for North-East-Down, as it has its three orthogonal axes oriented along the geodetic directions defined by the Earth surface; the x-axis points north parallel to the geoid surface, in the polar direction, the y-axis points east parallel to the geoid surface, along a latitude curve, and the z-axis points downward, towards and antiparallel to the Earth surface, as presented in Figure **2**.

It is a widely recognized and accepted reference system, particularly in aerospace applications, like the quadrotor presented in the current diploma thesis. One plausible reason is that it provides an intuitive representation of position and orientation; humans have used North and East directions to navigate themselves for centuries, while the Down direction aligns with gravity, making it easier to visualize.

While following the NED reference system, it is possible to use the right-hand rule with the thumb pointing downwards and towards the positive z-axis to efficiently portray the positive direction of the three axes. Moreover, the orientation of movement can be represented through the thumb rule,

while also bearing in mind that the axis at which the rotational movement is performed remains constant.

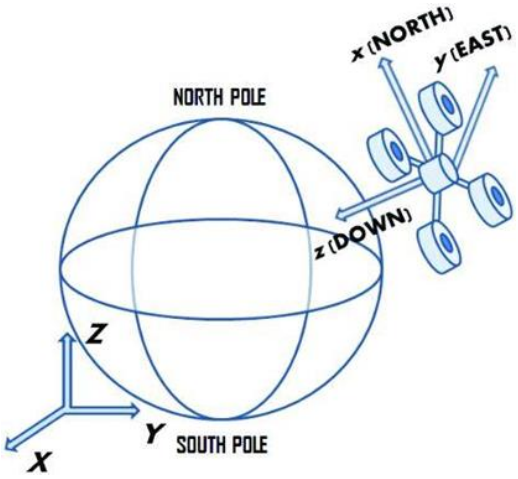


Figure 2. The NED (North-East-Down) reference system for the quadrotor

1.3 The Space Motion

The space motion of the rigid body aircraft can be divided into two parts, which are the barycenter movement and the movement around the barycenter with six degrees of freedom (DOF); three translational and three rotational movements along the three axes. By adjusting the speeds of its four motors, forces and moments such as thrust, rolling, pitching and yawing are produced. More specifically:

- *Thrust*, otherwise known as throttle, refers to the vertical movement of the quadrotor. There are three scenarios: if the thrust provided is equal to the quadrotor's weight (i.e., *mg*) then the quadrotor hovers in place, if the thrust is bigger in value than *mg* then the quadrotor moves upwards and if it is smaller the quadrotor moves downwards. The logic behind this is that the direction of thrust is the direction of motion of the quadrotor, so to change the direction of the vertical motion means altering the thrust.
- *Roll* refers to the rotation around the x-axis of the body frame, so a left or right lateral movement which occurs when propellers 2-3 (of Figure **3**) speed quicker and 1-4 speed slower, or when 1-4 speed quicker and 2-3 speed slower, respectively.
- *Pitch* refers to the rotation around the y-axis of the body frame, so a forward or reverse lateral movement of the quadrotor; for forward motion propellers 3-4 speed quicker and 1-2 speed slower, while for reverse motion the exact opposite holds.

- *Yaw* is the rotation around the z-axis of the body frame, and it can be realized by a reactive torque relative to the rotor speed. It refers to the rotational movement towards the left (CCW) or the right (CW); for CCW motion motors 1-3 speed quicker and 2-4 slower, while for CW motion 2-4 speed quicker and 1-3 slower.

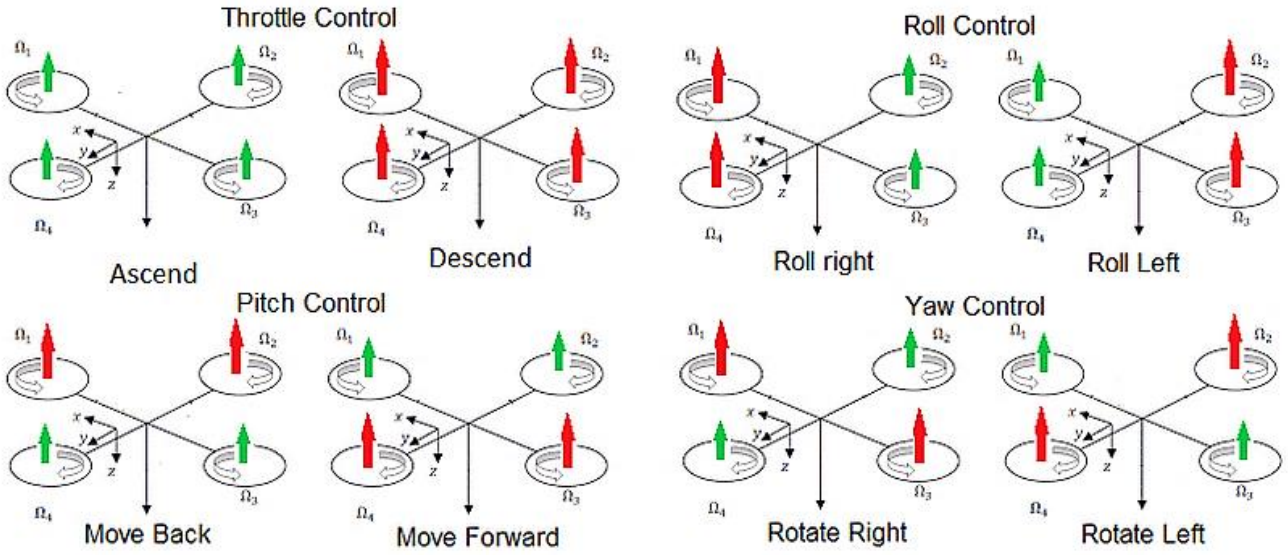


Figure 3. Roll, pitch, and yaw movement for the quadrotor

The quadrotor system is a flight vehicle of lightweight structure and, therefore, gyroscopic effects or moments resulting from the rotation of the rigid body and the four propellers should be included in the dynamic model. Specifically, gyroscopic effect is the ability (or tendency) of the rotating body to maintain a steady direction of its axis of rotation and only appears in the lightweight construction quadrotor. [2]

One crucial point that affects the motion of the quadrotor, especially in outdoor environments, is disturbances, including the effect of wind on its flight dynamics. In the presence of wind, the quadrotor may experience drifting from its intended path or even instability. To address this issue, quadrotors use various techniques, with a common approach being to adjust the pitch and roll angles of the quadrotor to counteract the effects of the wind or using wind sensors to measure the wind speed and direction and then adjusting the motor speeds accordingly. Whatever the case, understanding the impact of wind on quadrotor motion is extremely important for a successful operation. For that reason, it is important to have a mathematical model which accurately represents its dynamics, while

considering the laws that describe its behavior and the numerous possible disturbances it may encounter.

1.4 The Mathematical Model

System modelling is a fundamental aspect of control engineering that involves creating the mathematical representation of real-world systems, by capturing the essential dynamics, behaviors, and relationships within them. A mathematical model is a set of equations that represent the behavior of a system, using differential equations, transfer functions, state-space representations, or other mathematical formalisms, as derived from physical principles, empirical data, or a combination of both.

That being said, it is important to highlight the fact that a model can never perfectly replicate real systems, as they often present complexities, uncertainties or unmodeled dynamics that make creating an exact replica in a mathematical form impossible. So, the goal is to create a representation that closely approximates the behavior of the real system, thus enabling control design and analysis.

In the context of the quadrotor, it is a system that has a highly nonlinear and time-varying behavior, making it a complex dynamic system. Its complicated design creates the necessity of certain assumptions for the creation of its mathematical model and thus its control:

1. The quadrotor and its propellers are rigid bodies.
2. The structure is ideally symmetric, so the moment of inertia tensor has just diagonal inertia terms.
3. There is an inertia frame and a body-fixed reference frame.
4. The origin and axes of the body-fixed frame coincide with the barycenter and the axis of the quadrotor.
5. The ground effect is ignored.[2][19]

The dynamics of the quadrotor can be adequately described through a mathematical model, derived from the application of Newton's law of motion or the Euler-Lagrange methodology, in the form of a set of first-order differential equations that relate the accelerations to the forces and torques.

The orientation of a quadcopter at each time instant can be described using numerous formalisms, one of the most widely known being the Euler angles. A set of three angles, introduced by Leonhard Euler, is used to represent the orientation of a rigid body in three-dimensional space relative to a fixed coordinate system. The three angles are often denoted by the symbols φ (for roll angle), θ (for pitch angle), and ψ (for yaw angle), for the x, y, and z axes, respectively. The order in

which these rotations are performed can vary, depending on the convention used, with the most common ones being the "XYZ" or the "ZYX" conventions.

For the ZYX Euler angles (Figure **4**) the sequence in which the rotation matrices are multiplied follows a rotation about the z axis, followed by the y axis, followed by the x axis, with every rotation occurring with respect to the body-fixed reference frame. The transformation matrix from the body reference system to the inertial system, using the aforementioned rotational conventions, can be described by the rotation matrix $\boldsymbol{R}_{zyx}(\varphi,\theta,\psi)$:

$$\boldsymbol{R}_{zyx}(\varphi,\theta,\psi) = \boldsymbol{R}_z(\psi)\cdot\boldsymbol{R}_y(\theta)\cdot\boldsymbol{R}_x(\varphi) =$$

$$\begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}\cdot\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} =$$

$$= \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \qquad (1.1)$$

where $c_\phi = cos(\phi)$, $s_\phi = \sin(\phi)$, $c_\theta = \cos(\theta)$, $s_\theta = \sin(\theta)$, $c_\psi = \cos(\psi)$ *and* $s_\psi = \sin(\psi)$ and the angles should be in the range of $\varphi,\psi \in [-\pi,\pi]$ and $\theta \in [-\frac{\pi}{2},\frac{\pi}{2}]$.

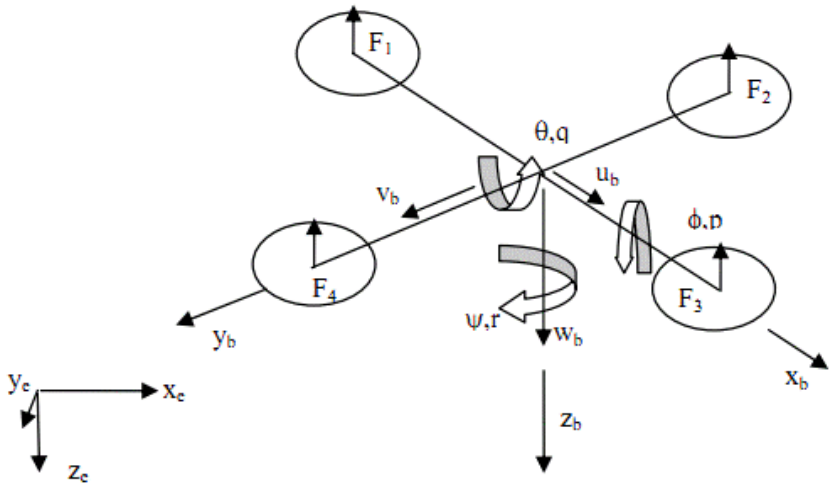


Figure 4. The Euler angles for the quadrotor

Referring to the vectors $[x\ y\ z\ \varphi\ \theta\ \psi]^T$ and $[u\ v\ w\ p\ q\ r]^T$, which contain the linear and angular position of the quadrotor with respect to the earth frame and the linear and angular velocities with respect to the body frame, respectively, from the kinematics analysis, it is concluded that:

$$\boldsymbol{v} = \boldsymbol{R_{zyx}} \cdot \boldsymbol{v_B} \tag{1.2}$$

$$\boldsymbol{\omega} = \boldsymbol{T} \cdot \boldsymbol{\omega_B} \tag{1.3}$$

where $\boldsymbol{v} = [\dot{x}\ \dot{y}\ \dot{z}]^T \in \mathbb{R}^3$, $\boldsymbol{\omega} = \left[\dot{\varphi}\ \dot{\theta}\ \dot{\psi}\right]^T \in \mathbb{R}^3$, $\boldsymbol{v_B} = [u\ v\ w]^T \in \mathbb{R}^3$, $\boldsymbol{\omega_B} = [p\ q\ r]^T \in \mathbb{R}^3$ and T is a matrix for angular transformations:

$$\boldsymbol{T} = \begin{pmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{pmatrix} \tag{1.4}$$

where $t_\theta = \tan(\theta)$. So, the kinematic model of the quadrotor can be written as followed:

$$\begin{gathered} \dot{x} = w\left[s_\phi s_\psi + c_\phi c_\psi s_\theta\right] - v\left[c_\phi s_\psi - c_\psi s_\phi s_\theta\right] + u\left[c_\psi c_\theta\right] \\ \dot{y} = v\left[c_\phi c_\psi + s_\phi s_\psi s_\theta\right] - w\left[c_\psi s_\phi - c_\phi s_\psi s_\theta\right] + u\left[c_\theta s_\psi\right] \\ \dot{z} = w\left[c_\phi c_\theta\right] - u[s_\theta] + v\left[c_\theta s_\phi\right] \\ \dot{\varphi} = p + r\left[c_\phi t_\theta\right] + q[s_\phi t_\theta] \\ \dot{\theta} = q\left[c_\phi\right] - r[s_\phi] \\ \dot{\psi} = r\frac{c_\phi}{c_\theta} + q\frac{s_\phi}{c_\theta} \end{gathered} \tag{1.5}$$

From Newton's law about the total force acting on the quadrotor, the dynamic model in the body frame can be expressed as:

$$
\begin{aligned}
f_x &= m(\dot{u} + qw - rv) \\
f_y &= m(\dot{v} - pw + ru) \\
f_z &= m(\dot{w} + pv - qu) \\
m_x &= \dot{p}I_x - qrI_y + qrI_z \\
m_y &= \dot{q}I_y + prI_x - prI_z \\
m_z &= \dot{r}I_z - pqI_x + pqI_y
\end{aligned}
\tag{1.6}
$$

with $\boldsymbol{m_B} = [m_x\ m_y\ m_z]^T \in \mathbb{R}^3$ denoting the torques along the three axes, $\boldsymbol{I}$ the diagonal inertia matrix and $\boldsymbol{m}$ the mass of the quadrotor. Bearing in mind that the external forces and the external moments in the body frame are given by:

$$\boldsymbol{f_B} = mg\boldsymbol{R}^T \cdot \hat{\boldsymbol{e}}_{\boldsymbol{z}} - f_t\hat{\boldsymbol{e}}_{\boldsymbol{3}} + \boldsymbol{f_w} \tag{1.7}$$

$$\boldsymbol{m_B} = \boldsymbol{\tau_B} - \boldsymbol{g_a} + \boldsymbol{\tau_w} \tag{1.8}$$

respectively, with $\boldsymbol{f_w}$ representing the forces produced by wind, $\hat{\boldsymbol{e}}_{\boldsymbol{z}}$ the unit vector in the z axis, $f_t\hat{\boldsymbol{e}}_{\boldsymbol{3}}$ the product of the total thrust generated and the unit vector in the body z axis, $\boldsymbol{g_a}$ the gyroscopic moments because of the combined rotation of the rotors, $\boldsymbol{\tau_B}$ the control torques as a result of differences in the rotor speeds and $\boldsymbol{\tau_w}$ the torques produced by winds. So, the dynamic model of the quadrotor in the body frame can be expressed as:

$$
\begin{aligned}
-mgs_\theta + f_{wx} &= m(\dot{u} + qw - rv) \\
mg\left[c_\theta s_\phi\right] + f_{wy} &= m(\dot{v} - pw + ru) \\
mg\left[c_\theta c_\phi\right] + f_{wz} - f_t &= m(\dot{w} + pv - qu) \\
\tau_x + \tau_{wx} &= \dot{p}I_x - qrI_y + qrI_z \\
\tau_y + \tau_{wy} &= \dot{q}I_y + prI_x - prI_z \\
\tau_z + \tau_{wz} &= \dot{r}I_z - pqI_x + pqI_y
\end{aligned}
\tag{1.9}
$$

while making the fair assumption that the inertia of each rotor is small.

By organizing the state's vector as a combination of the three Euler angles $[\phi \ \theta \ \psi]^T$, the three rotational velocities $[p \ q \ r]^T$, the three translational velocities $[u \ v \ w]^T$ and the three positions along x, y, and z axis $[x \ y \ z]^T$ as follows:

$$\boldsymbol{x} = [\varphi \ \theta \ \psi \ p \ q \ r \ u \ v \ w \ x \ y \ z]^T \in \mathbb{R}^3 \tag{1.10}$$

the equations of the dynamic behavior of the quadrotor can be written in state-space form [2] as follows:

$$\dot{\varphi} = p + r\left[c_\phi t_\theta\right] + q\left[s_\phi t_\theta\right]$$

$$\dot{\theta} = q\left[c_\phi\right] - r\left[s_\phi\right]$$

$$\dot{\psi} = r\frac{c_\phi}{c_\theta} + q\frac{s_\phi}{c_\theta}$$

$$\dot{p} = \frac{I_y - I_z}{I_x}rq + \frac{\tau_x + \tau_{wx}}{I_x}$$

$$\dot{q} = \frac{I_z - I_x}{I_y}pr + \frac{\tau_y + \tau_{wy}}{I_y}$$

$$\dot{r} = \frac{I_x - I_y}{I_z}pq + \frac{\tau_z + \tau_{wz}}{I_z}$$

$$\dot{u} = rv - qw - g[s_\theta] + \frac{f_{wx}}{m}$$

$$\dot{v} = pw - ru - g\left[s_\phi c_\theta\right] + \frac{f_{wy}}{m}$$

$$\dot{w} = qu - pv + g\left[c_\theta c_\phi\right] + \frac{f_{wz} - f_t}{m}$$

$$\dot{x} = w\left[s_\phi s_\psi + c_\phi c_\psi s_\theta\right] - v\left[c_\phi s_\psi - c_\psi s_\phi s_\theta\right] + u\left[c_\psi c_\theta\right]$$

$$\dot{y} = v\left[c_\phi c_\psi + s_\phi s_\psi s_\theta\right] - w\left[c_\psi s_\phi - c_\phi s_\psi s_\theta\right] + u\left[c_\theta s_\psi\right]$$

$$\dot{z} = w\left[c_\phi c_\theta\right] - u[s_\theta] + v\left[c_\theta s_\phi\right]$$

(1.11)

The (1.11) is a set of twelve overly complex and non-linear first-order differential equations with intertwined terms. The roll, pitch and yaw rates are represented by the first three equations that consider the rotational velocities and they use trigonometric functions to relate these rates of change of the angles to each other. The next three equations describe the proportional relation between the rotational accelerations with the moments of inertia and the torques acting on the quadrotor. The following three equations describe the quadrotor's translational dynamics, while considering the forces acting on it, including gravity, wind forces and the thrust force generated by the rotors. The final three equations describe the position rates and their relation to the velocities and angles of the quadrotor.

Nevertheless, instead of describing the wind disturbances through the terms τ_w and f_w for the torques and forces that are created due to the wind, it is preferable to use the friction coefficients. These are denoted as k_p, k_q, k_r, k_u, k_v and k_w for the rotational and translational accelerations in the x, y, and z axes, respectively [2]. So, they can be incorporated into the (1.11) set of equations, affecting only the $\dot{p}, \dot{q}, \dot{r}, \dot{u}, \dot{v}, \dot{w}$ formulas as follows:

$$\dot{p} = \frac{I_y - I_z}{I_x} rq + \frac{\tau_x + \tau_{wx}}{I_x} - k_p p$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_y + \tau_{wy}}{I_y} - k_q q$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{\tau_z + \tau_{wz}}{I_z} - k_r r$$

$$\dot{u} = rv - qw - g[s_\theta] + \frac{f_{wx}}{m} - k_u u$$

$$\dot{v} = pw - ru - g\left[s_\phi c_\theta\right] + \frac{f_{wy}}{m} - k_v v$$

$$\dot{w} = qu - pv + g\left[c_\theta c_\phi\right] + \frac{f_{wz} - f_t}{m} - k_w w$$

(1.12)

However, the equations presented so far consider neither the gyroscopic effect nor the motors' angular velocities. That being said, it is possible to relate the forces and torques acting on the system with the angular velocities $\Omega_1, \Omega_2, \Omega_3, \Omega_4$ through the following equations [2]:

$$f_t = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$
$$\tau_x = bl(\Omega_3^2 - \Omega_1^2)$$
$$\tau_y = bl(\Omega_4^2 - \Omega_2^2)$$
$$\tau_z = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \quad (1.13)$$

where l is the distance between the rotors and the center of the quadrotor, b the thrust coefficient, and d the drag coefficient. The thrust coefficient is a parameter used to model the relationship between the thrust generated by the motor and the square of its rotational speed. It depends on many factors, including the motor's design and propellers, as well as the occurring environmental conditions. On the contrary, the aerodynamic drag force is a resistive air force that opposes to the motion of the quadrotor, so the drag coefficient represents the proportionality between the drag force and the square of the velocity of the quadrotor. It is an essential part to consider when designing and controlling the quadrotor, as it highly influences its stability, maneuverability, and energy efficiency. From the equations (1.13) it is possible to derive the angular velocities through the matrix form:

$$\begin{bmatrix} f_t \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ -bl & 0 & bl & 0 \\ 0 & -bl & 0 & bl \\ -d & d & -d & d \end{bmatrix} \cdot \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \rightarrow Q = A \cdot \Omega \rightarrow \Omega = A^{-1} \cdot Q \quad (1.14)$$

so, the angular velocities are:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \sqrt{\Omega} \quad (1.15)$$

Consequently, bearing in mind that the overall motor's rotation for the quadrotor system is given by:

$$\omega_r = \omega_1 - \omega_2 + \omega_3 - \omega_4 \quad (1.16)$$

it is possible to rewrite the equations related to $\dot{p}$ and $\dot{q}$, by adding in combination both the actuator dynamics and the gyroscopic effect as follows, where J_r is the gyroscopic coefficient:

$$\dot{p} = \frac{I_y - I_z}{I_x} rq + \frac{\tau_x + \tau_{wx}}{I_x} - k_p p - J_r p \omega_r$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_y + \tau_{wy}}{I_y} - k_q q + J_r q \omega_r \tag{1.17}$$

Thus, the complete mathematical model in the state-space form, is given by the following set of equations (1.18), with ω_r deriving from a combination of equations (1.13), (1.14) and (1.15):

$$\dot{\varphi} = p + r\left[c_\phi t_\theta\right] + q\left[s_\phi t_\theta\right]$$

$$\dot{\theta} = q\left[c_\phi\right] - r\left[s_\phi\right]$$

$$\dot{\psi} = r\frac{c_\phi}{c_\theta} + q\frac{s_\phi}{c_\theta}$$

$$\dot{p} = \frac{I_y - I_z}{I_x} rq + \frac{\tau_x + \tau_{wx}}{I_x} - k_p p - J_r p \omega_r$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_y + \tau_{wy}}{I_y} - k_q q + J_r q \omega_r$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{\tau_z + \tau_{wz}}{I_z} - k_r r$$

$$\dot{u} = rv - qw - g[s_\theta] + \frac{f_{wx}}{m} - k_u u$$

$$\dot{v} = pw - ru - g\left[s_\phi c_\theta\right] + \frac{f_{wy}}{m} - k_v v$$

$$\dot{w} = qu - pv + g\left[c_\theta c_\phi\right] + \frac{f_{wz} - f_t}{m} - k_w w$$

$$\dot{x} = w\left[s_\phi s_\psi + c_\phi c_\psi s_\theta\right] - v\left[c_\phi s_\psi - c_\psi s_\phi s_\theta\right] + u\left[c_\psi c_\theta\right]$$

$$\dot{y} = v\left[c_\phi c_\psi + s_\phi s_\psi s_\theta\right] - w\left[c_\psi s_\phi - c_\phi s_\psi s_\theta\right] + u\left[c_\theta s_\psi\right]$$

$$\dot{z} = w\left[c_\phi c_\theta\right] - u[s_\theta] + v\left[c_\theta s_\phi\right] \tag{1.18}$$

The (1.18) set of non-linear first-order differential equations represents the mathematical model used in the presented diploma thesis. Having considered the underlying physics and the sophisticated mathematical framework regarding the system's geometry and the possible numerous disturbances, it is anticipated that this model will effectively demonstrate the complex dynamic behavior of the quadrotor.

1.5 Open-loop Simulation

Although the open-loop simulation is not an exact representation of the real world and usually includes limitations or inaccuracies, it is considered a valuable tool for the control systems area. The reason behind that is that it offers an insight into how the system responds to different inputs and how its behavior is affected by various parameters.

In the context of the quadrotor, control refers to the ability to manipulate the thrusts and torques applied to the vehicle, to achieve the desired flight behavior. The thrust is related to the vertical movement while the torques produce rotations, which help the quadrotor move in the x or y direction.

The inputs to the system are the total thrust and the torques in the x, y, and z axes, known as the control vector $\boldsymbol{u} = [f_t \ \tau_x \ \tau_y \ \tau_z]^T \in \mathbb{R}^4$, while the output is the state-space vector $\boldsymbol{x} = [\varphi \ \theta \ \psi \ p \ q \ r \ u \ v \ w \ x \ y \ z]^T \in \mathbb{R}^{12}$ with twelve variables that describe the exact condition of the quadrotor in the 3D environment. By choosing specific values for the control vector u and the parameters, it is possible to create an open-loop graph that depicts the free behavior of the quadrotor in the form of the twelve system states.

It is noted that for the open-loop simulation, and for the following simulations presented in Chapter 2., the ode3 (Bogacki-Shampine) solver will be employed; it is a fixed-step solver that computes the state of the model as an explicit function of the current value of the state and the derivatives using the Bogacki-Shampine formula integration technique. The characteristics of the quadrotor simulated are presented in Table **1** and the state-space variables examined are presented in Table **2** as follows:

Table 1. Constants for the simulation of the quadrotor

Symbol	Description	Arithmetic value	Units
m	Mass of the quadrotor	1	kg
l	Arm length of the quadrotor	0.24	m
g	Gravitational acceleration	9.81	m/s^2
J_r	Gyroscopic moment	$1.08 \cdot 10^{-6}$	$kg \cdot m^2$
I_x	Moment of inertia in roll	$8 \cdot 10^{-3}$	$kg \cdot m^2$
I_y	Moment of inertia in pitch	$8 \cdot 10^{-3}$	$kg \cdot m^2$
I_z	Moment of inertia in yaw	$14.2 \cdot 10^{-3}$	$kg \cdot m^2$
b	Thrust constant	$54.2 \cdot 10^{-6}$	unitless
d	Drag constant	$1.1 \cdot 10^{-6}$	unitless
k_p	Friction coefficient for translational movement in the x axis	0.03	unitless
k_q	Friction coefficient for translational movement in the y axis	0.03	unitless
k_r	Friction coefficient for translational movement in the z axis	0.01	unitless
k_u	Friction coefficient for rotational movement in the x axis	0.048	unitless
k_v	Friction coefficient for rotational movement in the y axis	0.11	unitless
k_w	Friction coefficient for rotational movement in the z axis	0.046	unitless

Table 2. The state-space variables of the quadrotor system

Symbol	Description	Units
φ	Roll angle	rad
θ	Pitch angle	rad
ψ	Yaw angle	rad
p	Rotational velocity in the x axis	rad/sec
q	Rotational velocity in the y axis	rad/sec
r	Rotational velocity in the z axis	rad/sec
u	Translational velocity in the x axis	m/sec
v	Translational velocity in the y axis	m/sec
w	Translational velocity in the z axis	m/sec
x	Position x	m
y	Position y	m
z	Position z	m

For the open-loop simulation for the quadrotor system, assuming an input vector of zero and that all of the state parameters have an initial value of zero, it is anticipated that, even in the absence of control inputs, the quadrotor will present changes due to the vehicle's inherent characteristics and external forces acting on it.

More specifically, based on the Equations 1.18, it is clear that the quadrotor system is subject to three main forces: thrust force (produced by the propellers), wind forces and gravity. That being said, for a control vector equal to zero, meaning that there is no control input applied to the system, the quadrotor is not actively controlled or stabilized, so its motion is governed by the balance of the forces acting on it.

Considering the NED reference system, the gravity force acting on the vehicle is positive as it coincides with the barycenter of the earth, causing it to move downwards. Therefore, in order for the quadrotor to move upwards, the thrust force needs to be applied in the negative direction, but it will cause the quadrotor to move this way only if that force, also known as lift force, is greater than the force of gravity. Additionally, any external forces acting on the quadrotor, such as wind forces, could affect the amount and direction of the thrust generated by the rotors, and, as a result, the quadrotor's altitude in the z-axis and the corresponding translational velocity.

On the other hand, in order for the quadrotor to rotate in x, y or z axes (roll, pitch, yaw motion, respectively) there should be torques acting on the system, which is not the case for a control input as the one employed in this simulation. The open – loop simulation graph for a simulation time of 10 seconds is presented in Figure **5**.

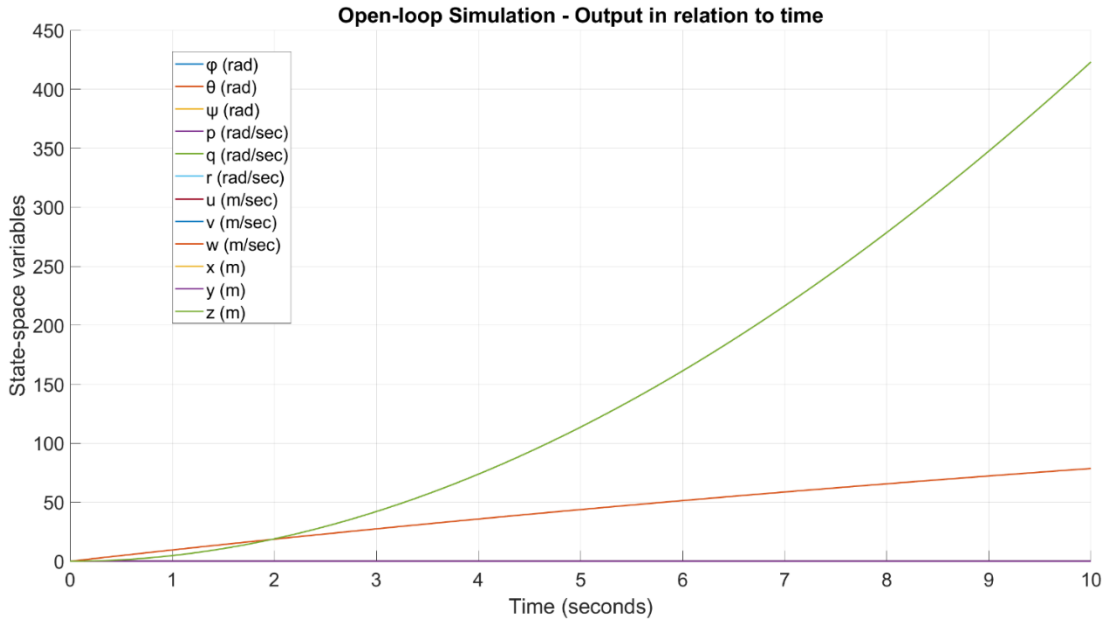


Figure 5. Open-loop simulation for zero input and zero initial values for the quadrotor system

Given the specified input, it is evident that solely the altitude, z, and the translational velocity in the z axis, w, were affected, as was anticipated, and that the quadcopter moves “downwards”, since there is not ground in the simulation, driven by the force of gravity.

Although the graph does not show all the details about how the quadrotor behaves, it confirms that the quadrotor is an overly sensitive system, with an extremely quick reaction to changes, while also highlighting the fact that some of its characteristics are inherent to its dynamic nature.

The information obtained from the open-loop simulation will be used in the upcoming chapters, to develop control strategies that are both robust and agile. By considering these factors in the control design process, it is expected that the resulting control system will perform with improved stability, accuracy, and responsiveness.

CHAPTER 2: PID Control of the Quadrotor

In the second chapter, a thorough examination of a single quadrotor control system employing PID controllers is presented. The chapter begins with an examination of the basic principles of the PID controller, including its mathematical formulation. Subsequently, the relevant control system's design is provided. Finally, the system's performance is demonstrated through simulation tests for both step inputs and trajectory following, for a comprehensive evaluation over the suitability and the challenges of such approach for the control of the quadrotor.

2.1 The PID Controller

A PID (Proportional-Integral-Derivative) controller is a widely used feedback control mechanism first introduced by Elmer Sperry in 1911, but it was not until the 1930s that it became popular [20]. Its acronym is indicative of the three primary control actions that it applies to the system, as they work together to keep a stable and accurate control of a physical process.

More specifically, the Proportional (P) part of a PID controller is the first and simplest control action as it depends solely on the difference between the desired set point and the value of the controlled variable, known as the error term. Its goal is to reduce the error by producing an output that is directly proportional to the size of it; the larger the error, the larger the control signal, and the faster the system's response to it. The coefficient that scales the output is called the Proportional gain or Gain factor (K_p) and the output signal is given by:

$$P_{output} = K_p \cdot error = K_p \cdot e(t) \tag{2.1}$$

However, using proportional control is not sufficient to successfully control a complex system, since it produces overshoots and/or oscillations and it does not eliminate the steady-state error.

The Integral (I) part of the PID controller is the one responsible for eliminating the steady-state error, since it continuously sums up the error over time, integrates it and generates an output that is proportional to the accumulated error; the larger the error and the longer it persists; the larger the output signal produced by the I term and the faster the controller's response to eliminate it. As the error decreases, the proportional part diminishes while the integral part intensifies until the error is zero. The Integral gain (K_i) determines the amount of response to the accumulated error with the output signal given by:

$$I_{output} = K_I \cdot Integral(error) = K_I \cdot \int_0^t e(t)dt \quad (2.2)$$

Nonetheless, the I term often presents challenges as it creates the possibility of *integral windup*, a phenomenon that occurs when the accumulated error exceeds the limits of the controller's range causing the integral term to continue accumulating the error and producing an increasing control signal even though the output of the controller has reached its saturation limits. This relates to the fact that the I term increases the number of poles at zero, which can result in overshoots, oscillations, or instability, especially for nonlinear complex systems.

Finally, the Derivative (D) part of the PID produces a response proportional to the process variable's rate of change. So, by dampening the response of the controller to sudden changes, it improves the transient response (i.e., smaller overshoot) and produces fewer oscillations. Using the Derivative gain (K_D), the formula for the output signal is given by:

$$D_{output} = K_D \cdot \frac{d(error)}{dt} = K_D \cdot \frac{de(t)}{dt} \quad (2.3)$$

Yet, the D term also presents disadvantages with the main one being the possibility of the phenomenon called *derivative kick*, which occurs when the reference signal is non-differentiable. One possible approach to combatting this phenomenon is to design a specialized PID controller circuit, that will be explained in detail in sub-chapter 2.3.1. Moreover, in practical control systems, the use of the derivative term can be problematic due to its sensitivity to noise, as it amplifies the high-frequency components of the error signal. Therefore, it should be approached with caution and careful tuning to avoid instability or deficient performance, especially when there is significant noise in measurements. However, there are methods to deal with the noise amplification problem, such as adding a lowpass filter in series with the derivative gain.

Figure **6** presents the typical structure of a PID controller, used to control the response of a process.

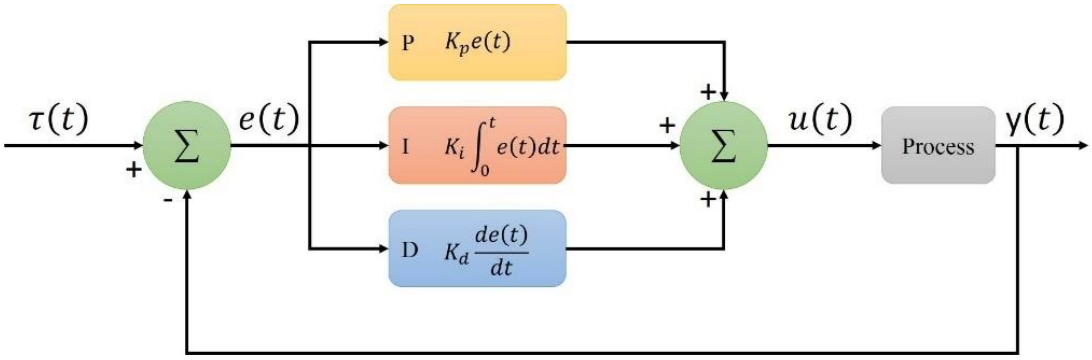


Figure 6. The Block Diagram of a PID controller

As illustrated in Figure **6**, the PID controller takes as input the error signal $e(t)$, which is the difference between the reference signal $\tau(t)$ and the output signal of the process, $y(t)$. This error signal is then used to calculate the control signal $u(t)$ that is applied to the system. This is a combination of the three terms of the PID controller (the Proportional, the Integral, and the Derivative part) summed together and weighted by their respective gain constants:

$$u(t) = K_p \cdot e(t) + K_I \cdot \int_0^t e(t)dt + K_D \cdot \frac{de(t)}{dt} \tag{2.4}$$

In consequence, the process of choosing the appropriate K_p, K_i and K_d gains, called *tuning*, is crucial for achieving a satisfactory performance and stability of the closed-loop control scheme. Even though there are several methods for PID tuning, like the Ziegler-Nichols or the Cohen-Coon method, it is a process extremely dependable on trial-and-error, especially since the objectives of the three parts are often contradictory. Thus, it is important to bear in mind the specific system that the PID controller is applied on, as well as the aims and the prioritized goals set. [20]

2.2 Design of the PID Control System

As far as the quadrotor is concerned, it is a sensitive and dynamic system, so the control system designed for it must be capable of responding rapidly to changing flight conditions, while also ensuring its stability and safety. It is also crucial to highlight the fact that it is an underactuated nonlinear complex mechanical system since it has six degrees of freedom (DOF) but only four inputs, which complicates the design even further.

For such underactuated system as the one presented, for all six degrees of freedom to be controlled, it is necessary to find a way to use the available control inputs to achieve the desired movements. One common approach is the use of a cascaded formulation where the output of one controller is used as the setpoint for the next controller in the hierarchy, a process known as *Cascaded Control Loops*. [21]

In a cascaded PID control system, there are multiple PID controllers working in series in a way that the output of one PID controller is fed into the input of the PID controller next to it. In a typical system of that kind, the internal controller is responsible for controlling the variable, through which the overall process variable is controlled. More specifically, as presented in Figure **7**, the process target is given as reference input to the outer PID controller which, in turn creates the inner target and feeds it to the inner loop controller. The latter produces the control action applied to the process. The design is completed with a feedback control mechanism for both the outer and the inner loop controllers.

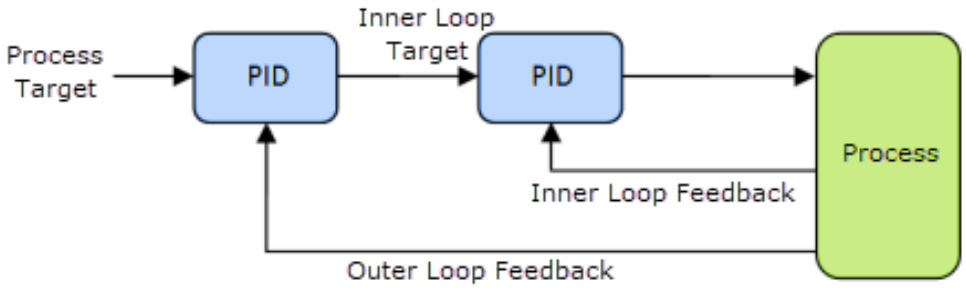


Figure 7. The Block Diagram of a cascaded PID system

In the case of the quadrotor, there is the need for at least two targets to be controlled through others, so it is possible to use four PID controllers for instantaneous regulation; one for the altitude z which produces the f_t, one for the roll angle φ, one for the pitch angle θ and one for the yaw angle ψ. While bearing in mind that the quadrotor's lateral motion in the x and y axes is affected by the lateral thrusts as well as the angles φ and θ, it is possible to use the cascade control strategy, as follows:

- For the x-direction control, the outer loop PID controller uses the x position reference as input and produces a θ angle output, which is fed as reference input to the inner loop PID controller for the pitch angle regulation. Then, this controller produces the torque τ_y as output.
- For the y-direction control, the outer loop PID controller uses the y position reference as input and produces a φ angle output, which is fed as reference input to the inner loop PID controller for the roll angle regulation. Then, this controller produces the torque τ_x as output.

With that control design, as depicted in Figure **8**, the quadrotor's system can be controlled effectively and with adequate precision, despite the limitations of the available control inputs.

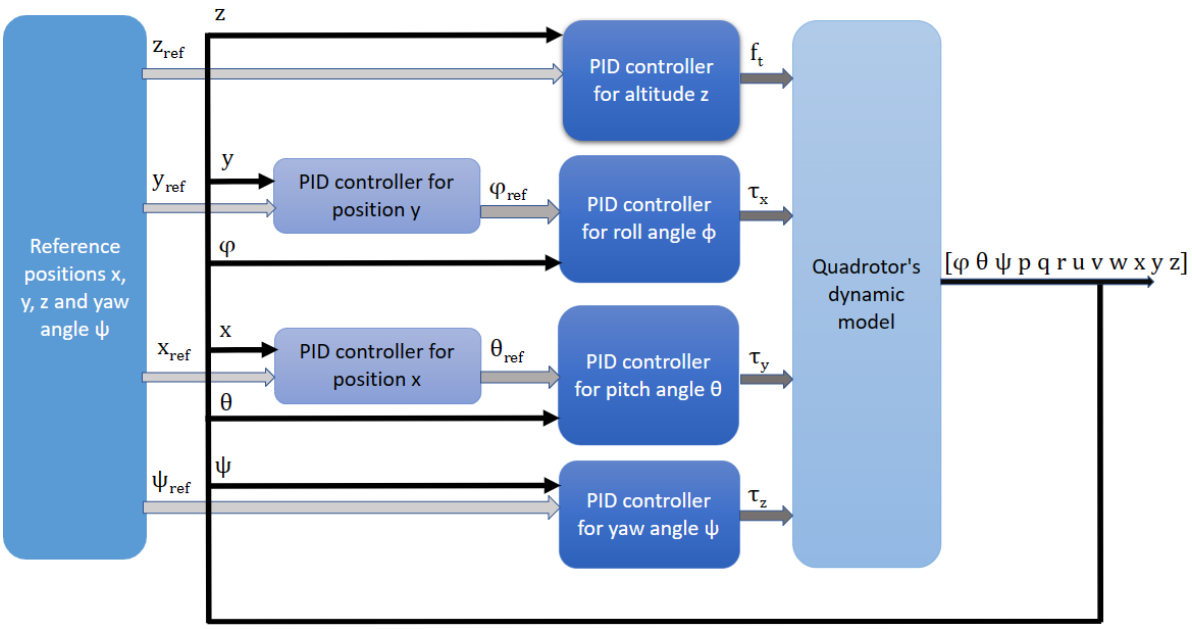


Figure 8. The designed cascaded PID control system for the quadrotor

More specifically, the inputs to the control system are the position references x, y, and z, as well as the reference for the yaw angle ψ, which usually equals to zero, while the outputs of the system are the twelve variables that utterly describe the quadrotor's behavior in 3D space. These references are fed as inputs to the PID controllers: the one for altitude z, the one for the yaw angle and the outer-loop controllers for position x and y. The first two instantly produce the variables of the control vector f_t and τ_z, respectively, while the latter two feed their outputs into the inner loop controllers for roll and pitch angle, eventually producing the torques τ_y and τ_x.

In that way, the quadrotor's plant, which follows the mathematical model explained at Chapter 1., receives the control vector $\boldsymbol{u} = [f_t \ \tau_x \ \tau_y \ \tau_z]^T \in \mathbb{R}^4$ and produces as output the state-space vector $\boldsymbol{x} = [\varphi \ \theta \ \psi \ p \ q \ r \ u \ v \ w \ x \ y \ z]^T \in \mathbb{R}^{12}$. The state-space variables are given as feedback to the respective controllers to enable closed-loop control of the system and thus being able to adjust the input to achieve the desired output. This feedback control mechanism, apart from being part of the PID control, is an essential aspect of the quadrotor control, as it provides the conditions for the vehicle to adapt to changes or disturbances in the environment.

2.3 Tuning the PID controllers

The process of tuning the PID controllers, while following the cascaded PID design explained, required a thorough understanding of the system dynamics. More specifically, when employing PID controllers, the signal given as input is the error signal which corresponds to the difference between the desired-reference signal and the current-actual signal, as follows:

$$error\ signal = reference\ signal - actual\ signal = r(t) - y(t) \tag{2.5}$$

When working with the NED coordinate system, the positive z axis, which in the quadrotor's case is relevant to the altitude, is defined pointing downwards. That said, increasing the altitude corresponds to a negative change in z and, as a result, a negative thrust command should be applied. During the tuning process, it is important to also remember that the PID controller for the altitude has an extra term added to its output, the gravity compensation. Another particularity of the NED system which affects the tuning of the PIDs is that a negative pitch angle drives the system to the positive x direction; these are all important notes taken into consideration during the tuning and the simulation part of the system's design.

Moreover, to achieve the desired performance, careful coordination between the inner and outer loop controllers is crucial. More specifically, a good technique turned out to be tuning the inner loop PIDs first and then fine tuning the outer loop PIDs, while taking into account the interdependence with the inner ones. Furthermore, a step-by-step approach was utilized; adjusting the proportional gains first and, upon achieving a satisfactory system response, the derivative gain was employed to enhance the system's stability. Finally, a small integral gain was introduced to eliminate the steady-state error, while ensuring that the response was not negatively affected.

The objective in mind was to produce responses that are as similar as possible to first-order responses (i.e., zero overshoot and zero oscillations), while also maintaining a quick enough response for such a dynamic and sensitive system as that of the quadrotor.

2.4 Simulation analysis and results

The design of a system can often be affected by the type of input used as a reference signal, and this is also true when employing PID controllers. To investigate such an impact, two cases will be presented in the following sub-chapters. The first case will involve simulating the control system with a step input to examine how the controller responds to a sudden change in the reference signal. The

second case will involve simulating the control system for trajectory tracking, which is a more complex scenario that involves following a specific path or trajectory. Through the examination of these two cases, valuable insights can be gained into the behavior and performance of PID controllers under different input signal conditions for the quadrotor system.

Before presenting the simulation results, it is important to highlight the fact that, in many cases, the cascaded PID system is designed in a way that the inner loop PIDs have a faster response (i.e., smaller sampling time) than the outer loops. This ensures that the inner loops can quickly respond to errors in the system and/or correct them, providing greater overall accuracy. However, in the simulations presented in this thesis, the fundamental sample time is constant, equal to $T_s = 0.02$ seconds and the same for all PID controllers, for a more general representation.

2.4..1 Simulation for step input

When applying a step input, one of the challenges faced is that it is not a continuous function; this creates the phenomenon previously referred to as *derivative kick.* As a result, calculating the signal's derivative becomes a perplexing task, which poses a significant problem when the control system design includes PID controllers. Without the derivative part, the PID controller cannot be finely tuned and thus the overall performance is negatively impacted.

One solution to this problem is to use the state variables as feedback in the derivative path, rather than the error signal. In that way, the controller responds to changes in the state variable instead of changes in the error signal, enabling the simulation of the system despite the discontinuity of the step input. In order to implement this solution, the derivative term of the PID controller should be subtracted from the controller's output, instead of being added.

In more detail, as defined in Equation 2.5, the error signal is expressed as $e_0 = r(t) - y(t)$, where $r(t)$ is the reference signal at time t and $y(t)$ is the output signal at time t. Calculating the derivative of the error signal will give $\frac{de(t)}{dt} = \frac{dr(t)}{dt} - \frac{dy(t)}{dt}$, and for a step input fired at time t_1 as reference, the derivative of the error signal will be:

- undefined for t_1, as the derivative of a discontinuous function goes to infinity.
- $\frac{de(t)}{dt} = 0 - \frac{dy(t)}{dt} = -\frac{dy(t)}{dt}$ for $t > t_1$

This method provides a simple and practical solution for dealing with step inputs, which ensures that the controller can effectively regulate the system, even in the presence of discontinuous

inputs. As a result, the system's performance can be improved, leading to better control of the process being regulated.

Below, in Figure **9**, the diagram of such PID design is presented for the controllers corresponding to the angles and the positions x and y, whereas the PID design for the altitude z is depicted in Figure **10**, including the gravitational compensation term. In Table **3**, the gains k_P, k_I and k_D for all the PID controllers used in the simulation design are presented, after the process of tuning.

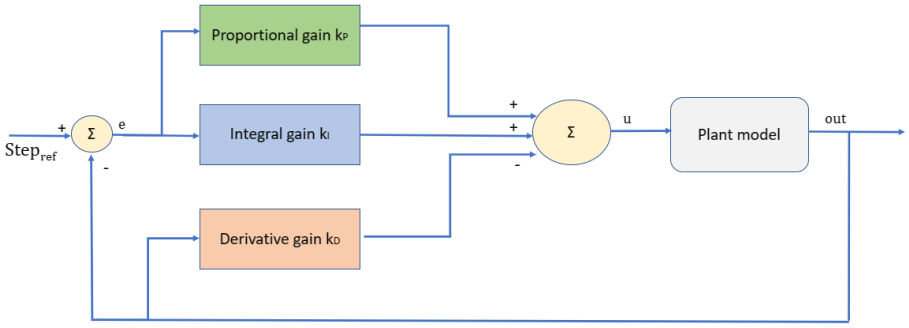


Figure 9. The general PID block diagram for step input

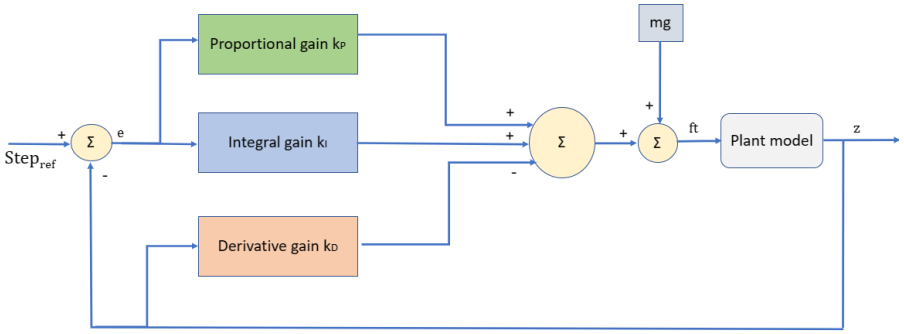


Figure 10. The PID block diagram for the altitude z, when employing a step reference input

Table 3. The PID parameters for step reference input – PID design control

	Proportional gain k_P	Integral gain k_I	Derivative gain k_D
PID controller for position x	-0.25	-0.0001	-0.3
PID controller for position y	0.15	0.0001	0.25
PID controller for altitude z	-1.5	-0.001	-2.55
PID controller for roll angle φ	0.9	0.01	0.1
PID controller for pitch angle θ	0.6	0.001	0.1
PID controller for yaw angle ψ	1.5	0.001	0.65

Assuming the step reference inputs $z_{ref} = -1\ [m]$, $x_{ref} = 1\ [m]$, $y_{ref} = 1\ [m]$ and $\psi_{ref} = 0.1\ [rad]$, the graphs regarding the positions x, y, and z (Figure 11Figure 11, Figure **12** and Figure **13**), as well as the outputs of the PID controllers, φ and θ (Figure **14** and Figure 15), the angle psi (Figure **16**), the thrust (Figure 17) and the torques produced (Figure **18**, Figure 19 and Figure **20**), are presented below:

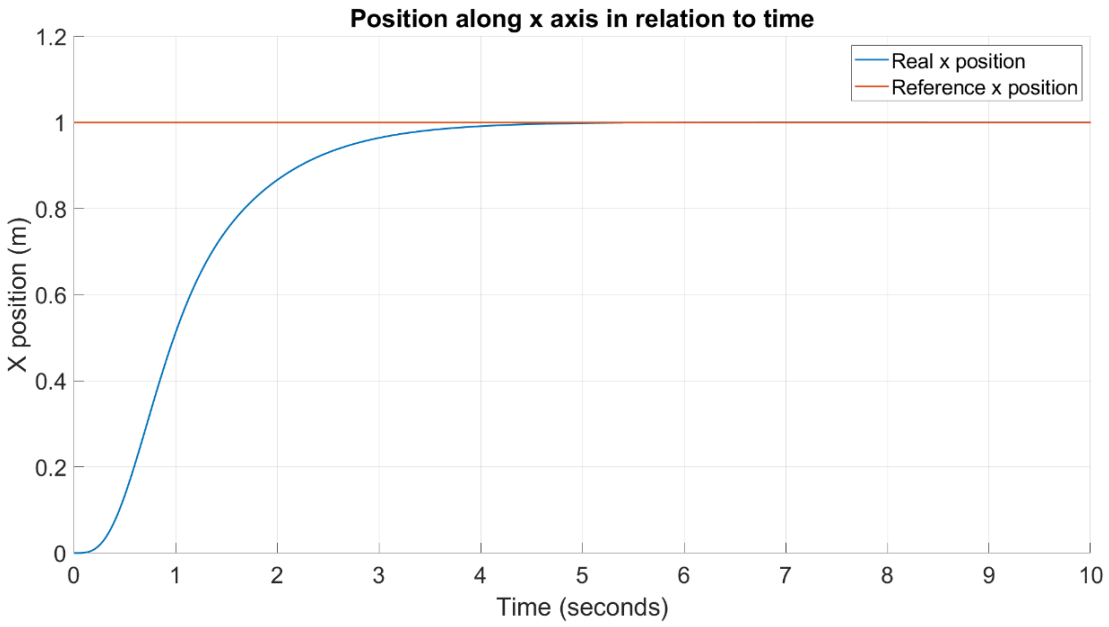


Figure 11. Position x in relation to time for step reference input equal to 1 [m] using the PID design

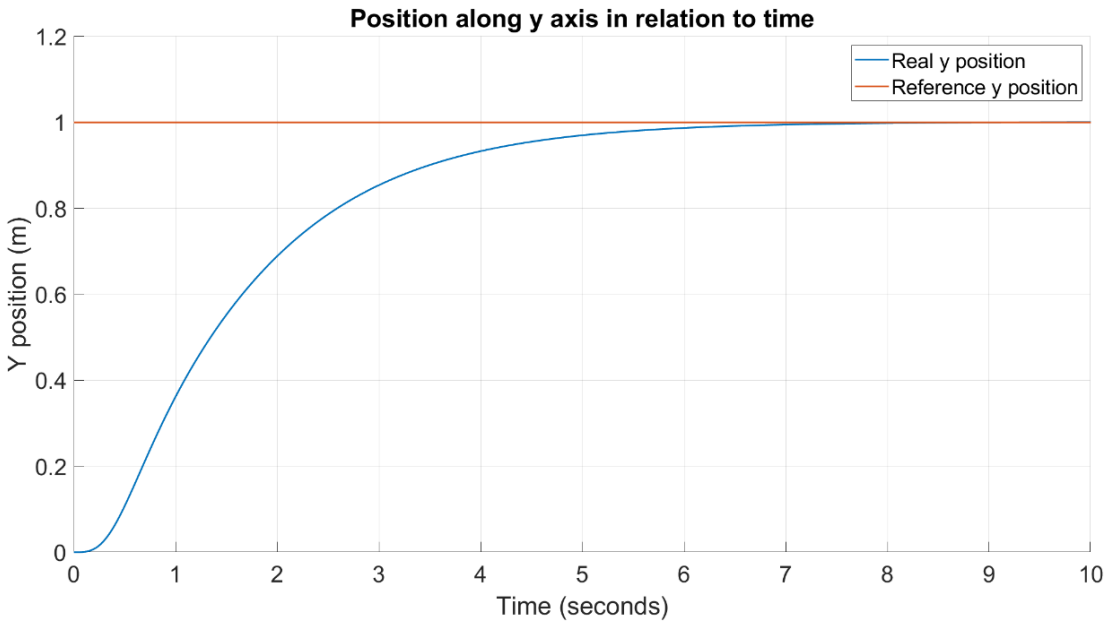


Figure 12. Position y in relation to time for step reference input equal to 1 [m] using the PID design

Figure 13. Position z in relation to time for step reference input equal to -1[m] using the PID design

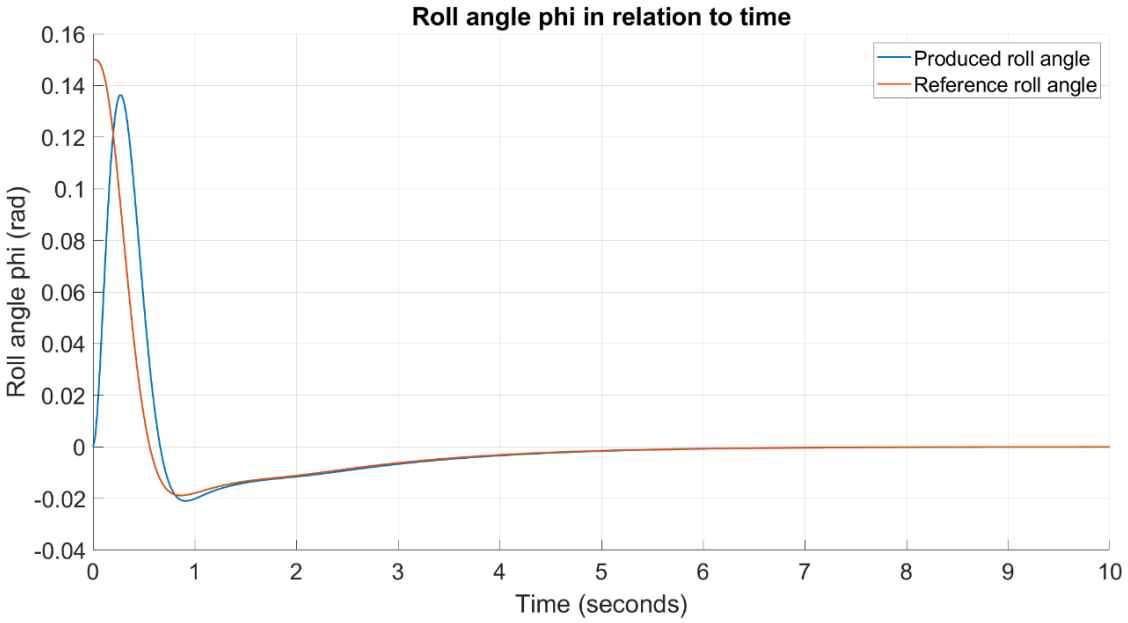


Figure 14. Roll angle φ in relation to time for step reference input using the PID design

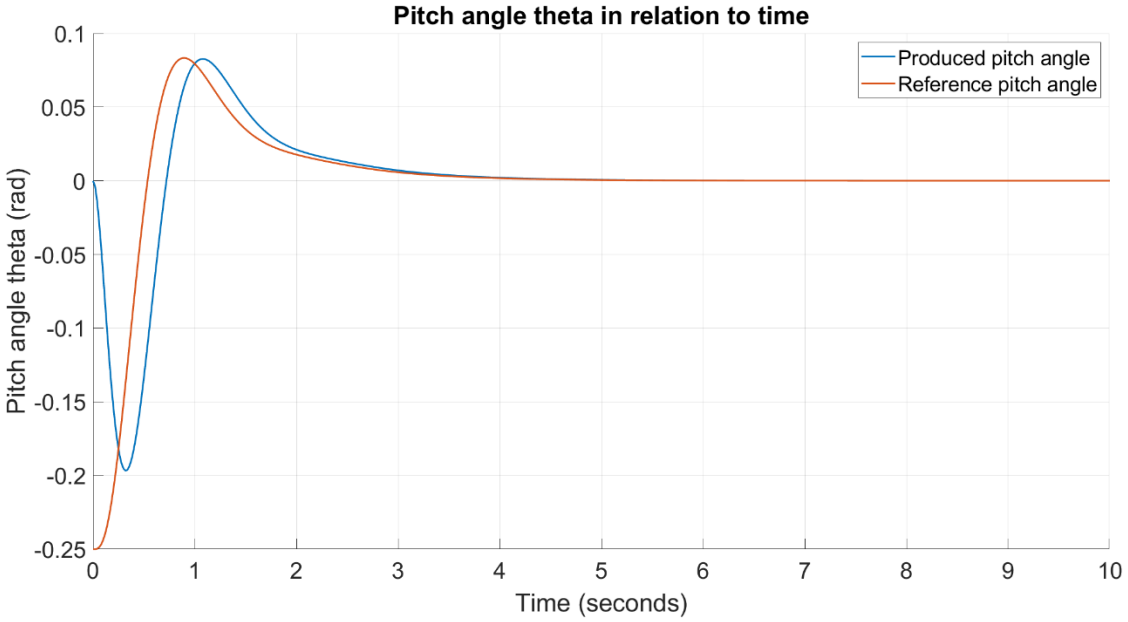


Figure 15. Pitch angle θ in relation to time for step reference input using the PID design

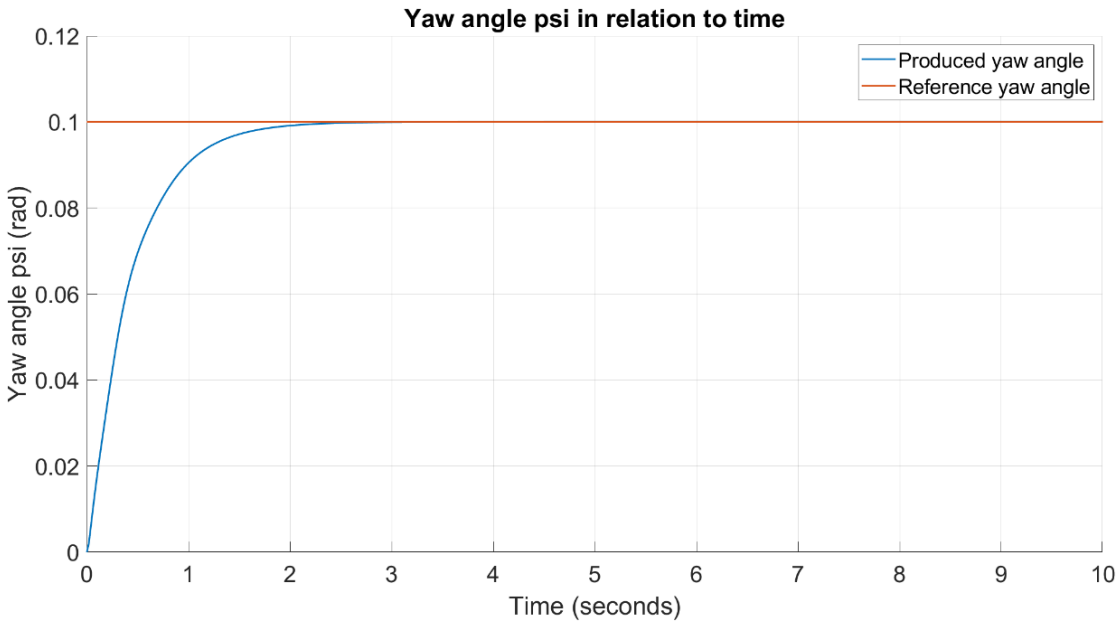


Figure 16. Yaw angle ψ in relation to time for step reference input using the PID design

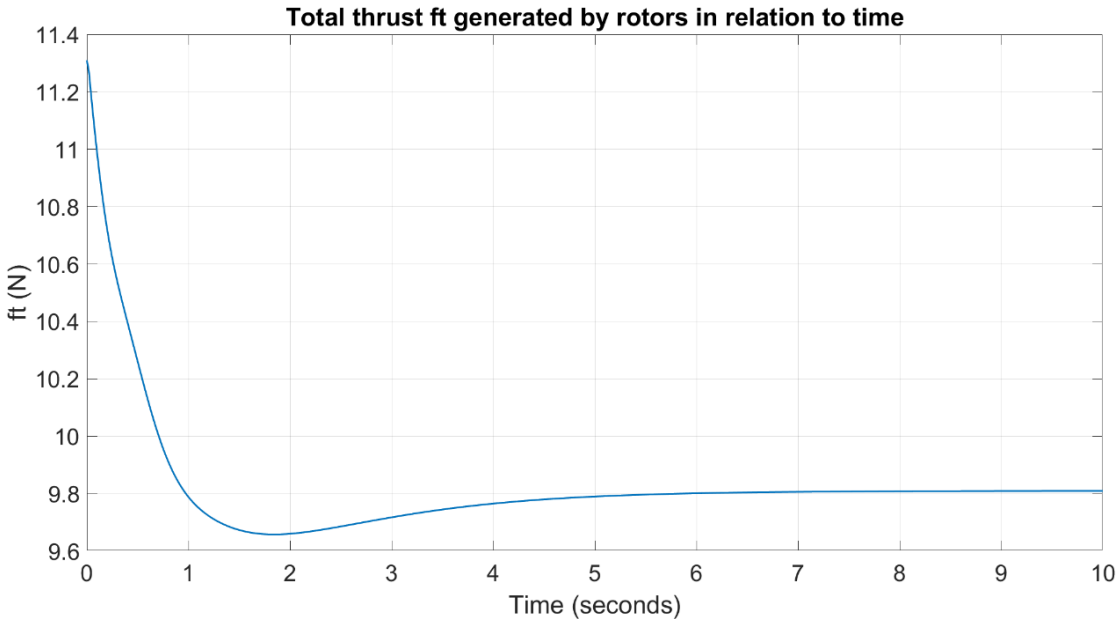


Figure 17. Thrust f_t in relation to time for step reference input using the PID design

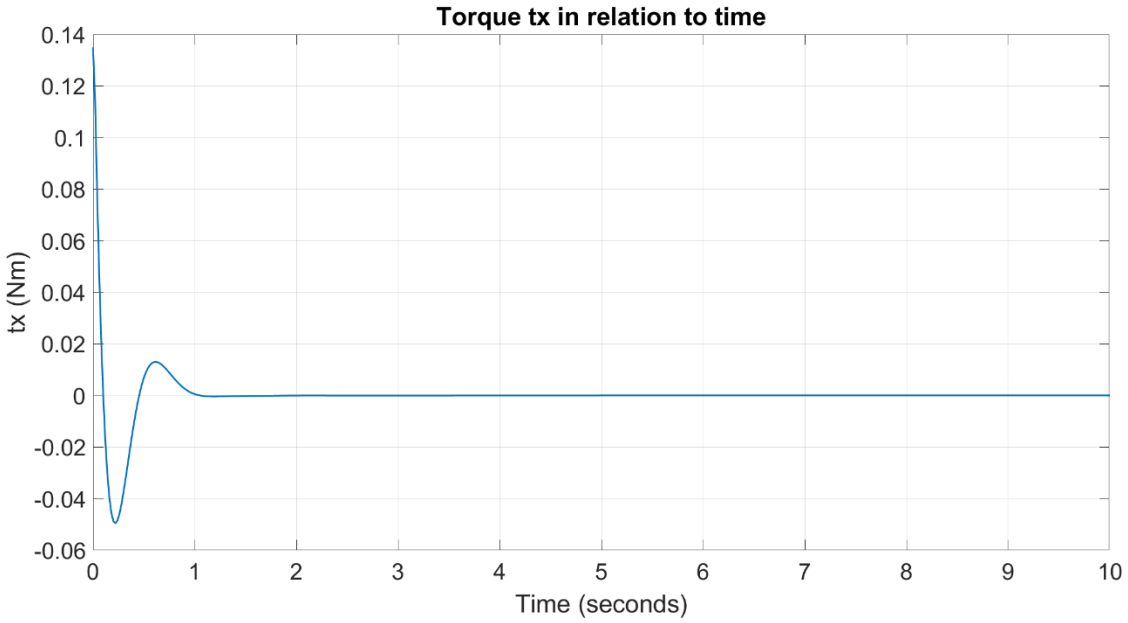


Figure 18. Torque t_x in relation to time for step reference input using the PID design

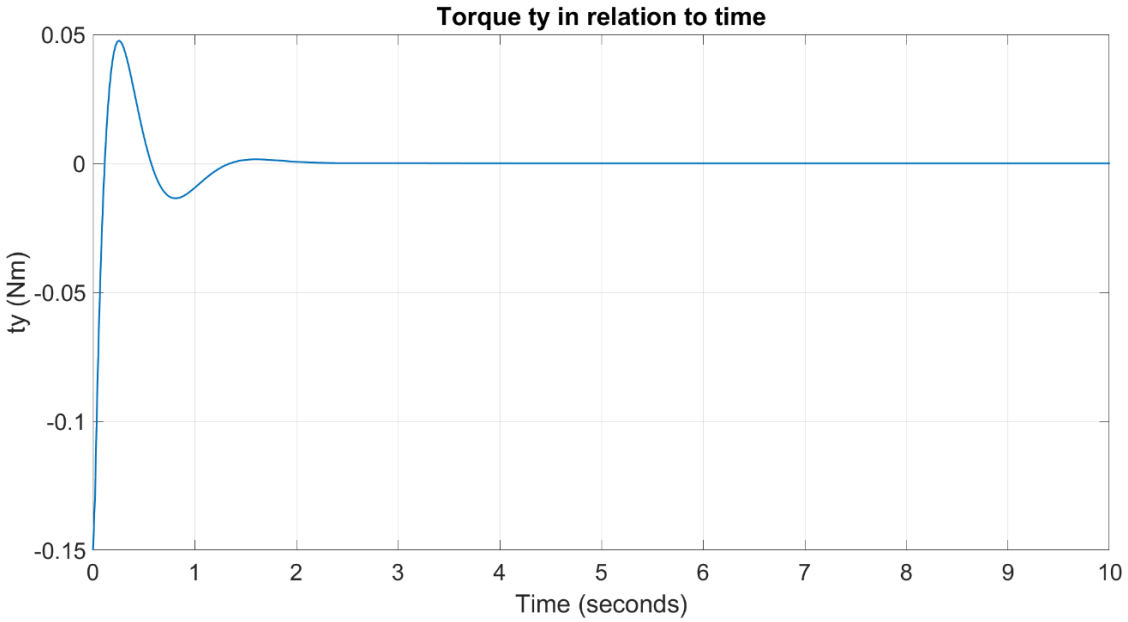


Figure 19. Torque t_y in relation to time for step reference input using the PID design

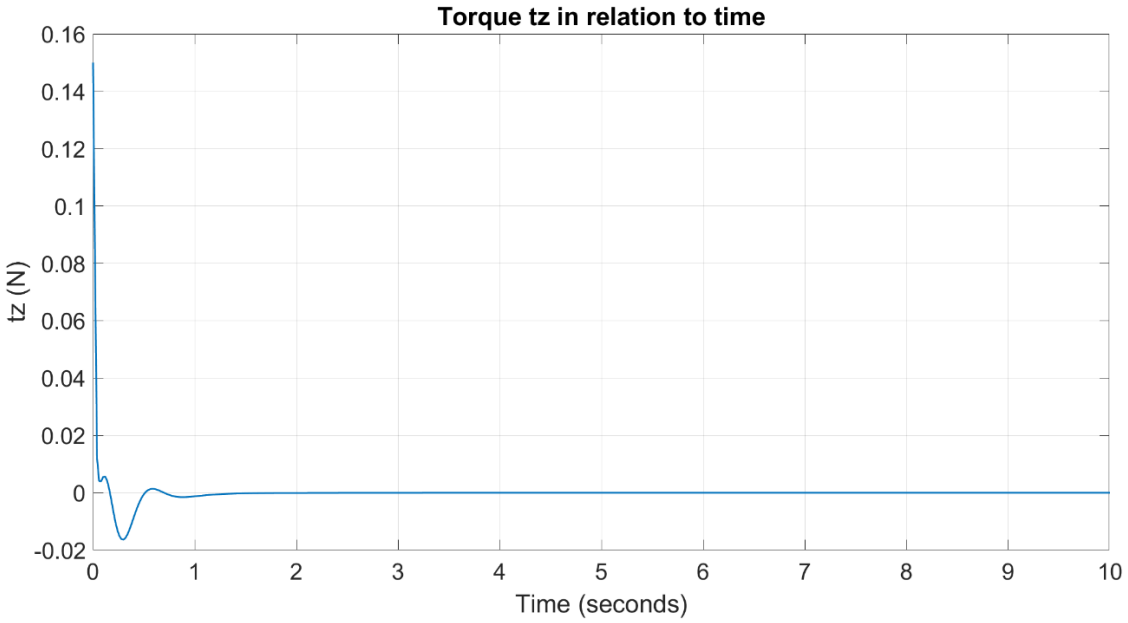


Figure 20. Torque t_z in relation to time for step reference input using the PID design

Based on the graphs presented, it can be observed that the designed PID control system performs well in controlling the quadrotor for a step reference input. More specifically, the x, y, and z positions, as well as the yaw angle ψ, reach their appropriate values in a relatively short settling and rising time, without any overshoot or oscillations. At the same time, the integral part of the PID controllers, despite its small value, ensures that there is zero steady state error.

The roll and pitch angles have an interesting response, due to the fast dynamics of the quadrotor system; they exhibit sharp movements before stabilizing to zero. Nevertheless, these angles remain small and do not exceed the generally considered boundaries of $\left[-\frac{\pi}{2}\ \frac{\pi}{2}\right]$. The same logic applies for the torques and the thrust produced (i.e., the control vector) that are small in value, present a quick response and stabilize at zero when the quadrotor has reached the desired reference. The graphs that show the response of the signals produced by the PID controllers are always crucial in evaluating not only the simulation performance, but also the ability of such simulation to successfully work in real-life conditions.

Overall, the graphs demonstrate the success of the cascaded PID control system in controlling a quadrotor for a step reference input, as all controlled variables reach their setpoints in a timely manner and without significant deviations. However, there is always room for improvement, especially when it comes to how fast the system's response is. Comparable results are expected for subsequent steps in different timesteps.

For the purpose of simulation and analysis, let us assume the following steps that describe a flight trajectory consisting of takeoff, a square flight pattern and landing:

1. step $z = -5\ [m]$ at time $t = 0\ [sec]$
2. step $x = 2\ [m]$ at time $t = 15\ [sec]$
3. step $y = 2\ [m]$ at time $t = 30\ [sec]$
4. step $x = 0\ [m]$ at time $t = 45\ [sec]$
5. step $y = 0\ [m]$ at time $t = 60\ [sec]$
6. step $z = 0\ [m]$ at time $t = 75\ [sec]$

In this scenario, the quadrotor takes off vertically from a stationary position until it reaches a certain altitude. Then, it transitions to a horizontal flight square pattern and finally to a vertical flight pattern, until it descends back to the starting point. Figure **21**, presents a visual representation of this movement.

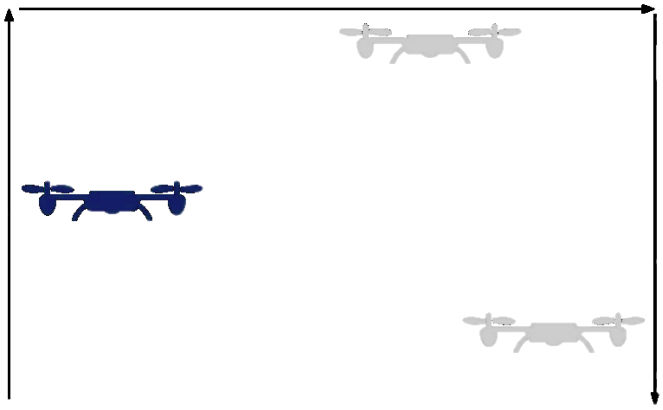


Figure 21. Consecutive steps path visualization

The PID control system's performance for the concept of consecutive steps path is shown in the figures below. It should be noted that the tuning of the PIDs is the same as that of the previous simulation and that the yaw angle ψ is kept at zero [rad]. This is the case for most quadrotor flights, as they use differential thrust to control their yaw motion. When the yaw angle is zero, the vehicle is in a stable hover, the thrust from each rotor is balanced and any desired change in yaw direction is achieved by adjusting the relative speeds of the rotors. Figures 22 to 31 present the produced graphs.

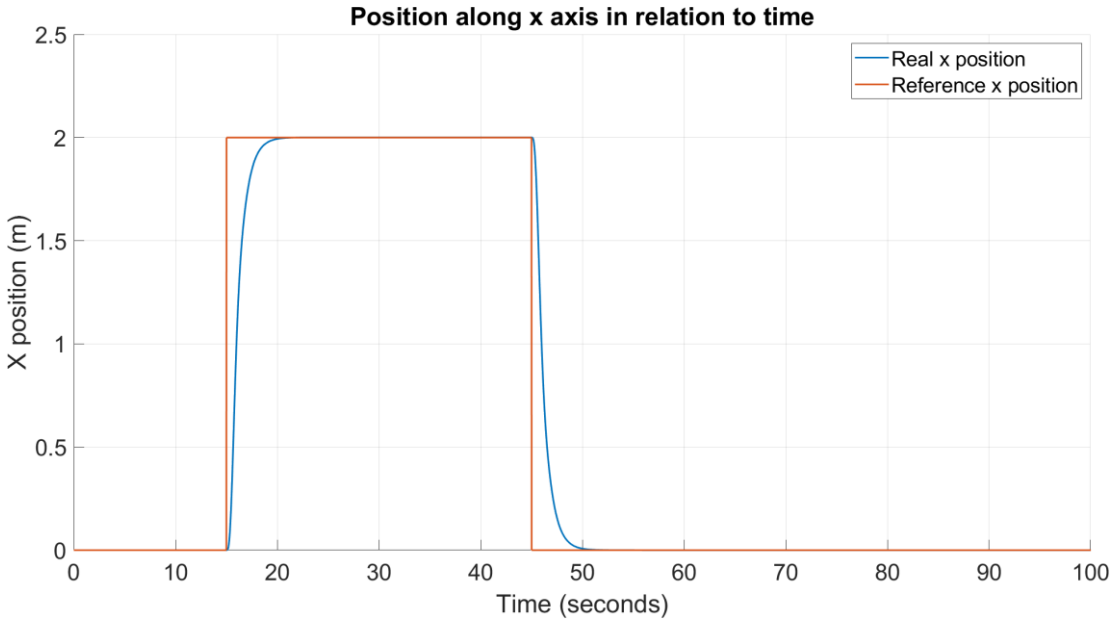


Figure 22. Position x in relation to time for step path reference using the PID design

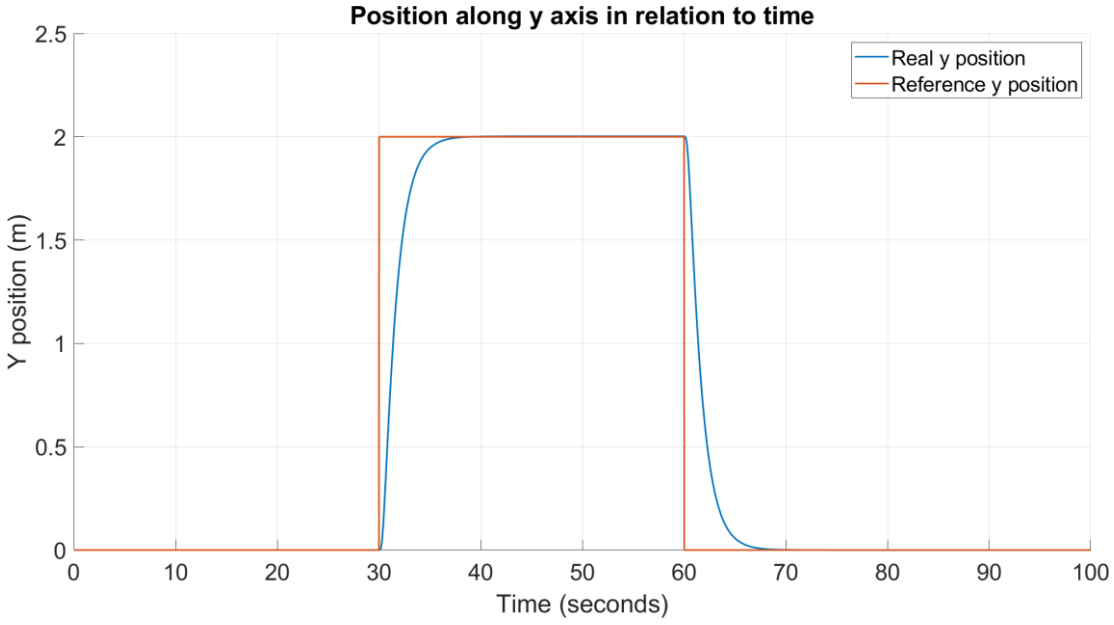


Figure 23. Position y in relation to time for step path reference using the PID design

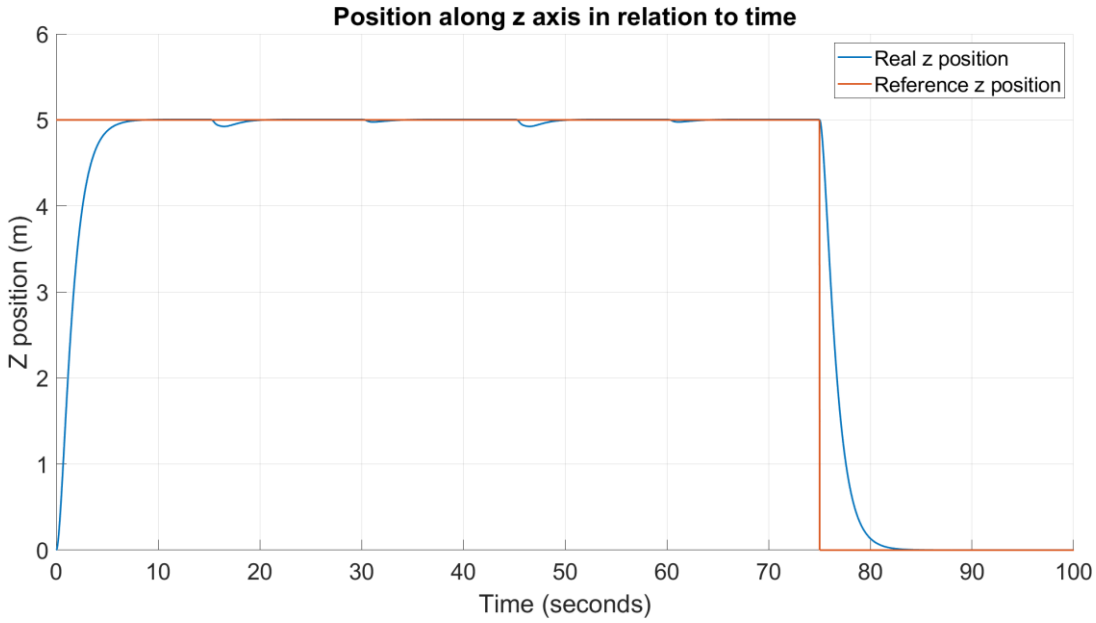


Figure 24. Position z (altitude) in relation to time for step path reference using the PID design

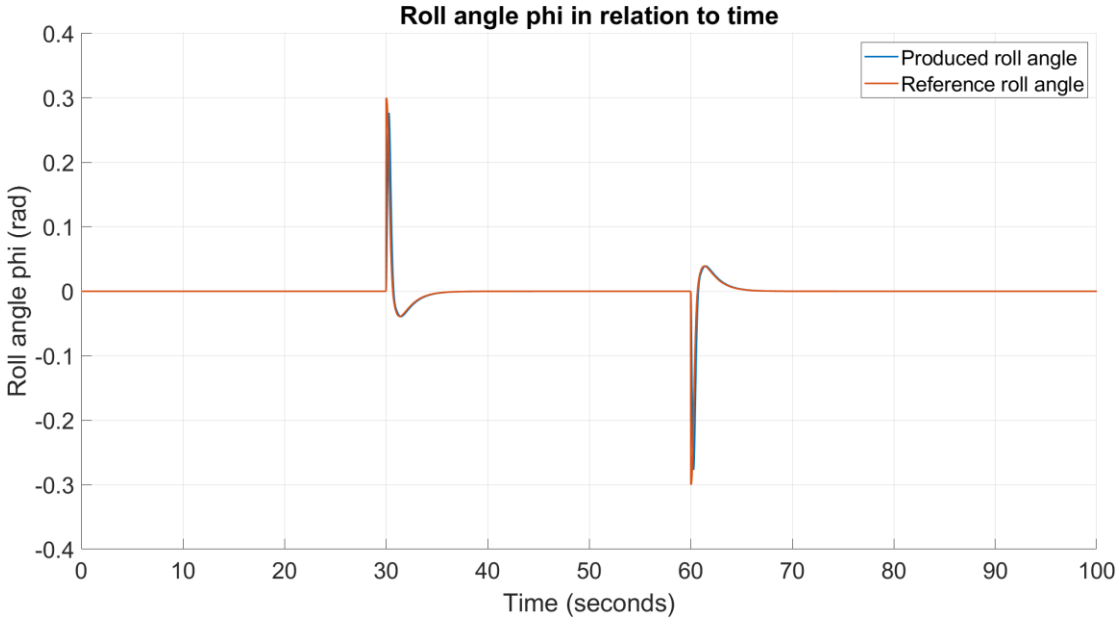


Figure 25. Roll angle φ in relation to time for step path reference using the PID design

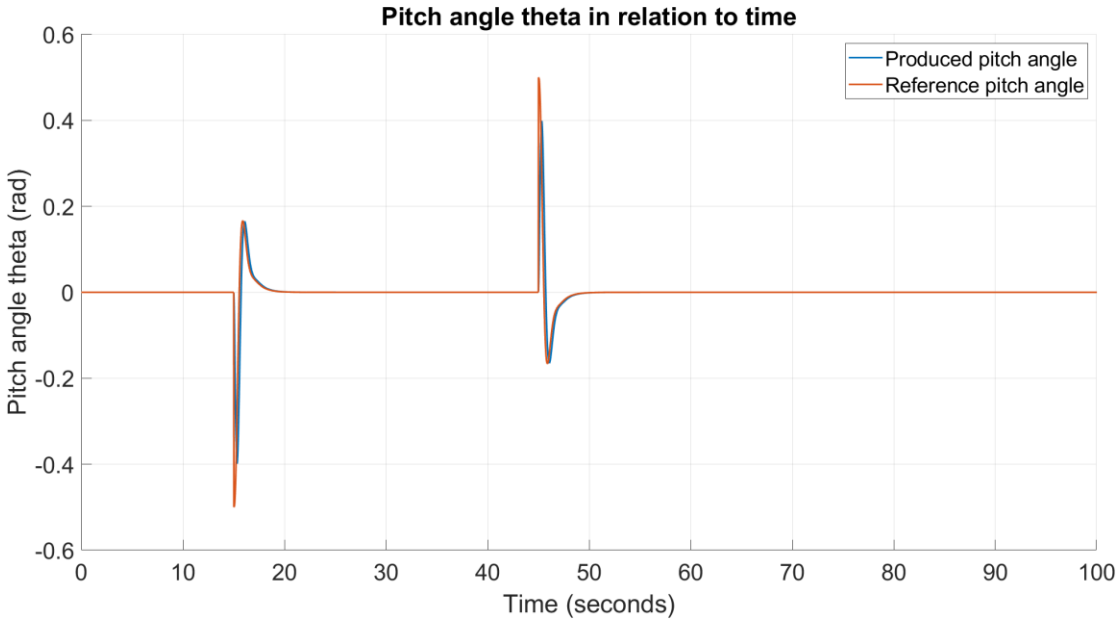


Figure 26. Pitch angle θ in relation to time for step path reference using the PID design

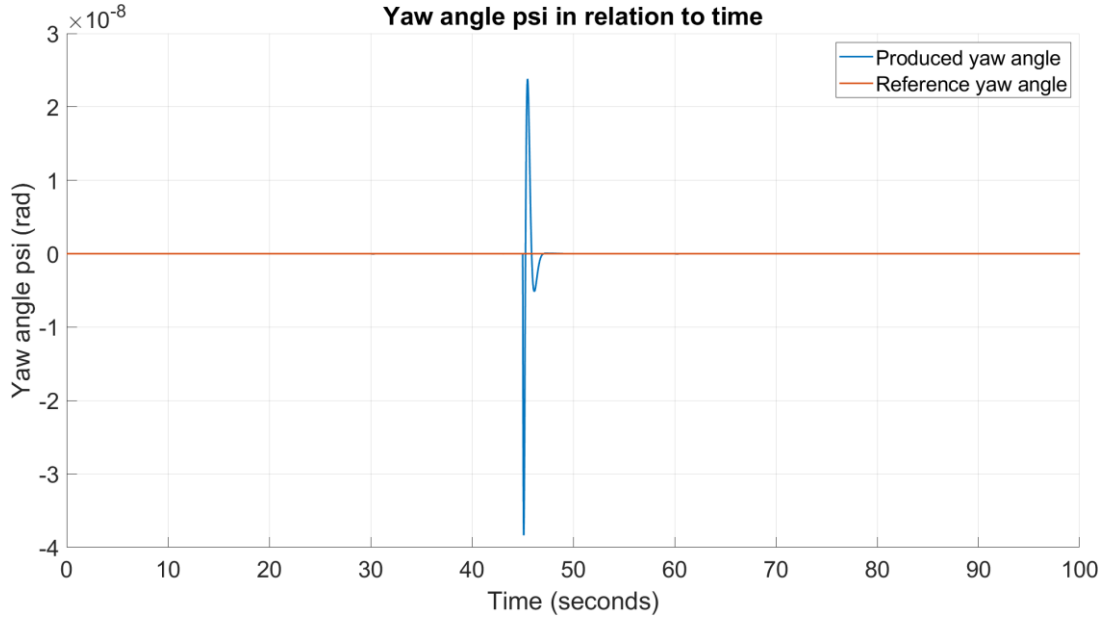


Figure 27. Yaw angle ψ in relation to time for step path reference using the PID design

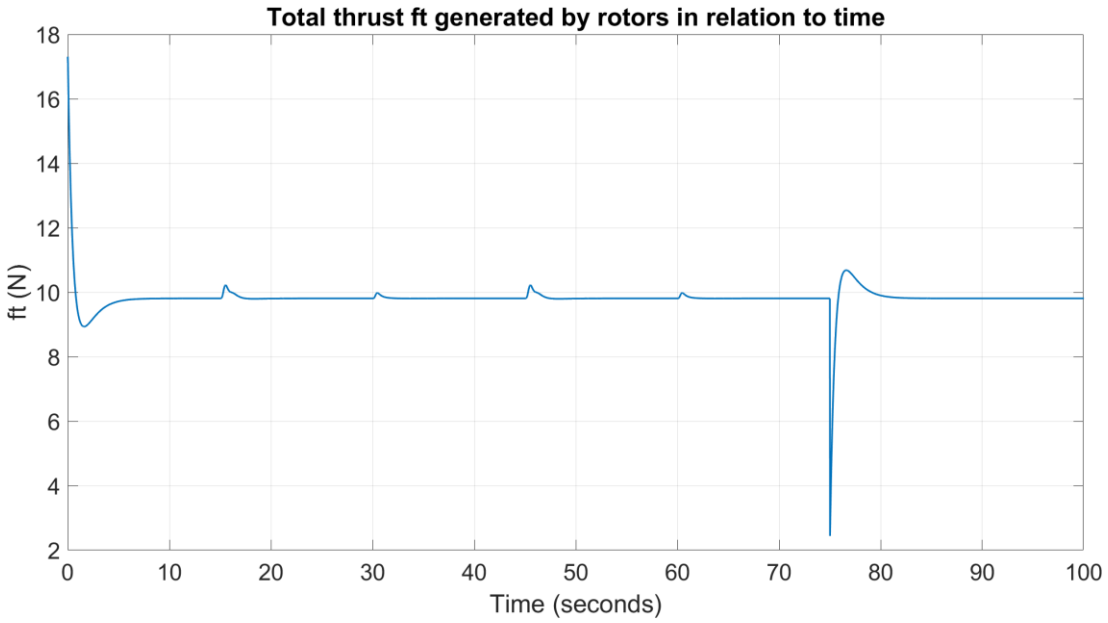


Figure 28. Thrust f_t in relation to time for step path reference using the PID design

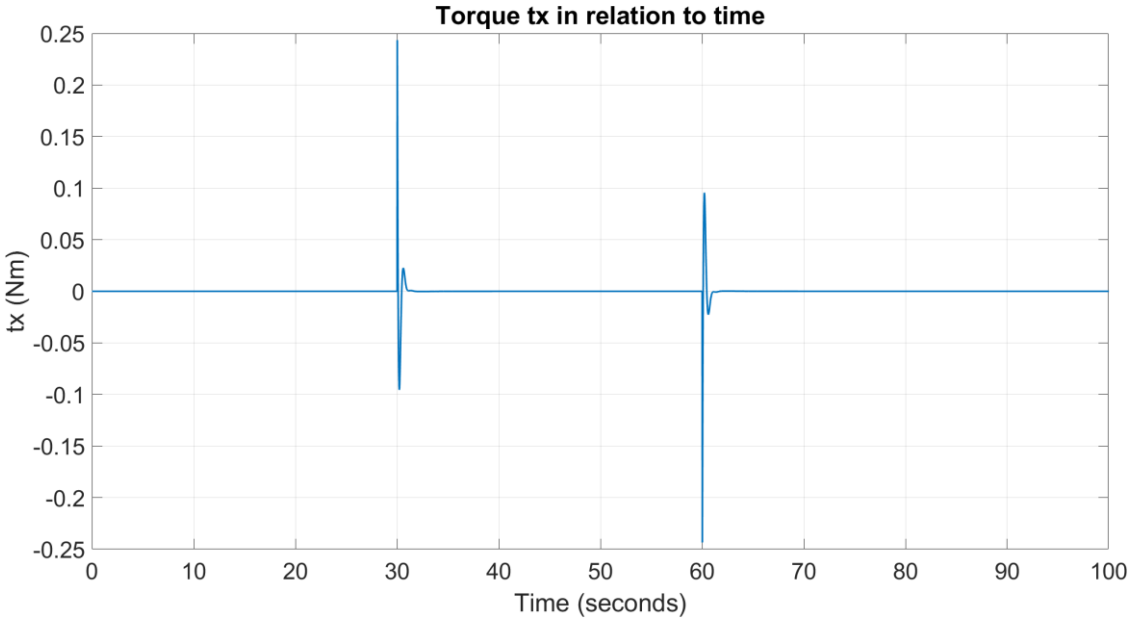


Figure 29. Torque t_x in relation to time for step path reference using the PID design

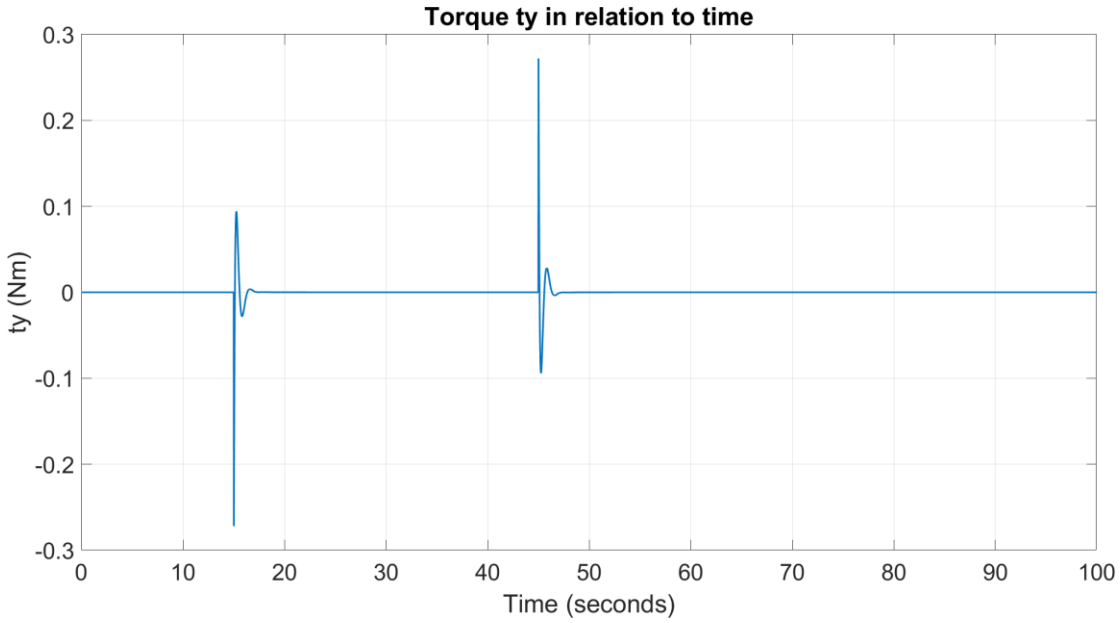


Figure 30. Torque t_y in relation to time for step path reference using the PID design

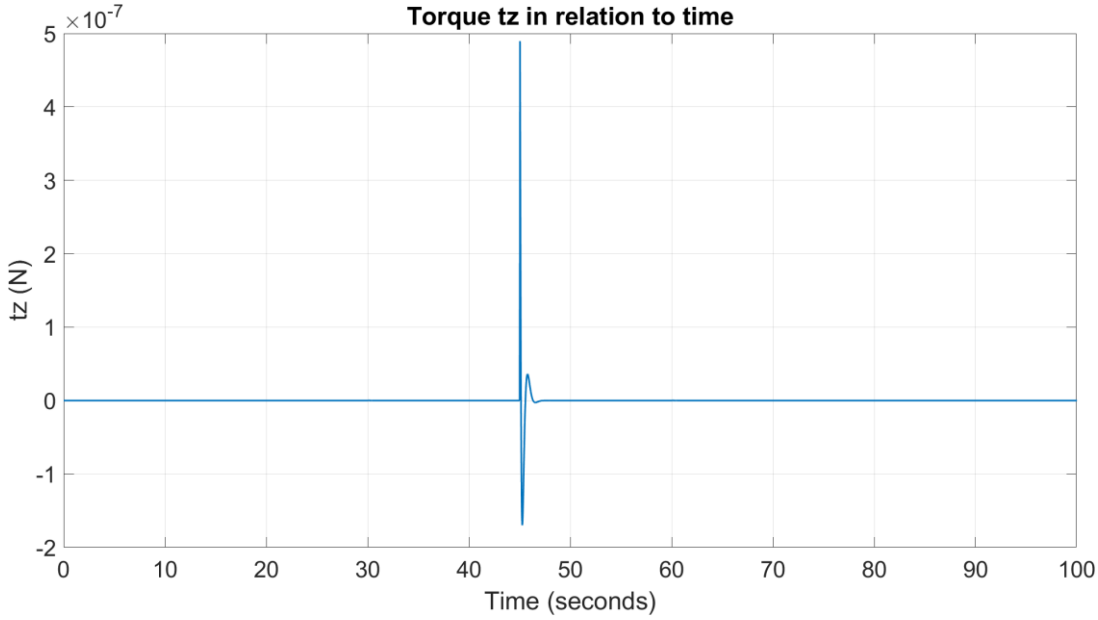


Figure 31. Torque t_z in relation to time for step path reference using the PID design

This simulation demonstrates the success of the designed cascaded PID control system for the step path reference input. Although there are slight differences compared to the better performance of the control system at step input (i.e., minor deviations at some points for the height z), the system's performance is satisfactory.

2.4..2 Simulation for trajectory tracking

The ability of a system to follow a desired course or trajectory over time is referred to as *trajectory tracking* or *path following*. The reference trajectory designates the desired path that the system should follow, while the end goal is to achieve precise and reliable control of the system.

Trajectory path is considered more complex than the step inputs, as it involves tracking a continuously changing path, rather than simply responding to a step change. More specifically, in trajectory tracking the control system must take into consideration not only the current state of the system, but also other parameters, to smoothly follow the desired path without overshooting or oscillating. For comparison and analysis reasons, two trajectory paths are examined:

1) The first trajectory set is:

$$x_{ref(t)} = 0.5 \cdot \cos(0.15 \cdot t)$$
$$y_{ref(t)} = 0.5 \cdot \sin(0.15 \cdot t)$$
$$z_{ref(t)} = -1 - 0.1 \cdot t$$

2) The second trajectory set is:

$$x_{ref(t)} = 0.5 \cdot \cos(0.5 \cdot t)$$
$$y_{ref(t)} = 0.5 \cdot \sin(0.5 \cdot t)$$
$$z_{ref(t)} = -1 -\cdot t$$

The trajectory paths used in this study are continuous, which eliminates the need for the PID controllers to be designed in the same way as for a step input. However, trajectory tracking is typically more challenging than following a constant input, particularly for rapid trajectories. As a result, it is expected that the designed PID control system will not perform as well for the trajectory paths as it did for step references, especially for the second set of trajectories which is faster than the first one, as indicated by the higher frequency.

Figure **32** and Figure **33** present the updated design of the PID controllers, while Table **4** displays the PID tuning parameters used to track both trajectory sets. Finally, the simulation graphs are presented, as well as a graph representing the system's performance by showing both the actual and the reference trajectories on the same axes.

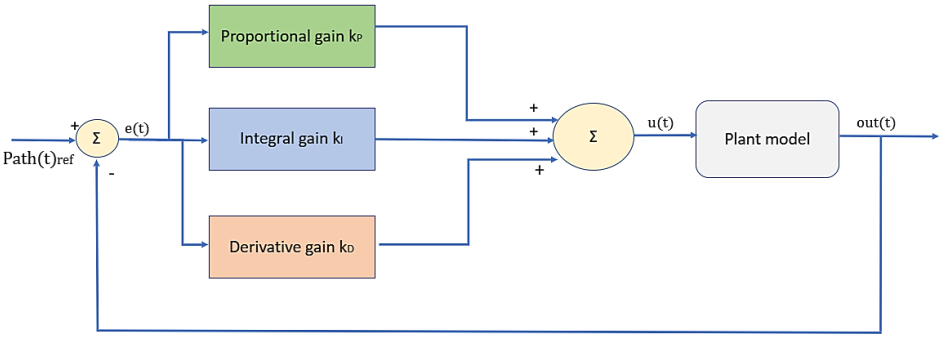


Figure 32. The general PID design for the trajectory tracking

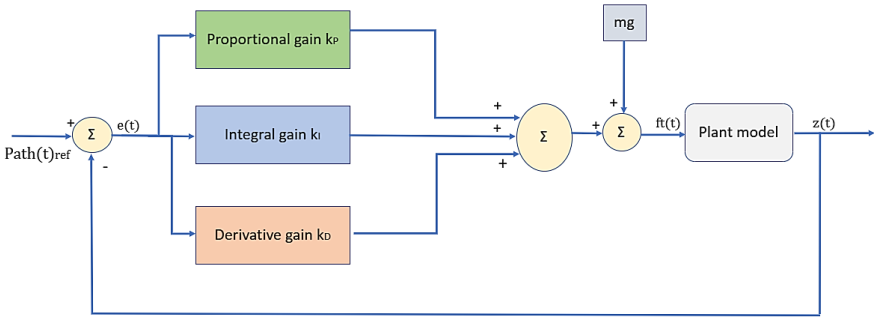


Figure 33. The PID design for the controller of the altitude z, for trajectory tracking

Table 4. The PID parameters for the trajectory tracking – PID design control

	Proportional gain k_P	Integral gain k_I	Derivative gain k_D
PID controller for position x	-0.25	-0.001	-0.3
PID controller for position y	0.05	0	0.45
PID controller for altitude z	-3.5	-0.001	-4
PID controller for roll angle φ	0.2	0	0.1
PID controller for pitch angle θ	0.4	0.0001	0.1
PID controller for yaw angle ψ	0.01	0	0.001

2.4..2.1 Simulation for the 1st trajectory set

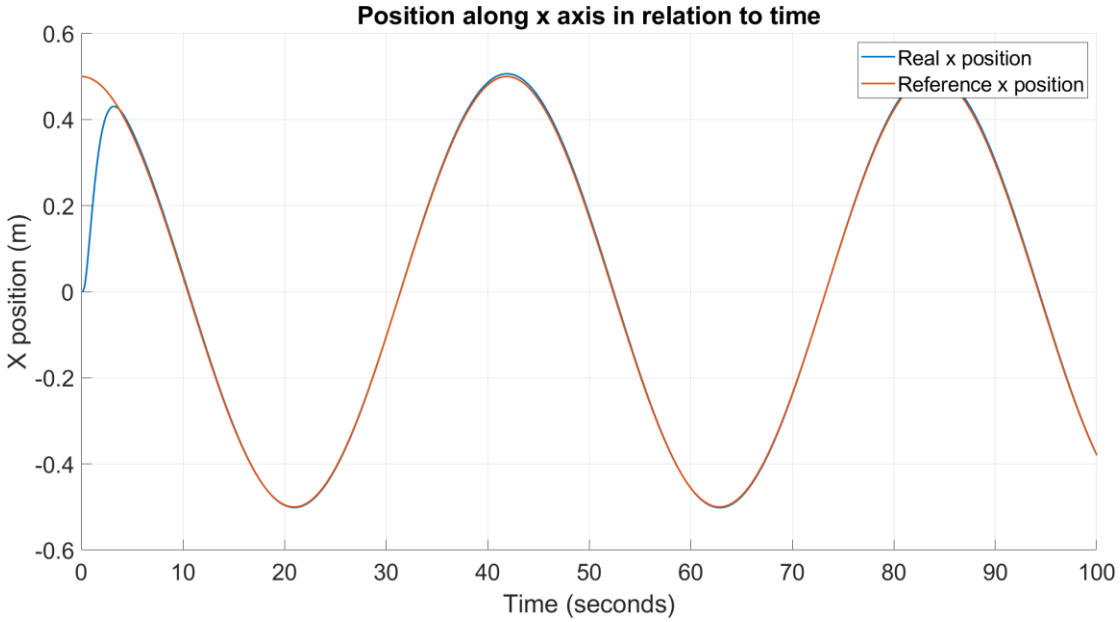


Figure 34. Position x and x_{ref} in relation to time for trajectory tracking (1st set) using the PID design

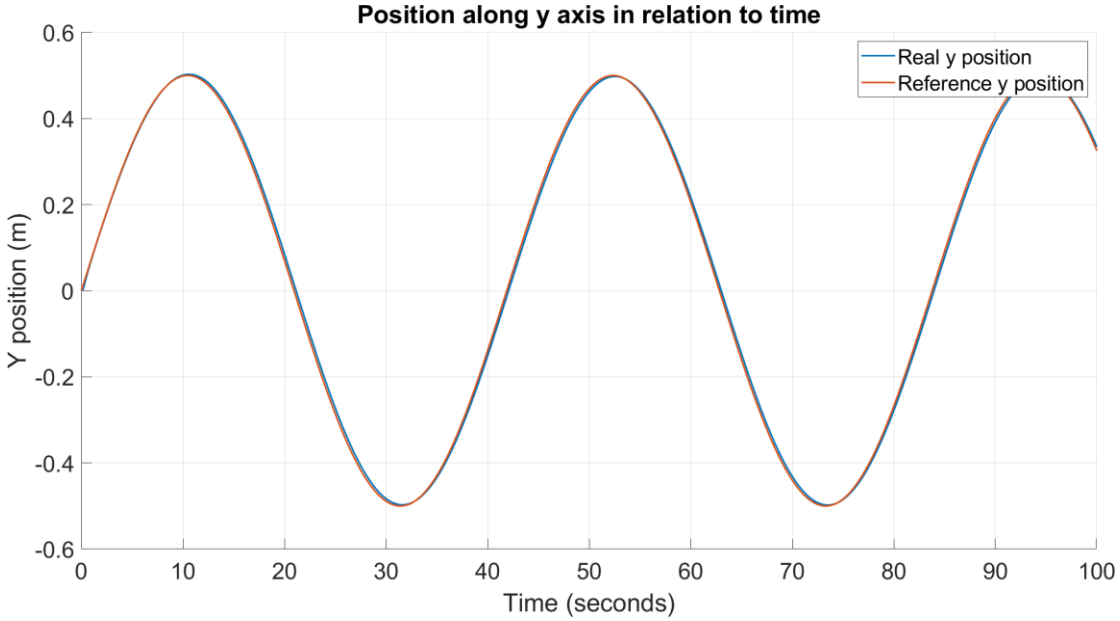


Figure 35. Position y and y_{ref} in relation to time for trajectory tracking (1st set) using the PID design

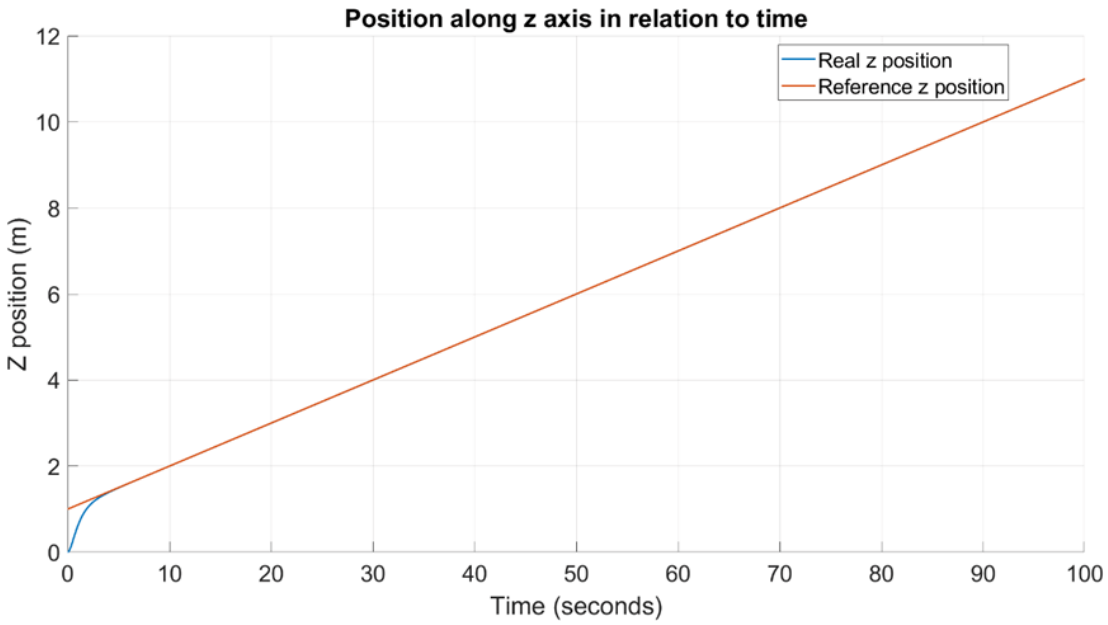


Figure 36. Position z and z_{ref} in relation to time for trajectory tracking (1st set) using the PID design

Based on the simulation graphs depicting the controller's response for the three axes and the reference trajectories (Figure **34**, Figure **35**, Figure **36**), it can be observed that the controller performs adequately well, with room for improvement. The best performance is observed for the y-axis trajectory, followed by the z-axis trajectory, whereas the most challenging one is the x-axis trajectory.

However, it is important to highlight the fact that the most evident deviations from the reference trajectory are at the start of the simulation, which is understandable since the quadrotor's initial conditions may not coincide with the initial reference positions. Therefore, observing initial deviations followed by convergence towards the reference, is an encouraging sign for the controller's performance, as it indicates that the control system can adapt to the reference trajectory and track it, as intended.

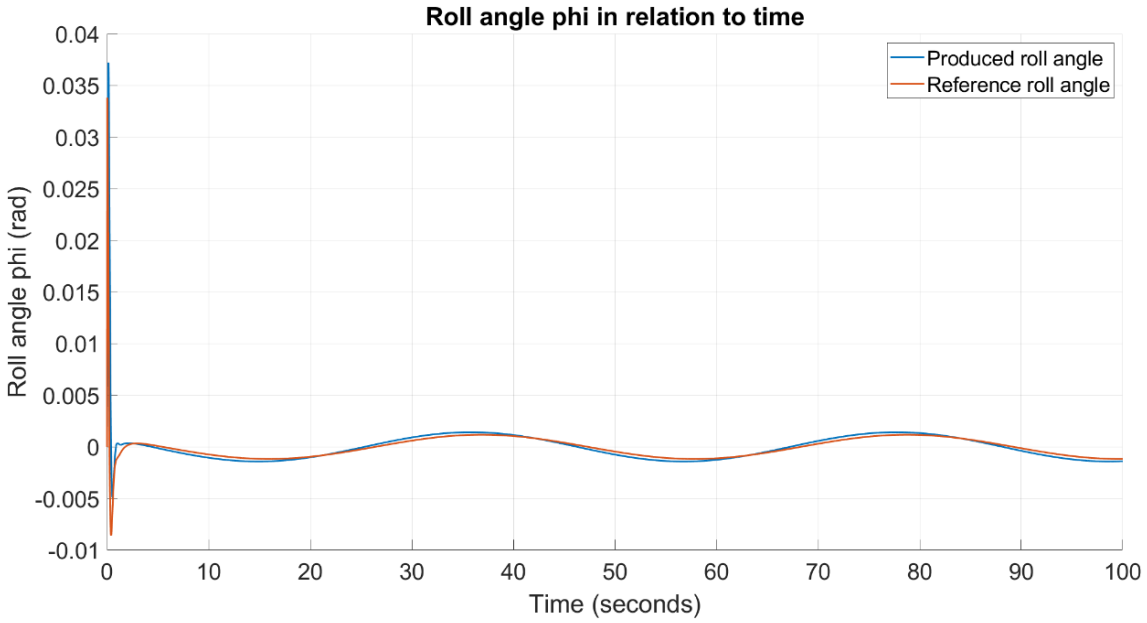


Figure 37. Roll angle φ in relation to time for trajectory tracking (1st set) using the PID design

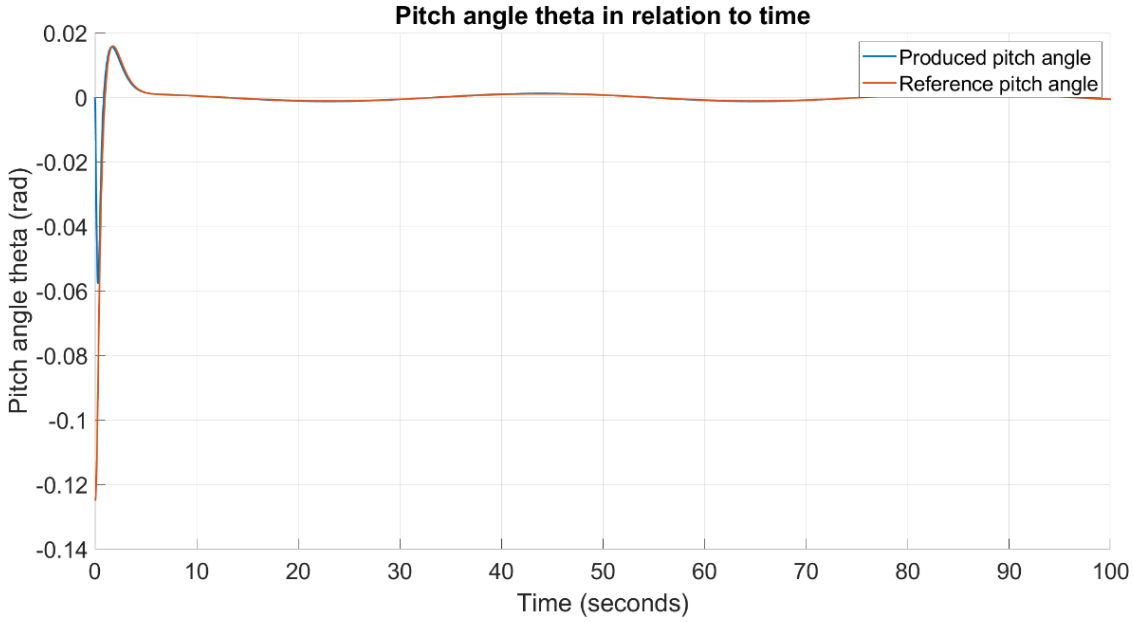


Figure 38. Pitch angle θ in relation to time for trajectory tracking (1st set) using the PID design

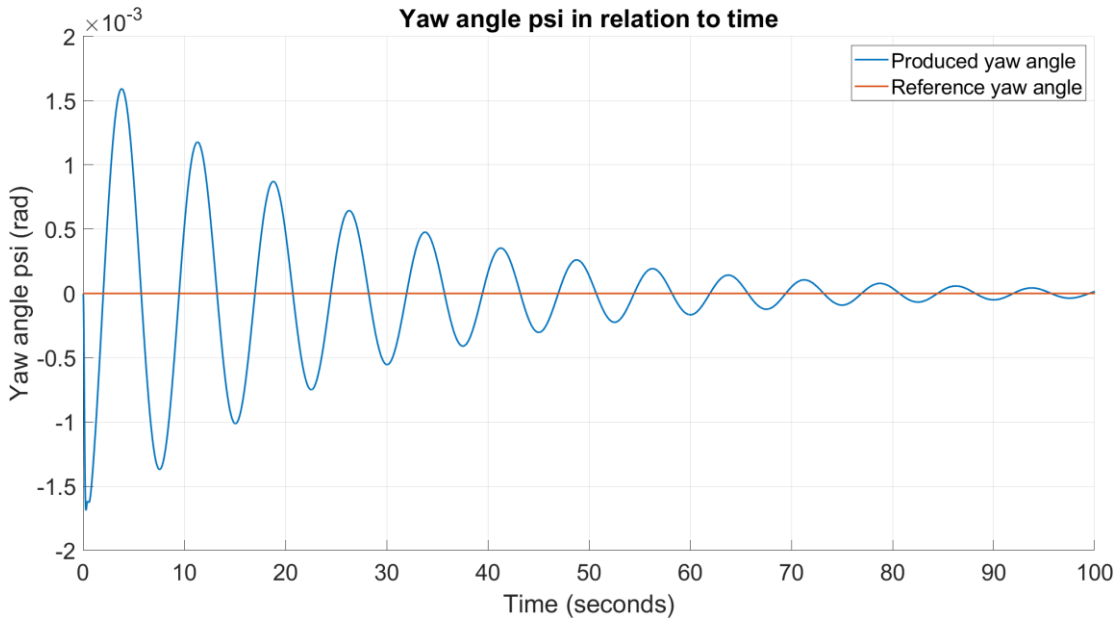


Figure 39. Yaw angle ψ in relation to time for trajectory tracking (1st set) using the PID design

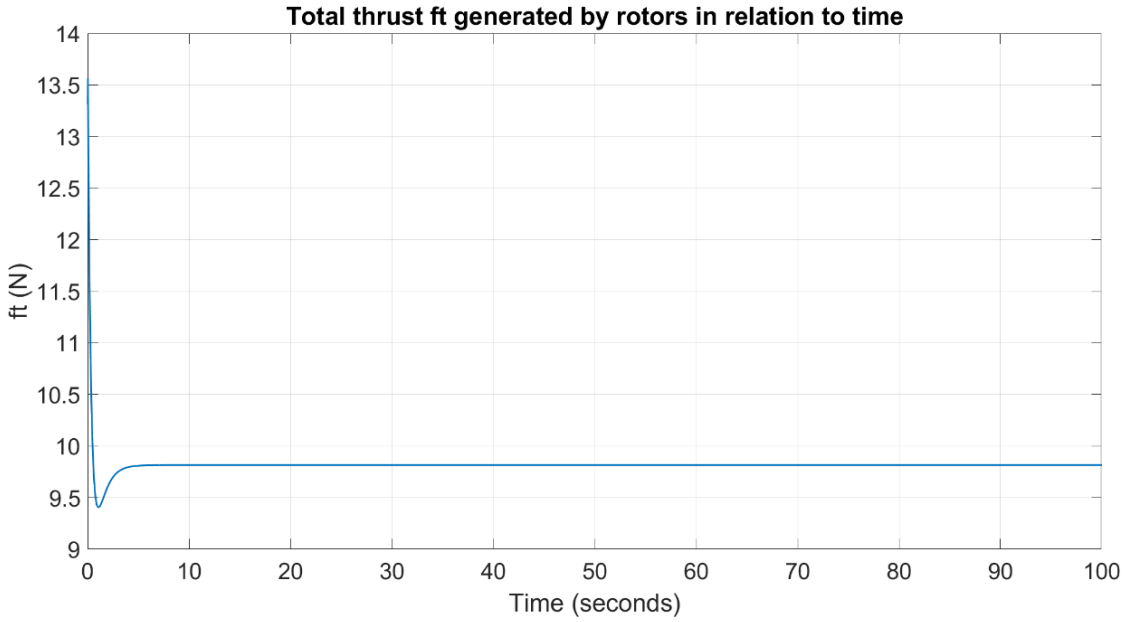


Figure 40. Thrust f_t in relation to time for trajectory tracking (1st set) using the PID design

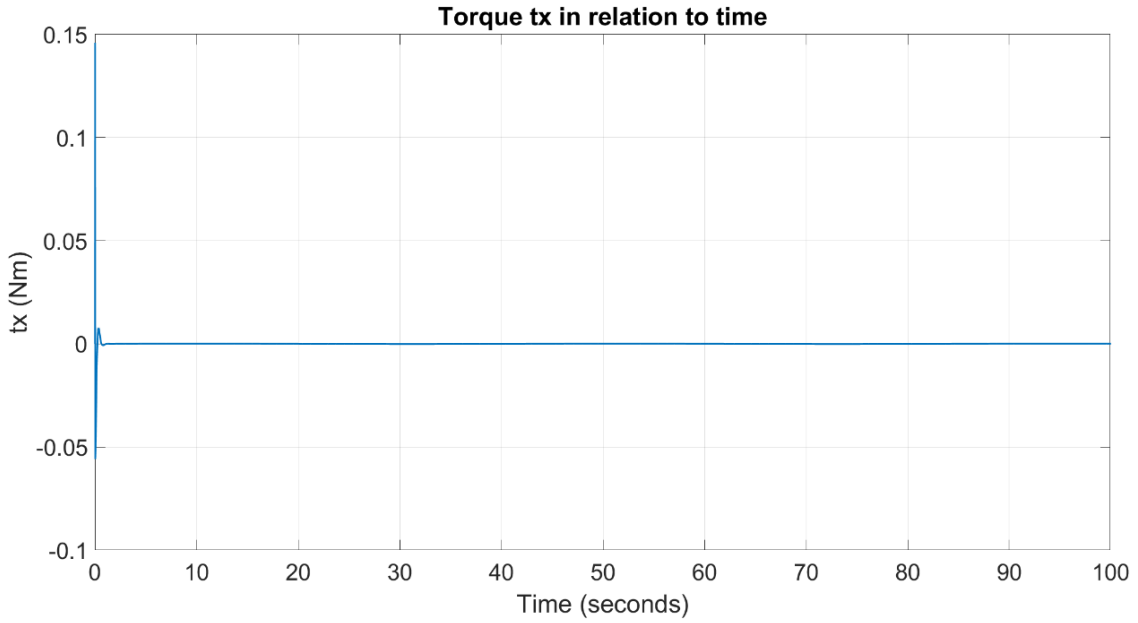


Figure 41. Torque t_x in relation to time for trajectory tracking (1st set) using the PID design

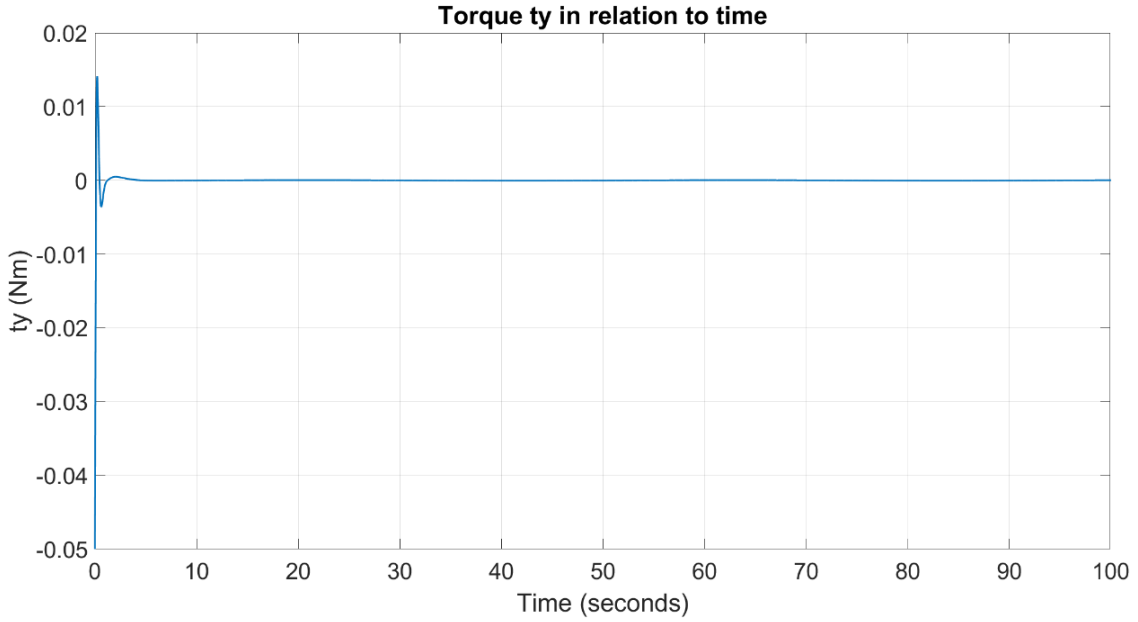


Figure 42. Torque t_y in relation to time for trajectory tracking (1st set) using the PID design

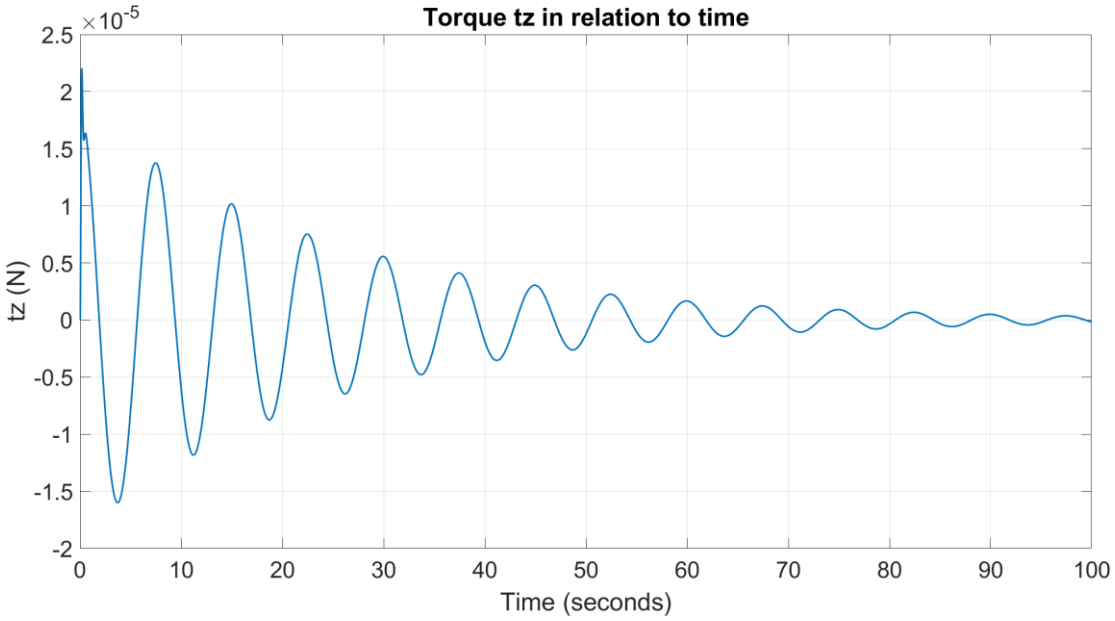


Figure 43. Torque t_z in relation to time for trajectory tracking (1st set) using the PID design

As observed in the corresponding responses, the quadrotor's angles (Figure **37**, Figure **38**, Figure **39**) display sharp movements at the beginning of the simulation and then oscillations. The same is true for the thrust and torques (τ_x, τ_y, and τ_z) produced (Figure **40**, Figure **41**, Figure **42**, Figure **43**). However, they do not exceed the generally considered boundaries and indicate that the controller can control the quadrotor's motion while respecting its physical limitations.

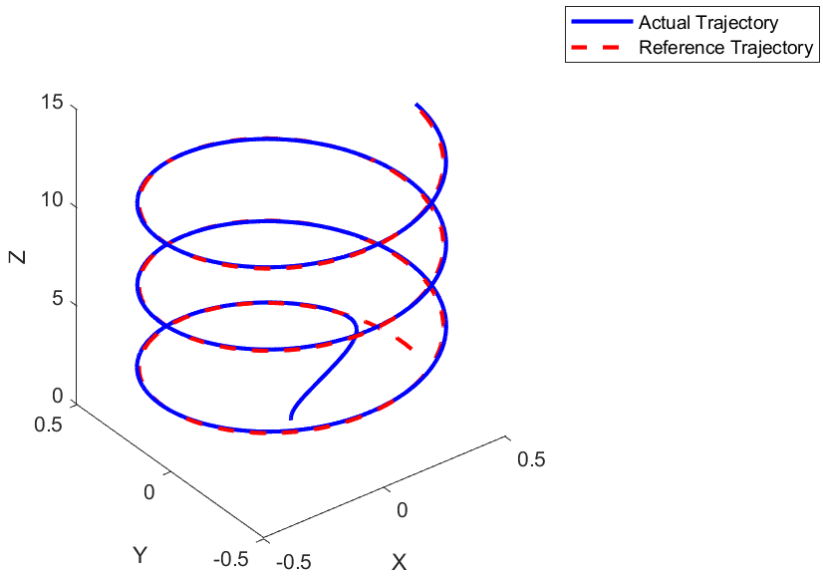


Figure 44. Actual trajectory compared to reference trajectory for the 1st set using the PID design

The graph presented in Figure **44** shows the spiral-like shape of the trajectory set examined for three time periods. More specifically, the quadrotor, starting at (0.5, 0, -1), follows a sinusoidal path in the x and y direction, while simultaneously ascending along the z axis with a linear slope.

By comparing the reference trajectory with the actual trajectory, the controller can track the desired trajectory well, with deviations particularly at the start of the simulation, as explained before, while it manages to minimize those over time. However, visual inspection alone may not provide an accurate assessment of the controller's performance; for more quantitative evaluation, quality estimators like MSE appear to be useful.

Mean Squared Error (MSE) is a commonly used evaluation measure in many fields, including control systems. It is used to evaluate the accuracy of a control system by comparing the actual-produced values to the reference-desired ones through the average of their squared differences, as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.6}$$

where n is the number of data points, y_i the reference value, and $\hat{y}_i$ the actual value. Squaring the differences eliminates negative values for the differences and thus ensuring that the MSE is greater than or equal to zero, with smaller values indicating that the desired trajectory is closer to the actual trajectory and so a better performance is achieved (a controller with MSE = 0 is an ideal one).

The MSE values can be computed for each axis separately, to get a more precise measure of the controller's performance for each reference, but also for the entire trajectory following as a way of evaluating the overall controller's tracking performance. This can be achieved through the average of the MSE calculated for each axis.

Overall, the calculation of MSE is a useful tool for optimizing the system's performance, identifying any potential error or issues that need to be addressed, ensuring that the controller is accurately tracking the desired trajectory and thus helping us make informed decisions about the examined control system or compare different ones.

Apart from the calculation of the MPC value for each one of the three axes, it is also important to calculate the average Euclidean distance between the actual and the reference trajectories; this quantitative evaluation metric provides insights into how closely the actual trajectories of the quadcopter follow the desired ones in the 3D space. That being said, it assesses the overall performance of the control system across all dimensions by considering both the magnitude and the direction of the deviations. As is the case with the simple MSE, smaller values indicate higher accuracy. The mathematical formula for the Mean Euclidean distance is:

$$Mean\ Euclidean\ Distance = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (z_i - \hat{z}_i)^2} \quad (2.7)$$

where n is the number of data points, x_i, y_i, z_i the reference values in the x, y, z axes, and $\hat{x}_i, \hat{y}_i, \hat{z}_i$ the actual values in the x, y and z axes, respectively. For the 1st trajectory set the MSE values calculated for the x, y, and z axes, as well as the Mean Euclidean Distance, are presented in Table 5:

Table 5. The MSE values for the three axes – 1st trajectory using the PID design

	Mean Squared Error (MSE)
x-axis trajectory	0.0020
y-axis trajectory	9.2778 e-05
z-axis trajectory	0.0059
Mean Euclidean Distance	0.0205

Based on the calculated MSE values for the x, y, and z axes trajectories, it becomes apparent that the quadrotor can follow the three trajectories well, with the y-axis trajectory showing the best performance, as indicated by the smallest MSE value (the one closest to zero). The x and z axes trajectories also perform well, with room for improvement, especially for the z axis which presents the largest MSE value. To improve the performance in these directions, it may be necessary to further tune the corresponding PID controllers or consider using a different control system altogether.

However, considering the tracking errors that appear due to the initial conditions in axes x and z, it is fair to say that the results are exceptionally well for a cascaded PID control system that is controlling a quadrotor. Besides, it is important to bear in mind that PID controllers are particularly good at controlling linear systems, but, generally, they do not perform as well for complex non-linear systems as the one examined.

2.4..2.2 Simulation for the 2nd trajectory set

For the tracking of the second trajectory, the corresponding graphs are presented in Figures 45 to 54.

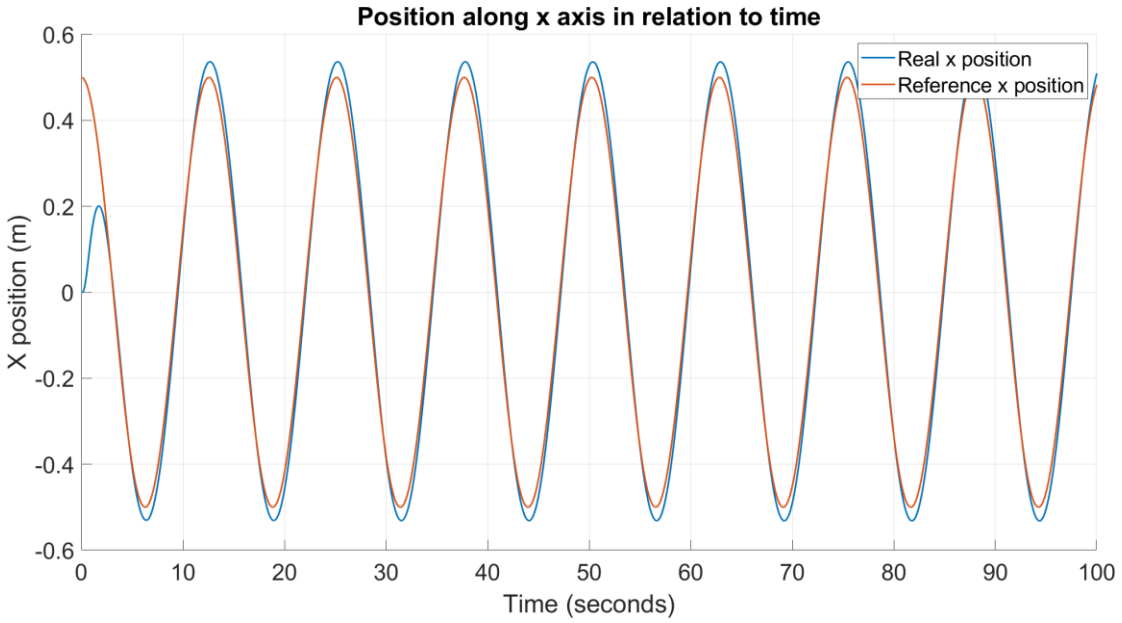


Figure 45. Position x and x_{ref} in relation to time for trajectory tracking (2nd set) using the PID design

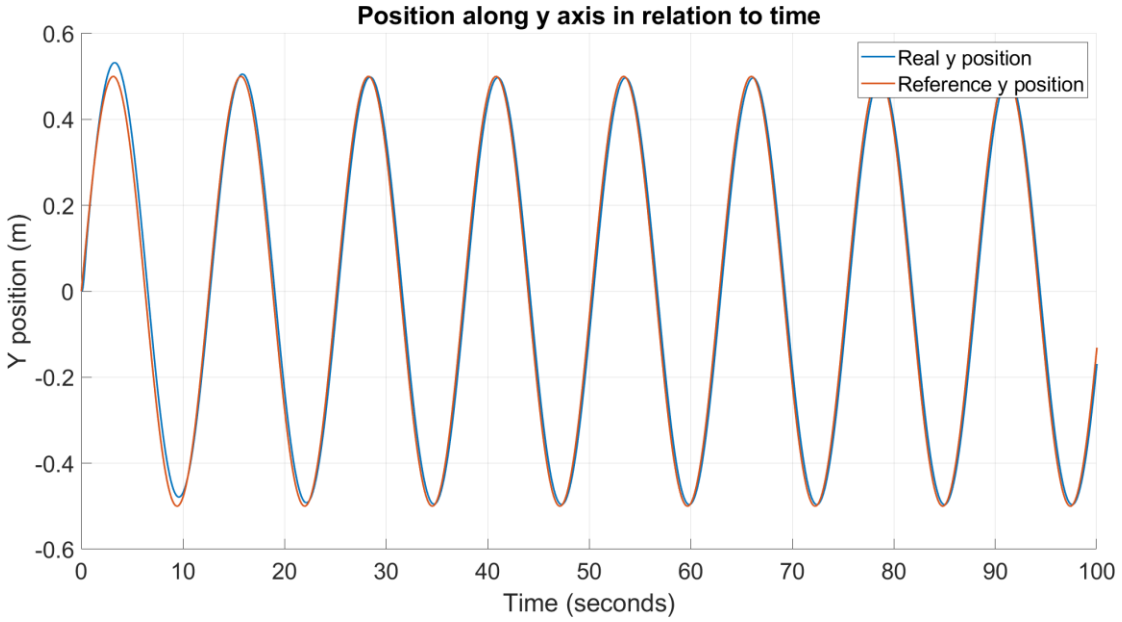


Figure 46. Position y and y_{ref} in relation to time for trajectory tracking (2nd set) using the PID design

Figure 47. Position z and z_{ref} in relation to time for trajectory tracking (2nd set) using the PID design

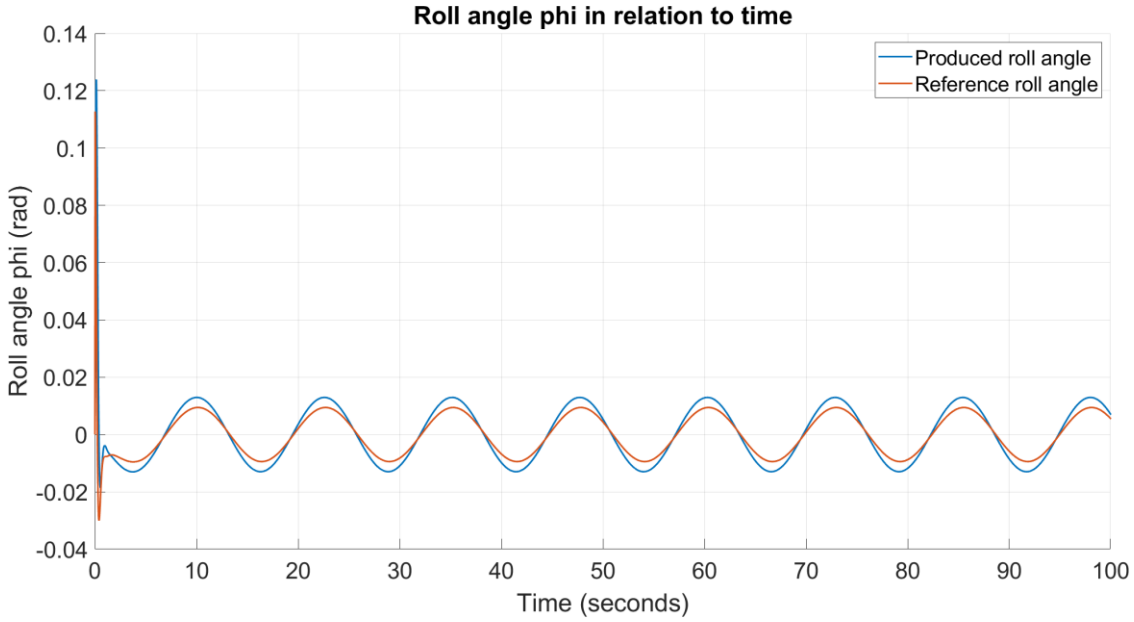


Figure 48. Roll angle φ in relation to time for trajectory tracking (2nd set) using the PID design

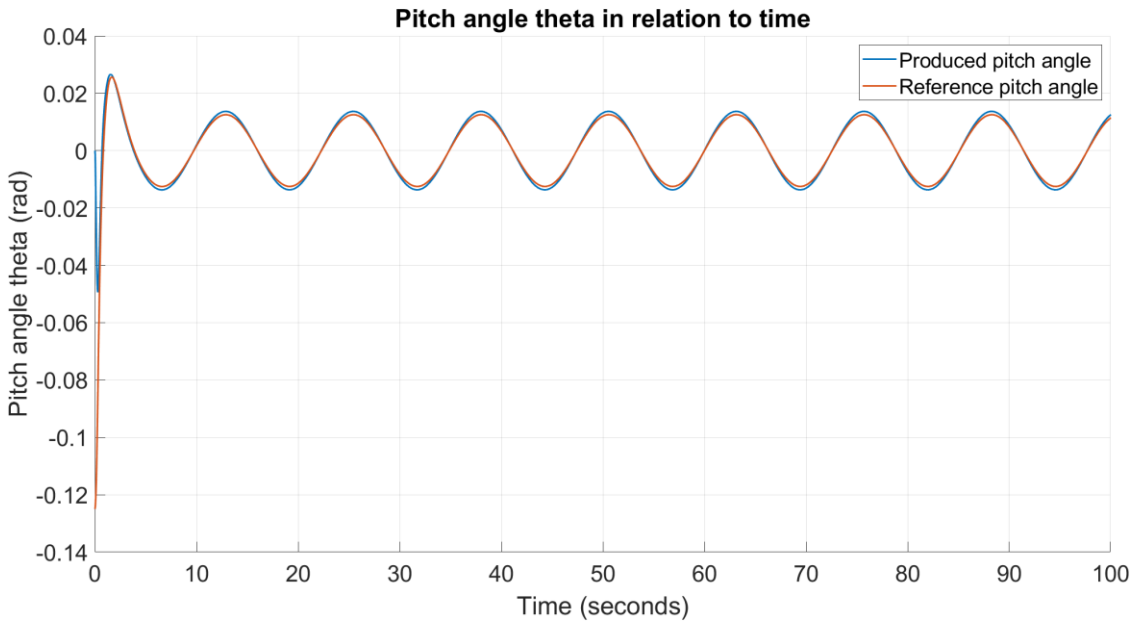


Figure 49. Pitch angle θ in relation to time for trajectory tracking (2nd set) using the PID design

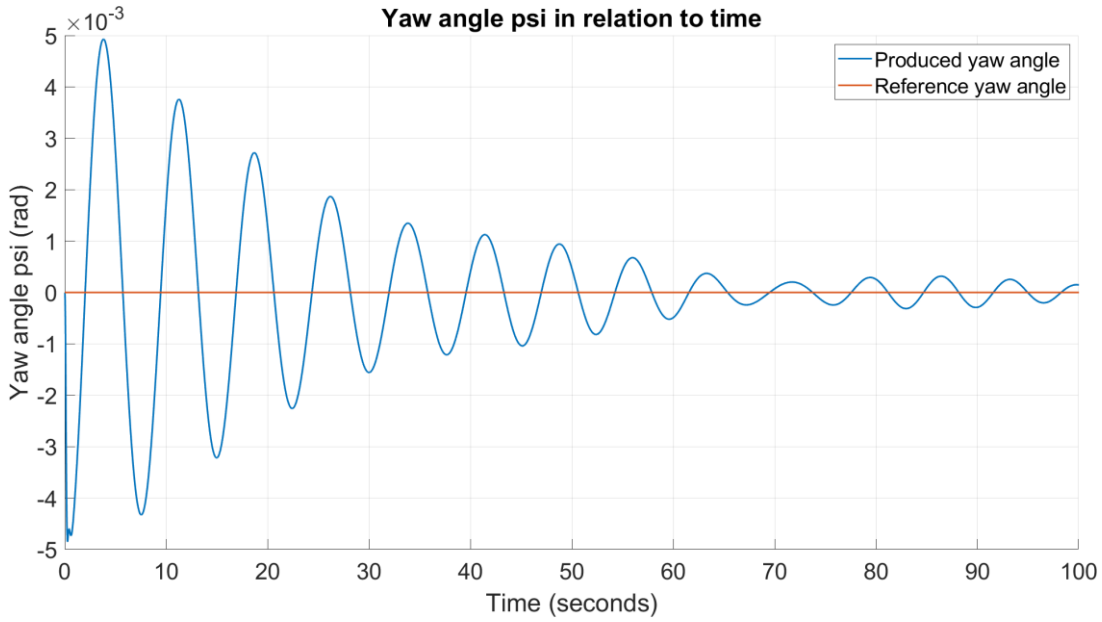


Figure 50. Yaw angle in relation to time for trajectory tracking (2nd set) using the PID design

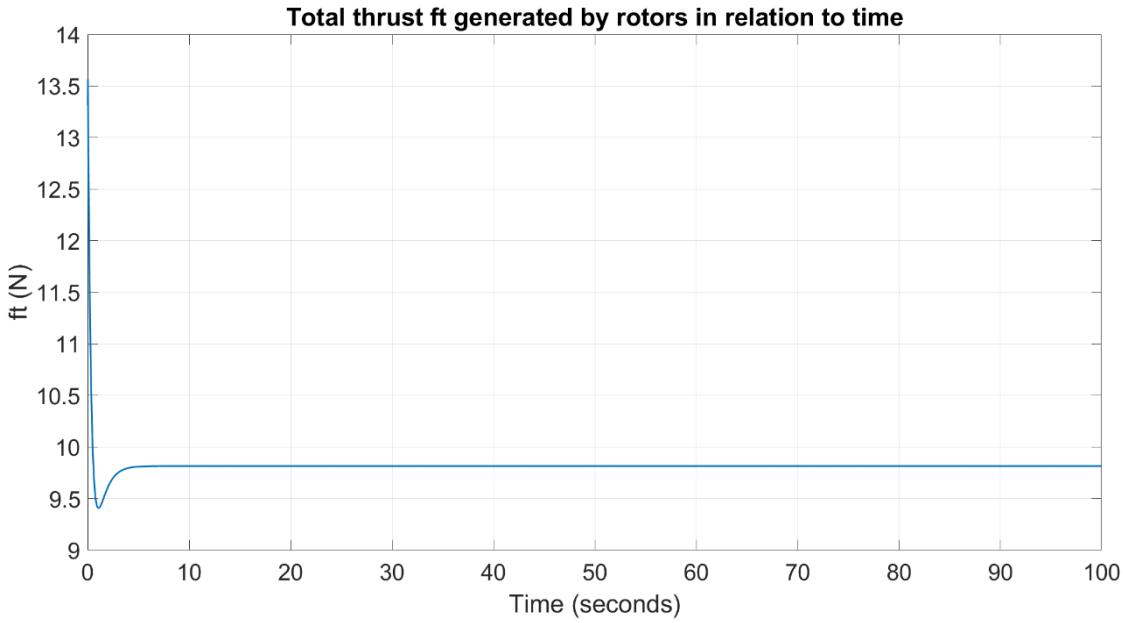


Figure 51. Thrust f_t in relation to time for trajectory tracking (2nd set) using the PID design

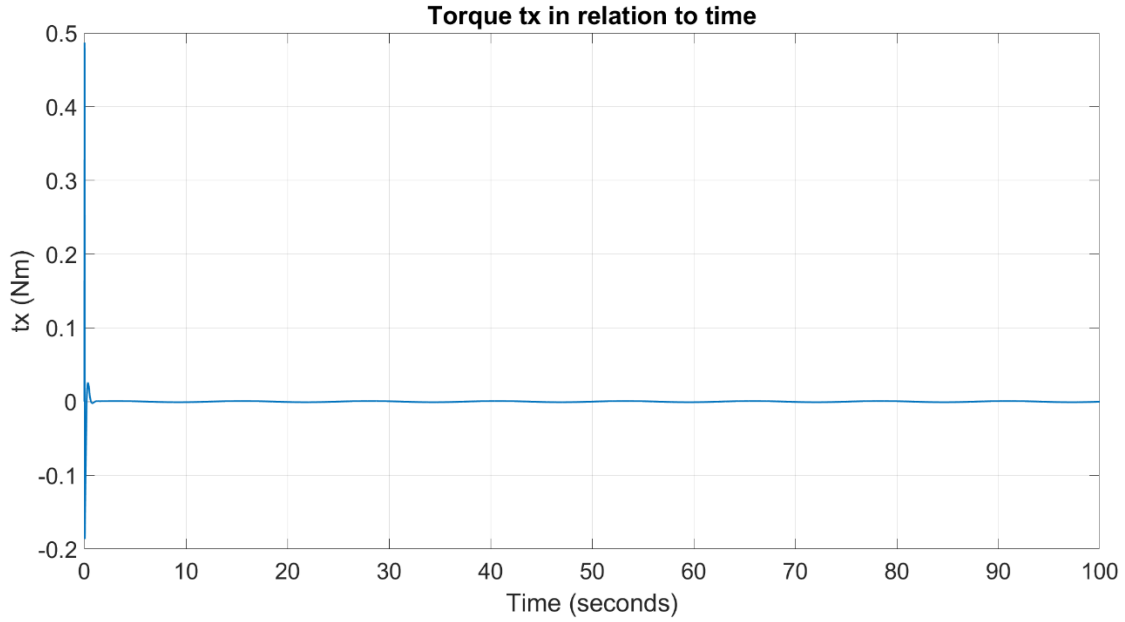


Figure 52. Torque t_x in relation to time for trajectory tracking (2nd set) using the PID design

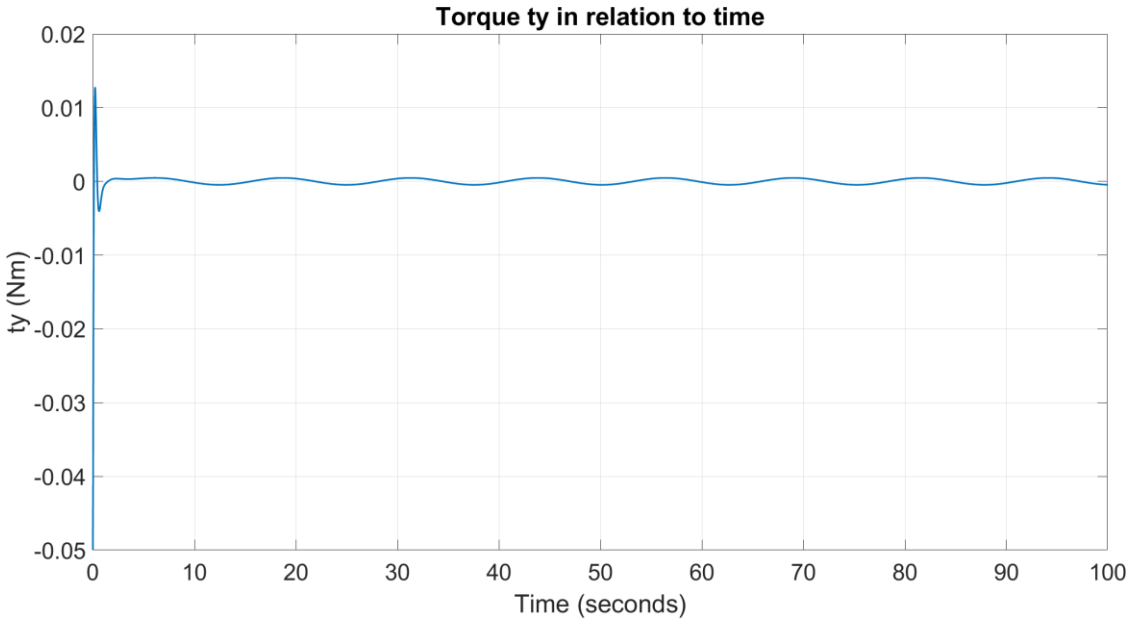


Figure 53. Torque t_y in relation to time for trajectory tracking (2nd set) using the PID design

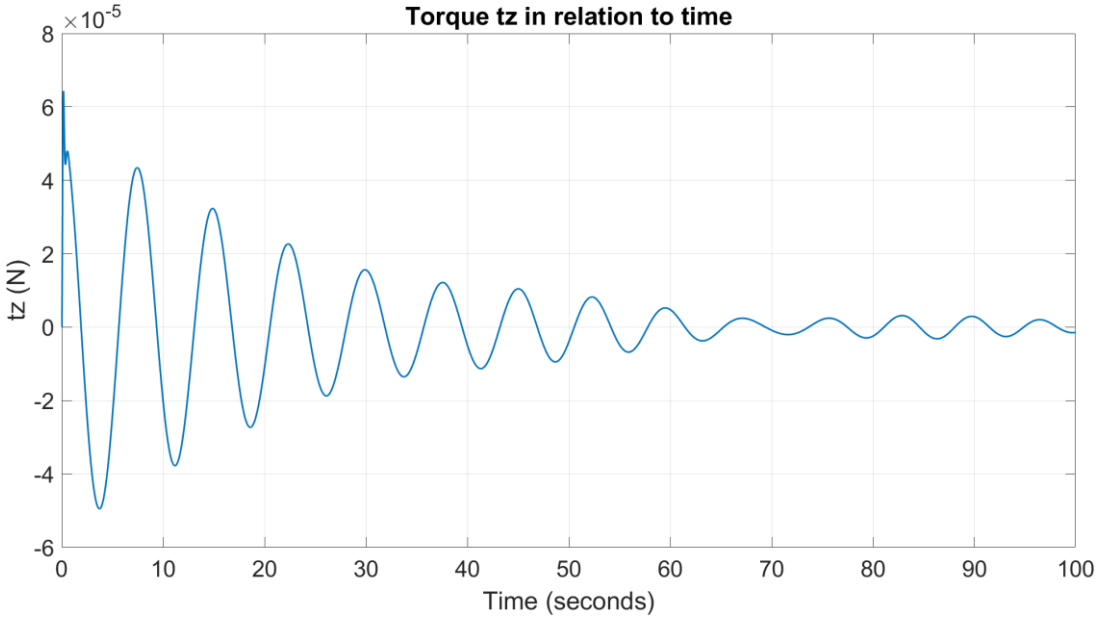


Figure 54. Torque t_z in relation to time for trajectory tracking (2nd set) using the PID design

As anticipated, the second trajectory set exhibits lower tracking performance compared to the first set, primarily due to its faster profile. More specifically, the PID control system appears to be struggling to accurately follow the trajectory in cases where there are rapid changes in orientation, as the system's response is slower than the rate at which the trajectory is changing. This is evident not

only in the simulation graphs for x, y, and z (Figure **45**, Figure **46**, Figure **47**), but also in the spiral graph (Figure **55**), where the deviations from the desired trajectory become more apparent:

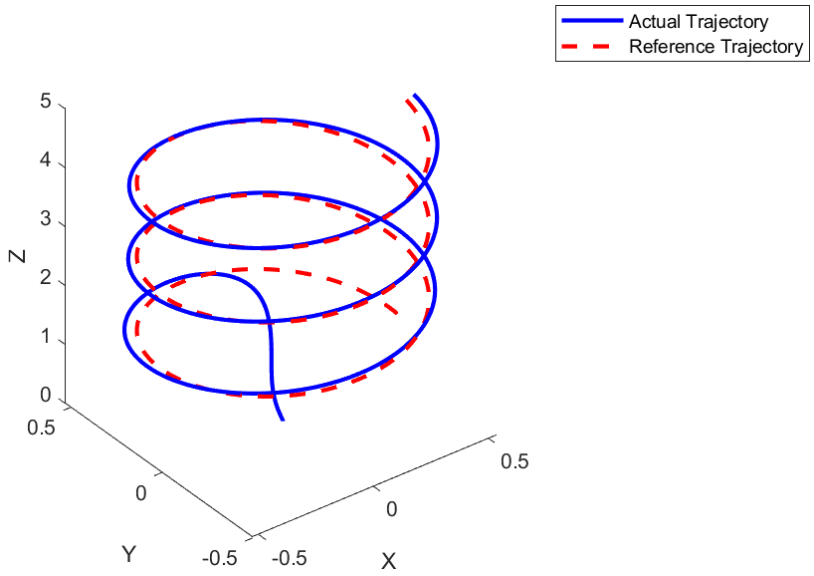


Figure 55. Actual trajectory compared to reference trajectory for the 2nd set using the PID design

Consequently, the calculated Mean Squared Error (MSE) values are also higher, reflecting the larger tracking errors between the reference trajectory and the actual response, as presented in the following table (Table **6**):

Table 6. The MSE values for the three axes – 2nd trajectory using the PID design

	Mean Squared Error (MSE)
x-axis trajectory	0.0068
y-axis trajectory	0.0011
z-axis trajectory	0.0197
Mean Euclidean Distance	0.0741

In summary, although the designed PID control system demonstrated acceptable results in the examined cases (step input, consecutive steps, path following for a slow and a fast set), it may not be the optimal choice for controlling the quadrotor system. This is graphically and quantitatively evident in the case of trajectory following, especially for the 2nd set examined, where the PID control system had a suboptimal performance in keeping up with the fast setpoint trajectory. While PID controllers excel in systems with fast dynamics, they struggle when dealing with high-frequency trajectories.

Moreover, the use of PID controllers for the quadrotor system presented many challenges, one of which is the difficulty of tuning. More specifically, tuning multiple PID controllers that control various aspects of the quadrotor, but have interlinked performances at the same time, is a challenging task. Additionally, tuning the values, following a method of trial-and-error, is susceptible to errors, which may lead to mediocre performance, instability or even inability to perform the simulation due to singularities. Furthermore, PID control is limited in its ability to accurately control a system, as it relies solely on the proportional, integral, and derivative values to adjust the system's response, which may not be enough for most dynamic systems.

Overall, nonlinear, and dynamic systems, such as the one examined, exhibit a complex behavior, making them a challenging control problem. Especially for the case of unexpected changes or disturbances in the process, classical control methods, like PID control, seem to fall short of ensuring reliable and accurate tracking. This is often called Virtual Control Loss (VCL) where the control system fails to effectively control the process, despite appearing functionally normal. To address these challenges, alternative and more advanced control methods are necessary for optimal results. Model Predictive Control (MPC) is one highly advanced control method that promises better performance and robustness for complex systems.

The next chapter will explore the application of MPC for controlling the quadrotor system. Its performance will be compared to that of the cascaded PID controller examined so far, providing insights into the relative strengths and weaknesses, and determining the best approach for the system.

CHAPTER 3: Model Predictive Control (MPC) for the Quadrotor system

This chapter aims to provide a comprehensive exploration of the Model Predictive Control (MPC) algorithm for a single quadcopter, which will act as the foundation for controlling a swarm of quadcopters. Firstly, a detailed overview of MPC is presented, emphasizing its operational mechanisms and its fundamental algorithmic principles. Then, the appropriate MPC algorithm is applied to a single quadrotor for the control of its position in the x, y, and z coordinates in 3D space, requiring precise control. Finally, in order to compare the performance of the MPC - based control strategy with that of PID control and determine the confidence that it can be used for a swarm of drones, simulations are conducted for trajectory tracking.

3.1 The Model Predictive Control (MPC)

Model Predictive Control, or MPC, is an intelligent control method that utilizes the model of a system in order to make predictions about its future outputs and ultimately compute the optimal input that will allow the plant to follow a desired reference. The history of MPC technology can be traced back to the early 1960s when Kalman explored the optimality of linear control systems. However, it was not until the late 1970s that notable applications of MPC emerged, such as MPHC or DMC (Dynamic Matrix Control) for the control of the transients of dynamic systems in chemical applications [22]. Over the years, multiple generations of MPC technologies have been developed, an evolution that has seen advancements in handling constraints and nonlinearities, improving stability, offering robust control performance, and providing more sophisticated control strategies.

Today, MPC is widely used in various fields, including process industries (e.g., for oil refining, chemicals or pharmaceuticals) [23], power systems (e.g. for power generation, transmission and energy storage) [24], automotive (e.g. in hybrid and electric vehicles for autonomous driving) [25], robotics and automation (e.g. for trajectory planning, motion control or robotic coordination) [26][10], aerospace and aviation (e.g. for navigation and control), building automation (e.g. for energy management) [27], and many more.

Furthermore, there is an abundance of different MPC variations, extensions and tailored algorithms developed for specific applications. Some of the most widespread MPC algorithms are Nonlinear MPC (NMPC) [13], Linear MPC (LMPC) (Adaptive or Gain-scheduled) [28], Dynamic Matrix Control (DMC) [29] or Quadratic Dynamic Matrix Control (QDMC), Robust MPC (RMPC) [30], Economic MPC (EMPC) [31], Explicit MPC [32] and more, each of which presents its benefits and limitations, as presented in Figure **56**; the selection of the algorithm depends on the specific control

objectives, system dynamics, requirements, constraints and the trade-off between performance and computational complexity.

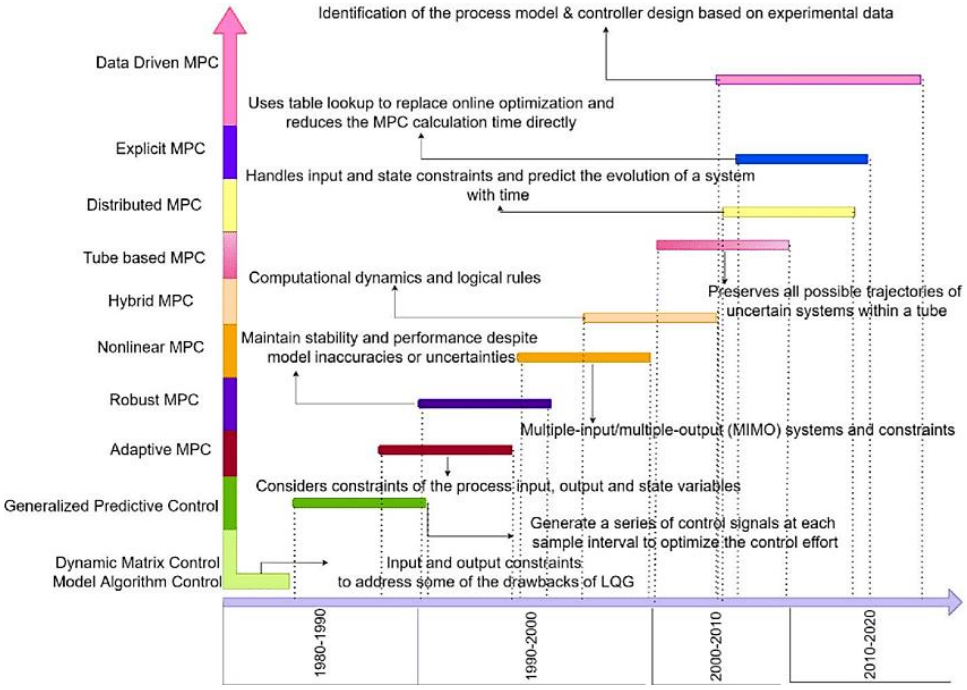


Figure 56. Model Predictive Control Algorithms over the years with their relative strengths

Most systems exhibit nonlinear or time – varying dynamics, making it crucial to employ an appropriate MPC control algorithm that can handle such characteristics. Nonlinear model predictive control is commonly utilized in systems featuring nonlinear constraints and a nonlinear cost function to enhance precision and accuracy. This approach is effective for systems characterized by nonlinear behavior, ensuring better performance and reliability. On the other hand, if the system can be adequately linearized, a linear MPC approach is recommended due to its simplicity and reduced computational cost.

More specifically, Linear MPC can be utilized not only for linear cost functions, but for nonlinear cost functions and linear dynamic systems, as well. LMPC can be further categorized into Adaptive MPC and Gain – Scheduled MPC. In the first case, the states and constraints remain constant across different operating conditions, and a linear model is derived at runtime, since the system is linearized at each operating point. Conversely, in the second case, each controller is independently executed since the dynamics and characteristics of the system change across different operating conditions. Furthermore, Adaptive LMPC, unlike traditional LMPC where the model parameters are

assumed to be known and constant, continuously adapts the parameters during the control process based on the measured data, making it a suitable control strategy for time – varying systems [22].

By carefully selecting the appropriate MPC variant based on the system's linearity and characteristics, more accurate and efficient control can be achieved. In general, MPC is such a widespread control method because it presents various distinct advantages over others, with the main ones being:

- *Handling MIMO (Multiple Input Multiple Output) systems*: in complex control problems there are usually multiple variables, interacting loops and interactions between the inputs and the outputs. Thus, there is the need of a multivariable controller that can handle such systems through a systematic framework; MPC is able of doing so, in contrast to simplified control methods, like PID controllers, in which the control loops operate as if the interactions are zero.
- *Handling constraints*: MPC can handle constraints on both the system states and the control inputs, which is especially crucial in process control applications, where satisfying constraints is affiliated with operating within safe limits. In other advanced control methods, like LQR, the constraints cannot be directly handled and may require additional techniques to accommodate them.
- *Handling Nonlinear systems*: by incorporating a nonlinear objective function and nonlinear constraints, MPC can effectively handle complex nonlinear systems, in contrast to other control methods that assume linearity.
- *Preview capability*: in MPC the preview capability is similar but more advanced than feedforward control; it takes into consideration the future system behavior – the predicted future states, to proactively respond to changes in the system, even in the case of disturbances, uncertainties, modelling errors or time delays. This improves the controller's overall performance, robustness, adaptability, and efficiency and wins over other control methods that do not account for future predictions.
- *Optimal control performance*: MPC solves an optimization problem at each time step, which allows a comprehensive control strategy that captures the desired control objectives, such as minimizing energy consumption, and allows time-varying control.

Model Predictive Control, despite its advantages, comes with certain disadvantages, as well. One significant drawback is its computational complexity, since MPC involves solving an optimization problem at each time step. This complexity increases with the increasing number of states in the system, the number of constraints, and the length of the design parameters of Prediction and Control

Horizons, that will be extensively analyzed in subchapter 3.2. Thus, the computational demands presented, which are also associated with the concern of memory usage, can pose challenges, especially if the implementation is done on embedded hardware with limited memory, or if the application requires that the algorithm be solved within a small time interval (in the order of milliseconds).

To mitigate these challenges, various techniques have been developed, such as algorithmic optimizations, sparse matrix representations, hardware acceleration, model order reduction techniques, setting a maximum number of iterations, as well as some special variations of MPC, like the aforementioned Explicit MPC. These techniques aim to improve the efficiency and feasibility of implementing MPC in resource-constrained environments. So, to implement Model Predictive Control practically and effectively, it is vital to carefully consider the computational requirements and limitations of the specific application.

3.2 The Design parameters of MPC

The MPC algorithm demands certain design parameters to be chosen, that highly affect the controller's performance, including responsiveness and robustness. Additionally, the computational burden and real-world implementation feasibility are also impacted by these parameter choices. Therefore, their selection necessitates careful consideration of the specific control objectives or requirements, while also striking a balance between control performance and computational efficiency. These design parameters, that will be discussed thoroughly are:

- Sample Time (T_s)
- Prediction Horizon (P or H_P)
- Control Horizon (M or H_C)
- Constraints
- Weights

The first crucial design parameter in MPC is *Sample Time* (T_s), which represents the rate at which the controller executes the control algorithm. In general, a smaller sample time allows more accuracy and responsiveness, as the controller is able to react fast enough to changes and disturbances. However, an exceedingly small T_s can result in excessive computational cost, thus potentially impeding real-world implementation and may also not manage to solve the optimization problem in the appropriate time. Contrary, a larger sample time may hinder the controller’s ability to react quickly enough, compromising the control performance. A commonly recommended guideline is to select a

sample time that accommodates 10-20 samples within the rise time of the open-loop response [33]; this can serve as a starting point, but the optimal choice of T_s is ultimately a multi-variable decision.

Another important design parameter is the *Prediction Horizon* (P or H_P) which determines the number of predicted future timesteps. Selecting an appropriate Prediction Horizon is vital, as it impacts the controller's ability to anticipate and respond to future system behavior. If it is too short, the controller may not be able to effectively capture the dynamics of the system, leading to inadequate precision. On the other hand, if H_p is excessively long, a significant portion of the computed predictions may be rendered irrelevant in the case of unexpected events or disturbances, thus resulting in unnecessary computational burden.

More specifically, for a faster trajectory with frequent changes, a smaller H_p is required as it needs to accurately anticipate changes in orientation. That being said, the choice of the Prediction Horizon can vary based on the specific application and requirements; recommendations such as 20-30 samples ensuring it covers the open-loop transient response of the system can serve as a starting point [33], but factors such as the dynamics of the system, the reference trajectory characteristics, and any constraints or limitations of the system are definitive in such choice.

Apart from the Prediction Horizon, the MPC algorithm is also strongly affected by another horizon called the *Control Horizon* (M or H_C), which refers to the number of control moves or adjustments considered until a particular time step M. Again, there should be careful consideration before choosing such horizon, as a value that is too small may not allow the controller to achieve the best possible maneuver, while by increasing it, the prediction accuracy is improved, as it can help account for variations in the trajectory, but at the cost of complexity. This is because each control move in the H_C is a free variable that needs to be computed by the optimizer, so the smaller the Control Horizon the fewer the design variables of the optimization problem and thus the fewer the computations.

The control horizon is always chosen to be less than or equal to the prediction horizon, because if the control horizon was bigger, then the optimizer would be instructed to determine control inputs of the system without knowing its state. This is contradictory and it would result to a certain and complete failure of the controller. So, by setting $H_C \leq H_P$ the control actions are concentrated within the range where the predictions are reliable, optimizing control performance within a well-defined time frame.

In the special case where the control horizon is equal to the prediction horizon, it is implied that the controller is making control moves by leveraging the maximum available predictive information, so it can determine a control action for each one of the predictions obtained. However, when it comes to linear models, it is observed that, only the initial control moves have a significant

impact on the system's response, while the influence of the subsequent control moves diminishes, because the future predictions become less accurate as the time horizon extends into the future.

This is proof that the choices of the two horizons in Model Predictive Control are intertwined, so selecting their appropriate values involves considering the diminishing returns associated with extending the horizons and ensuring that control actions are focused on reliable predictions within a relevant time span. A general rule of thumb is to set the Control Horizon to around 10%-20% of the Prediction Horizon, for a balance between obtaining accurate predictions and maintaining computational efficiency.

In the previous section, it was mentioned that one of the advantages of MPC is its ability to handle *constraints*. Constraints is an important factor that needs to be taken into consideration when employing Model Predictive Control, as they express the constraints that need to be satisfied when solving the optimization problem and so they allow the controller to account for the limitations and operational requirements of the system. Constraints can be classified into two main categories: hard constraints (e.g., safety limits), which cannot be violated, and soft constraints, which are usually associated with optimization criteria or performance objectives and can be violated under certain circumstances, without resulting in immediate failure of the system. The MPC algorithm considers both hard and soft constraints during the optimization process.

Finally, another fundamental point in the design of the MPC algorithm is the assignment of *weights*. MPC can have multiple objectives, which are typically defined through the objective function of the formulated optimization problem. These goals (for example tracking a setpoint as closely as possible, minimizing aggressive control moves, optimizing energy consumption etc.) can be conflicting or competing. To address this, the MPC algorithm allows for the weighting of different objectives to prioritize certain goals over others and finding a balance that aligns with the desired system behavior; the higher the weight the higher the significance of the specific objective. By adjusting the weights, system designers and control engineers can influence the controller's decision-making process according to the specific requirements and trade-offs of the application.

3.3 The Strategy of MPC

After having described the design parameters of the MPC algorithm, it is possible to analyze its strategy – how it works. So, model predictive control, or MPC, is an advanced feedback control mechanism that relies on a model of a system, hence its name, to predict its future behavior. Its objective is to select the best control action – the best input to the plant that drives it to reference.

More specifically, at each timestep k, MPC receives or estimates the current state of the plant and it then calculates the optimal sequences of control actions that minimize the cost over the finite time prediction horizon, i.e., for steps $k = t + 0, t + 1, , \dots, t + p$, where p is the number of upcoming points in the prediction horizon. This is achieved in a systematic way; by solving, at each timestep, a constrained optimization problem that relies on the model and depends on the current system state.

So, the optimization algorithm employed minimizes the objective - cost function, at each time step, over H_P, ultimately generating the optimal control input u(t); a vector with dimensions equal to the control horizon. The objective function (or cost function) is part of the modelling of the system and, in the general case, can be expressed as:

$$J = \sum_{i=1}^{H_P} w_e \cdot e_{k+i}^2 + \sum_{i=0}^{H_P-1} w_{\Delta u} \cdot \Delta u_{k+i}^2 \tag{3.1}$$

where H_P the prediction horizon, k the timestep at discrete time, e the error vector that represents the difference between the reference output and the actual produced output, Δu denotes the deviation of the control action between successive timesteps, and w_e , $w_{\Delta u}$ are the corresponding weights.

That being said, the predicted path with the smallest value of J gives the optimal solution to the problem. However, at the current time step k, MPC applies only the first step of the optimal sequence (i.e., the vector that has the same dimensions as the length of the control horizon), while the rest of them are used as initializations for the next iteration, at which the process is repeated. This is because the measurement that the controller gets at the next timestep $(k + 1)$ may be slightly different to what MPC predicted during the previous timestep, due to disturbances or unexpected events. So, implementing only the first control action provides some sort of adaptation to changing dynamics and measurements. [22]

Figure **57** presents the strategy followed by the MPC algorithm; a strategy that takes under consideration the past and current state of the system.

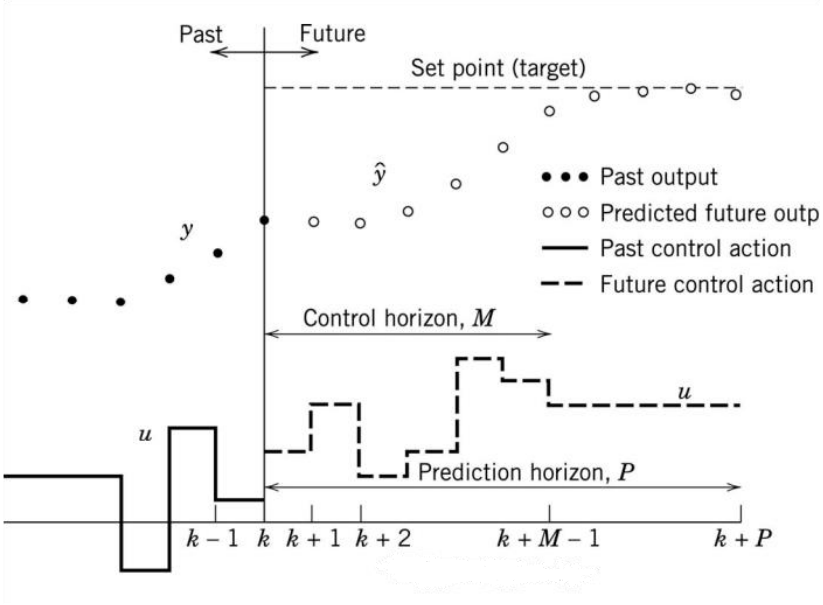


Figure 57. Visual representation of the MPC strategy

In a typical control circuit utilizing Model Predictive Control (MPC), like the one presented in Figure **58**, the reference signal is provided as input to the MPC controller, which comprises an optimizer, a prediction model (representing the system dynamics), constraints, and a cost function. The optimizer is the one responsible for solving the optimization problem to determine the optimal vector of control moves that minimizes the cost, while satisfying the constraints.

Then, only the first value of the optimal control trajectory is applied to the plant, which represents the physical system being controlled. The plant processes the control input (while also considering any disturbances), and produces an output vector, which serves as feedback to the MPC controller, allowing it to continuously update its predictions and adjust the control moves in response to changes in the system and external disturbances. This feedback loop enables the MPC controller to effectively track the desired references and maintain control over the plant's behavior.

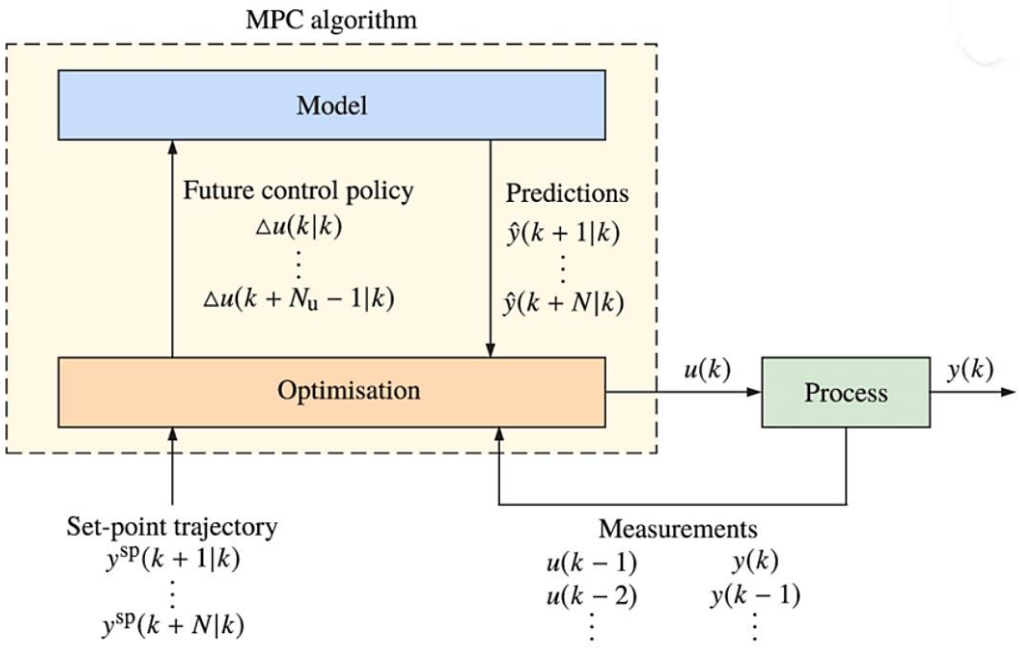


Figure 58. The typical design of an MPC circuit

3.4 Design of the MPC system

This sub-chapter focuses on the formulation of the problem for the quadrotor system within the context of MPC; it delves into the specific details of the employed MPC algorithm, the prediction model utilized, and the optimization problem that needs to be solved.

More specifically, in order to regulate the quadrotor system, three separate MPC controllers can be employed, one for position z, one for position x and one for position y. However, due to the interdependence or coupling between the variables x and y, an alternative approach is viable and expected to perform exceptionally well; the two MPCs corresponding to x and y can be integrated into a single unified MPC controller, utilized to address the control requirements of both these positions simultaneously.

The control system design utilizes both the positions (x, y, z) and their corresponding velocities as references, which are fed into the MPC controllers responsible for governing the quadrotor's motion. To ensure accurate and stable control, a closed-loop feedback system is implemented that incorporates both the relevant states of the quadrotor system and the already implemented (previous) control policy, which is used as input for the MPC controllers during the prediction phase. In this way, the control system can continuously monitor the quadrotor's performance and make necessary adjustments to maintain precise control.

The MPC controller for position x and y produces the reference angles θ (pitch angle) and φ (roll angle), respectively, that will be fed into separate PID controllers for angle regulation. It is noted that angle ψ (yaw angle) will also be controlled using a PID controller. By utilizing this cascaded control approach, as presented in Figure **59**, the quadrotor system is expected to introduce significant enhancements in terms of precision, stable regulation and adaptability of its position and orientation.

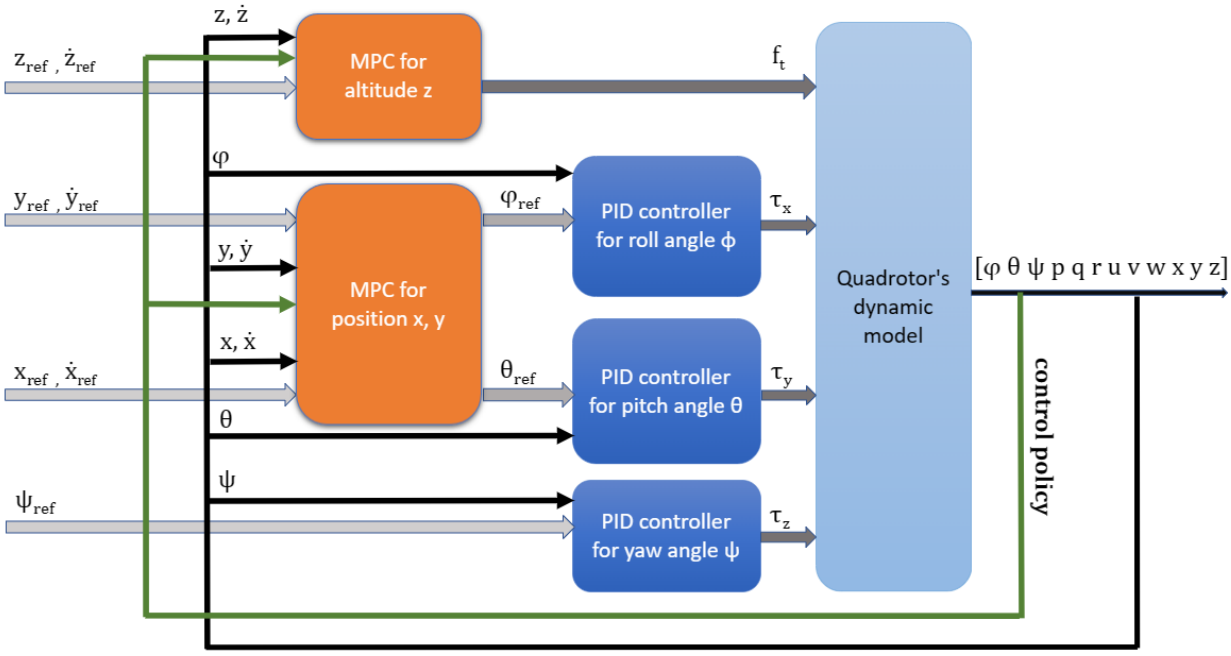


Figure 59. The cascaded control system utilizing MPC controllers and PID controllers

This design remains largely similar to the one presented in Chapter 2, as the fundamental characteristics of the quadrotor system have not changed; it is an underactuated system with six degrees of freedom, and so a cascaded design is still necessary to effectively control all its parameters. The main difference lies in the substitution of the outer – loop PID controllers with MPC controllers for position control in the x, y, and z directions. On the contrary, PID controllers are still employed for regulating the quadrotor's angles in the inner loop. Moreover, it should be highlighted that since MPC controllers have distinctive characteristics and operational principles than PID controllers, the tuning of the remaining inner-loop controllers is also expected to require adjustments to account for the updated specific dynamics.

Employing MPC controllers for position control in the x, y, and z directions while utilizing PID controllers for angle regulation is based on the characteristics and requirements of the quadrotor system. More specifically, MPC is a model-based control technique that excels in handling complex

dynamics, constraints, and predictive control objectives, so by utilizing it for position control the quadrotor system can benefit from precise and dynamic control of the quadrotor's position, while also accounting for factors such as trajectory tracking or obstacle evasion.

On the other hand, angles φ, θ and ψ are overly sensitive parameters that require accurate control, but they do not involve complex predictive dynamics, like position control does. In that case, PID controllers are well-suited for angle regulation due to their simplicity and effectiveness; MPC can also be employed for a more advanced and precise angle regulation, however, it is important to consider the trade-offs, as MPC controller is far more complex, and it certainly requires more computational resources in comparison to PIDs.

The choice to employ MPC for position control and PID controllers for angle regulation strikes a balance between computational efficiency and control accuracy. MPC handles the more complex and predictive aspects of position control, while PID controllers efficiently handle the sensitive and rapid angle regulation. This combination allows for a robust control strategy that ensures precise position control while maintaining stable and accurate orientation throughout the quadrotor's flight.

3.4..1 The prediction model (or system model)

In Model Predictive Control, the mathematical model plays a crucial role as the whole strategy is based on it to predict the system's future behavior over a finite time horizon and optimize the control inputs. That being said, the accuracy and fidelity of the employed model are determining the effectiveness of the MPC control. In Chapter 2, the mathematical model of the quadrotor system was derived, and building upon that foundation, it will be extended to obtain the prediction model necessary for implementing MPC.

More specifically, to derive the prediction model for the MPC strategy, the second derivatives of x, y and z will be examined. These can be derived from the set (1.18) of non-linear first order differential equations and Newton's law:

$$m\dot{\boldsymbol{v}} = \boldsymbol{R} \cdot \boldsymbol{f_B} = mg\hat{\boldsymbol{e}}_{\boldsymbol{z}} - f_t \boldsymbol{R} \cdot \hat{\boldsymbol{e}}_{\boldsymbol{3}} \tag{3.2}$$

and they can be expressed as follows:

$$\ddot{x} = -\frac{f_t}{m}[s_\phi s_\psi + c_\phi c_\psi s_\theta]$$
$$\ddot{y} = -\frac{f_t}{m}[c_\phi s_\psi s_\theta - c_\psi s_\phi]$$
$$\ddot{z} = g\ -\frac{f_t}{m}\ [c_\phi c_\theta)]$$

(3.3)

To obtain a suitable prediction model for the MPC strategy an approximation method like the Forward Euler method (Figure **60**) should be utilized; this numerical method can be used to numerically first-order differential equations, for linear systems and it can also be used to derive the corresponding discrete-time linear system. It discretizes the continuous-time system into discrete timesteps (k) using sampling time T_s. In the general case, assuming a first-order ODE:

$$\frac{dg}{dt} = f(t, g) \tag{3.3}$$

the approximation produces

$$\frac{g(t+T_s)-g(t)}{T_s} \approx f(t, g) \tag{3.4}$$

By discretizing equation (3.4) and assuming a small sample time $T_s = h = t_{n+1} - t_n$ the following form is derived:

$$\frac{g_{n+1}-g_n}{T_s} \approx f(t_n, g_n) \tag{3.5}$$

Moving forward, replacing $\approx$ with $=$ for convenience without forgetting the fact that the expression lurks an approximation error which grows with increasing T_s, equation (3.5) can be written as follows, when the future values are moved to the LHS and the past values to the RHS:

$$g_{n+1} = g_n + T_s \cdot f(t_n, g_n) \tag{3.6}$$

which represents a way to step forward in time. From equation (3.6) it is obvious that the smaller the sampling time the better the accuracy of the approximation model; the trade-off between computational burden as well as real-world implementation should be taken into consideration.

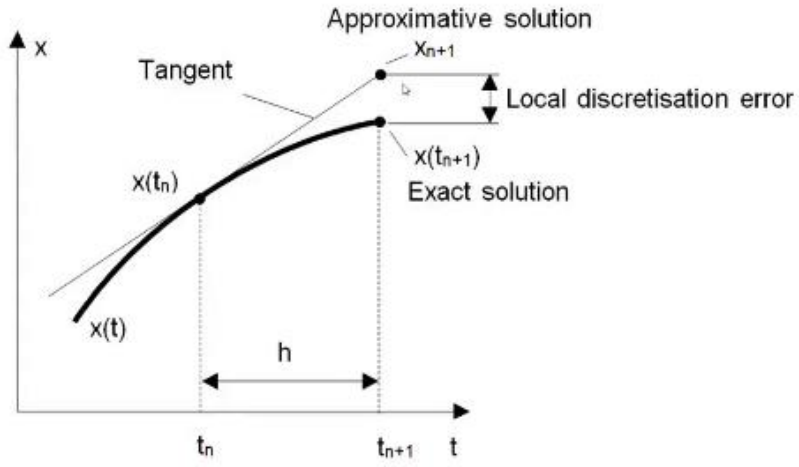


Figure 60. Visual representation of the Forward Euler method

So, by applying this specific first-order numerical procedure, the prediction model states that each new value of the solution is equal to the previous known value plus the product of the sampling time (T_s) and the values of its first-order differential equation. This method and the overall logic will be used to obtain the prediction model for the current diploma thesis as presented in detail for each of the MPC controllers in the following sections.

3.4..1.1 *The prediction model for the MPC related to altitude z*

As described in the set of equations (3.3), the second order derivative for the altitude is given by the expression $\ddot{z} = g\ -\frac{f_t}{m}\ [c_\phi c_\theta]$ that describes a time-varying complex system. Writing it in state space form produces the following two expressions:

$$z_1 = z\ \rightarrow\ \dot{z}_1 = z_2 \tag{3.7}$$

$$z_2 = \dot{z}\ \rightarrow\ \dot{z}_2 = g - \frac{f_t}{m}[c_\phi c_\theta] \tag{3.8}$$

Now, it is possible to apply the Forward Euler method which will produce the following expression:

$$z_1(k+1) = z_1(k) + T_s \cdot z_2(k) \tag{3.9}$$

$$z_2(k+1) = z_2(k) + T_s \cdot (g - \frac{f_t}{m}[cos\varphi(k)cos\theta(k)] \tag{3.10}$$

where $\dot{z}_1$ represents the altitude and $\dot{z}_2$ the velocity of the altitude. Using linear Algebra and matrix representation, the equations (3.9) and (3.10) can be written in the form $z(k+1) = A \cdot z(k) + B_k \cdot f_t(k) + E$, where A and E matrices are constant, B matrix is time-varying, and $z(k) = [z_1(k)\ \ z_2(k)]^T$ as follows:

$$z(k+1) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \cdot z(k) + \begin{bmatrix} 0 \\ \frac{-T_s}{m}[cos\varphi(k)cos\theta(k)] \end{bmatrix} \cdot f_t(k) + \begin{bmatrix} 0 \\ T_s g \end{bmatrix} \tag{3.11}$$

So, the prediction model (3.11) is the one used for the MPC of the altitude z and predicts the position z as well as the velocity of the position z. By breaking down the different terms in this equation a better analysis is achieved:

- The $A \cdot z(k) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \cdot z(k)$ term represents the state update equation, as it indicates that the next value of z is obtained by multiplying the previous, and so known, value of z with the transition matrix A. This matrix incorporates the dynamics of the system and relates the current state to the next state.
- The $B_k \cdot f_t(k) = \begin{bmatrix} 0 \\ \frac{-T_s}{m}[cos\varphi(k)cos\theta(k)] \end{bmatrix} \cdot f_t(k)$ term accounts for the impact of the control input on the next value of z. More specifically, it involves the total thrust generated by rotors, which corresponds to the first value of the control input u(x), as well as the angles φ and θ at the current timestep. By multiplying the control inputs with the appropriate scaling factors and incorporating the mass (m) of the system, this term influences the change in z between consecutive time steps.
- The $E = \begin{bmatrix} 0 \\ T_s g \end{bmatrix}$ term represents the effect of gravitational forces on the system's vertical position and thus on the next value of z, by incorporating the gravitational acceleration (g) and the sampling time (T_s).

Overall, the prediction model for the altitude of the quadrotor suggests that the next value of z at time step (k+1) is determined by combining the previous value z(k) with the influence of the system dynamics, control inputs, and gravitational forces. By iteratively applying this prediction model, the evolution of z over time can be estimated based on the current state and inputs.

3.4..1.2 *The prediction model for the MPC related to position x*

Following a similar logic for position x, the second-order derivative expression for x, $\ddot{x} = -\frac{f_t}{m}[s_\phi s_\psi + c_\phi c_\psi s_\theta]$ should be first written in state space form, as follows:

$$x_1 = x \rightarrow \dot{x}_1 = x_2 \tag{3.12}$$

$$x_2 = \dot{x} \rightarrow \dot{x}_2 = -\frac{f_t}{m}\left[s_\phi s_\psi + c_\phi c_\psi s_\theta\right] \tag{3.13}$$

and the Forward Euler method will result in the following two equations:

$$x_1(k+1) = x_1(k) + T_s \cdot x_2(k) \tag{3.14}$$

$$x_2(k+1) = x_2(k) + T_s \cdot \left(-\frac{f_t}{m}\left(sin\varphi(k)sin\psi(k) + cos\varphi(k)cos\psi(k)sin\theta(k)\right)\right) \tag{3.15}$$

By analyzing equation (3.15), it becomes evident that the control of position in the x axis is dependent on the angles for roll, pitch, and yaw, especially on the pitch angle, which is the one inner – loop design variable. So, since position x cannot be directly controlled without accounting for the pitch angle θ, the incorporation of what is called *virtual control* [34] becomes necessary, for developing a prediction model that controls the variable x. More specifically, in complex systems, there is often an interdependence between different variables and thus some may be controlled indirectly through others. In that situation, virtual control provides a flexible and efficient approach to address control challenges arising from interdependent variables, by facilitating the control process, regulating the desired variables indirectly and establishing a virtual relationship between them.

So, by employing virtual control, the controller generates control inputs – commands for the influencing variable, while the virtual control system translate those into corresponding actions – adjustments for the controlled variable; in the examined case, according to the cascaded control design in use, the influencing variable is the pitch angle θ, while the controlled variable is the position x. Assuming the following virtual control variable:

$$u_x = s_\phi s_\psi + c_\phi c_\psi s_\theta \rightarrow u_x(k) = sin\varphi(k)sin\psi(k) + cos\varphi(k)cos\psi(k)sin\theta(k) \tag{3.16}$$

the equation (3.15) gets the following form:

$$x_2(k+1) = x_2(k) + T_s \cdot \left(-\frac{f_t}{m}\right) \cdot u_x \tag{3.17}$$

Combining equations (1.14) and (1.17) using matrix representation in Linear Algebra, produces the following prediction model used for the Model Predictive Control of position x in the form $x(k+1) = A \cdot x(k) + B_k \cdot u_x$:

$$x(k+1) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \cdot x(k) + \begin{bmatrix} 0 \\ \frac{-T_s f_t(k)}{m} \end{bmatrix} \cdot u_x \tag{3.18}$$

where u_x is the virtual control variable of equation (3.16), matrix A is equal to $\begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$ and is constant, and matrix B_k is $\begin{bmatrix} 0 \\ \frac{-T_s f_t(k)}{m} \end{bmatrix}$ and depends on the total thrust generated by rotors at the current time step k. More specifically:

- The $A \cdot x(k) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \cdot x(k)$ term represents the state update equation for the position variable x. It indicates that the next value of x at time step (k+1) is obtained by multiplying the previous known value x(k) by the state transition matrix A, which incorporates the dynamics of the system and relates the current state to the next state.
- The term $B_k \cdot u_x = \begin{bmatrix} 0 \\ \frac{-T_s f_t(k)}{m} \end{bmatrix} \cdot u_x$ accounts for the impact of control inputs on the next value of x; it involves the virtual control input u_x and the total thrust generated control input f_t at time step k. By multiplying the first with the appropriate scaling factor and incorporating the system mass (m), this term influences the change in x between consecutive time steps. Moreover, the negative sign indicates that the force input acts in the opposite direction of the position change.

Overall, the prediction model suggests that the next value of x is determined by the previous value with the influence of the system dynamics and the virtual control variable; by iteratively applying this prediction model, the estimation of the evolution of x over time is obtained.

3.4..1.3 *The prediction model for the MPC related to position y*

The design of the prediction model for y follows a similar rationale to that of position x within the MPC framework. More precisely, like x, the position y cannot be directly controlled and requires regulation through the roll angle φ. Hence, the prediction model for y incorporates the principles of

virtual control to establish a relationship between y and the roll angle φ. So, starting with the second-order derivative expression $\ddot{y} = -\frac{f_t}{m}\left[c_\phi s_\psi s_\theta - c_\psi s_\phi\right]$, it will be written in state-space form:

$$y_1 = y \;\rightarrow\; \dot{y}_1 = y_2 \tag{3.19}$$

$$y_2 = \dot{y} \;\rightarrow\; \dot{y}_2 = -\frac{f_t}{m}\left[c_\phi s_\psi s_\theta - c_\psi s_\phi\right] \tag{3.20}$$

and the Forward Euler method will result in the following two equations:

$$y_1(k+1) = y_1(k) + T_s \cdot y_2(k) \tag{3.21}$$

$$y_2(k+1) = y_2(k) + T_s \cdot [-\frac{f_t}{m}(cos\varphi(k)sin\psi(k)sin\theta(k) - cos\psi(k)sin\varphi(k))] \tag{3.22}$$

Analogous to the virtual control variable u_x, by setting the virtual control variable u_y equal to:

$$u_y = c_\phi s_\psi s_\theta - c_\psi s_\phi \rightarrow u_y(k) = cos\varphi(k)sin\psi(k)sin\theta(k) - cos\psi(k)sin\varphi(k) \tag{3.23}$$

the equation (3.22) gets the following form:

$$y_2(k+1) = y_2(k) + T_s \cdot \left[-\frac{f_t}{m} \cdot u_y(k)\right] \tag{3.24}$$

Combining equations (3.21) and (3.24) using matrix representation in Linear Algebra, produces the following prediction model used for the Model Predictive Control of position y in the form $y(k+1) = A \cdot y(k) + B_k \cdot u_y$:

$$y(k+1) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \cdot y(k) + \begin{bmatrix} 0 \\ \frac{-T_s f_t(k)}{m} \end{bmatrix} \cdot u_x \tag{3.18}$$

where u_x is the virtual control variable of equation (3.23), matrix A is equal to $\begin{bmatrix}1 & T_s\\ 0 & 1\end{bmatrix}$ and it is constant, and matrix B_k is $\begin{bmatrix}0\\ \frac{-T_s f_t(k)}{m}\end{bmatrix}$ and dependent on the total thrust generated by rotors at the current time step k. More specifically:

- The $A \cdot y(k) = \begin{bmatrix}1 & T_s\\ 0 & 1\end{bmatrix} \cdot y(k)$ term represents the state update equation for the position variable y, indicating that the next value of y is obtained by multiplying the previous known value y(k) by the state transition matrix A.

  The term $B_k \cdot u_y = \begin{bmatrix}0\\ \frac{-T_s f_t(k)}{m}\end{bmatrix} \cdot u_y$ accounts for the impact of control inputs on the next value of y; it involves the virtual control input u_y and the total thrust generated control input f_t at time step k.

3.4..2 The optimization problem

The optimization problem in Model Predictive Control (MPC) is a principal component of the control strategy, which aims to determine the optimal control inputs over the finite prediction horizon. The objective of the optimization problem is to minimize a cost function, while satisfying operational constraints.

In that context, the formulation of the optimization problem typically involves three key elements: the design variables, which directly influence the system's behavior and are adjusted by the MPC algorithm to achieve the desired position tracking, the objective function, which reflects the performance objectives and priorities of the control system, comprising terms that quantify different aspects of control performance, and constraints, which represent limitations or bounds on the system's states, control inputs, or other variables. All of these will be analyzed in detail separately for the three MPCs over the next sections.

For this diploma thesis, the MATLAB function called fmincon will be used for solving the constrained nonlinear optimization or nonlinear programming problem formulated, through a Medium Scale Optimization algorithm called Sequential Quadratic Programming (SQP).

More specifically, the SQP algorithm, which presents good convergence properties, due to the use of second-order information and/or trust regions, and the ability to handle both equality and inequality constraints, works by iteratively adjusting the decision variables to minimize the objective function, while satisfying the constraints. More specifically, until a certain stopping criterion is met,

starting from an initial point guess, it solves a subproblem at each iteration, by performing a line search and by updating the estimate of the Hessian of the Lagrangian at each iteration of the algorithm.

So, once the optimization problem terminates, fmincon returns the optimal solution for the decision or control variables that minimize the objective function and respect the constraints.

3.4..2.1 *The optimization problem for the MPC related to the altitude z*

Regarding the control of altitude following the MPC strategy, the control input that directly affects z is the total thrust generated by rotors (f_t), as aforementioned. However, instead of choosing the absolute total thrust as the design variable to the optimization problem, it is beneficial to choose the change in total thrust, denoted as Δf_t, instead. By utilizing this specific design variable, the MPC algorithm can limit sudden and drastic variations in the motor's output, thus mitigating the risk of excessive stain or wear on the motors and ensuring the well-being and longevity of the real-life system.

More specifically, in the quadrotor's control system design it is always essential to consider the physical limitations and capabilities of the motors, as rapid or significant changes in thrust can lead to motor saturation, overheating or mechanical failure. By optimizing the change in thrust, the MPC controller is expected not only to respect the operational limits but also achieve better stability and performance of the control, as the MPC will be able to fine-tune the quadrotor's movements in the z direction.

Since Δf_t represents the change in thrust between consecutive time steps, or the relationship between the total thrust at the current timestep and the previous total thrust, it is given by the following equation:

$$\Delta f_t(k) = f_t(k) - f_t(k-1) \tag{3.19}$$

So, in essence, by optimizing the change in thrust, the MPC controller will indirectly also determine the optimal control sequence of the absolute thrust f_t, through (3.19), for each time step k. The first element of this optimal control sequence is chosen to be applied as the input to the system at the current time step, according to the MPC algorithm.

The objective function, by definition, is a function of the control variables and it quantifies the performance objectives of the control system. It consists of terms that capture desired behaviors, such as tracking a reference trajectory, minimizing control effort, or avoiding constraints violations, with the weights assigned to each term reflecting their relative importance. In the case of altitude z, the objective function J that needs to be minimized for $H_C \leq H_P$ is set to be the following:

$$J_z = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta f_t^T \cdot R \cdot \Delta f_t \quad (3.20)$$

where e is the error vector, $\dot{e}$ the derivative of the error, Δf_t the design variable, Q_1, Q_2 and R the penalty matrices. More specifically:

- e represents the error vector, which is the difference between the reference trajectory of altitude z and the predicted trajectory of z, over the prediction horizon, since $e(k+i),\ i = 1, \dots, hp$ is equal to $e(k+i) = r(k+i) - \hat{z}(k+i)$.
- The matrix Q_1 is a positive semi-definite weighting-penalty matrix with elements on the main diagonal, that influences how the controller responds to different state deviations by assigning weights to the different components of error vector e. By adjusting the values in the Q_1 matrix, the control of the importance or priority given to each component of the state error in the cost function is made possible.
- $\dot{e}$ represents the rate of change of the error vector e or, in other words, the velocity at which the system is deviating from the desired state trajectory. At this point, it is crucial to underscore the fact that the viable incorporation of trajectory derivatives arises solely from the foundational assumption made during the modeling of the quadrotor system. Specifically, it was presumed that the references, alongside their corresponding derivatives, are both boundedly limited and known a priori. By virtue of adopting this assumption within the model, the feasibility of leveraging trajectory derivatives in the trajectory following context becomes apparent.
- The matrix Q_2 is a positive semi-definite penalty matrix with elements on the main diagonal, that assigns weights to the components of the state error derivative vector. Similar to the Q_1 matrix, adjusting the values in Q_2 prioritize the components of the state error rate in the cost function.
- Δf_t is the design variable of the optimization problem and it represents the sequence of thrust changes over the control horizon H_C.

  The matrix R is a positive definite weighting matrix, with elements on the main diagonal, used to penalize deviations of the design variable between successive timesteps. Therefore, it determines the cost or penalty associated with the control effort and influences the control strategy to optimize the thrust changes, thus helping to balance the control effort and achieve smoother control actions.

The objective function combines these elements to form a composite cost function: the first term, $e^T \cdot Q_1 \cdot e$, penalizes state errors, aiming to minimize deviations between the predicted and desired state trajectories, the second term, $\dot{e}^T \cdot Q_2 \cdot \dot{e}$ dictates the MPC controller not only to consider the magnitude of the state error, but also to penalize its rapid changes, thus ensuring an accurate and a smooth tracking of the reference trajectory, and the third term, $\Delta f_t^T \cdot R \cdot \Delta f_t$, penalizes the change in thrust, encouraging control actions that are smoother and consistent with the system's capabilities.

Overall, the goal of the optimization problem is to minimize the value of the objective function J_z, thereby achieving a trade-off between tracking accuracy (minimizing state errors), tracking smoothness (minimizing the derivative if state errors) and control effort (minimizing thrust changes). The weights assigned in the penalty matrices allow for the customization of the control objectives, adapting the controller's behavior based on the specific requirements and performance goals of the quadrotor system.

Finally, after defining the design variables and the objective function, it is essential to establish constraints to form a complete optimization problem. The following constraints are the ones considered in the context of MPC for quadrotor control in the current diploma thesis.

The first constraint defines the allowable range-limits for the change in total thrust (Δf_t), which is the selected design variable, to prevent excessive or abrupt variations in the control input, that could lead to system instability of actuator limitations. The lower bound ensures a minimum change in thrust, while the upper bound restricts the maximum thrust change:

$$\Delta f_{t_{min}} \leq \Delta f_t(k+i) \leq \Delta f_{t_{max}}, \; i = 0, \dots, H_C \tag{3.21}$$

So, by optimizing the change in thrust, the MPC controller will indirectly also determine the optimal control sequence of the absolute thrust f_t, through (3.19), for each time step k. The first element of this optimal control sequence is chosen to be applied as the input to the system at the current time step, according to the MPC algorithm.

The second constraint defines the permissible range for the total thrust (f_t), thus ensuring that it remains within the operational limits of the quadrotor system preventing overloading the motors or operating outside the safe zone. The lower bound ensures a minimum thrust requirement, while the upper bound limits the maximum thrust that the system can generate:

$$f_{t_{min}} \leq f_t(k+i) \leq f_{t_{max}}, \; i = 0, \dots, H_P \tag{3.22}$$

The third constraint enforces the condition that the change in total thrust beyond the control horizon is zero. The controller plans for the thrust changes only within the control horizon, and beyond that horizon, the control action is held constant, thus simplifying the optimization problem, and ensuring a meaningful and feasible control sequence within the defined horizon:

$$\Delta f_t(k+i) = 0\ ,\ i = H_C + 1, \dots, H_P \tag{3.23}$$

By incorporating these constraints into the optimization problem, the MPC controller can generate control actions that satisfy both the desired control objectives (minimizing the cost function) and the system's operational constraints, resulting in safe and effective control of the quadrotor system. By combining equations (3.20), (3.21), (3.22) and (3.23), the complete optimization problem that will be used for the MPC of the altitude z is:

$$\begin{aligned} &\min_{\Delta f_t(k), \dots, \Delta f_t(k+H_C)} J = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta f_t^T \cdot R \cdot \Delta f_t \\ &subject\ to \qquad \Delta f_{t_{min}} \le \Delta f_t(k+i) \le \Delta f_{t_{max}}, for\ i = 0, .., H_C \\ &\qquad\qquad f_{t_{min}} \le f_t(k+i) \le f_{t_{max}}, for\ i = 0, .., H_P \\ &\qquad\qquad \Delta f_t(k+i) = 0, for\ i = H_C + 1, \dots, H_P \end{aligned} \tag{3.24}$$

3.4..2.2 *The optimization problem for the MPC related to the position x*

As aforementioned, based on the relationship between the position in the x-axis and the pitch angle θ, it is necessary to introduce the virtual control variable u_x to indirectly control x through angle regulation. In that context, similar to the approach taken for altitude control, the design variable is chosen to be the change in the virtual control variable, denoted as Δu_x, which is equal to:

$$\Delta u_x(k) = u_x(k) - u_x(k-1) \tag{3.25}$$

So, the MPC algorithm calculates the optimal sequence of Δu_x values, which are translated into the appropriate changes in the virtual control variable u_x, through equation (3.25). Moreover, since the MPC controller for the x position operates as the outer loop of the PID controller for θ angle regulation, it is responsible for providing the reference θ angle to the relevant PID controller. Consequently, the

expression for θ needs to be written in relation to the virtual control variable u_x, something that can be achieved by solving equation (3.16) for θ(k):

$$\theta(k) = \sin^{-1}\left(\frac{u_x(k)}{cos\varphi(k)}\right) \text{ , assuming that } \psi \cong 0$$

$$with\ cos\varphi(k) \neq 0 \ \rightarrow \ \varphi \neq \pm k \cdot \frac{\pi}{2} \ , k = 1, \dots, \mathbb{Z}$$

or

$$\theta(k) = \sin^{-1}\left(\frac{u_x(k) - sin\varphi(k) sin\psi(k)}{cos\varphi(k) cos\psi(k)}\right) \text{ , for } \psi \neq 0$$

$$with\ cos\varphi(k) \cdot cos\psi(k) \neq 0 \begin{cases} cos\varphi(k) \ \neq 0 \ \rightarrow \varphi \neq \pm k \cdot \frac{\pi}{2} \ , k = 1, \dots, \mathbb{Z} \\ cos\psi(k) \neq 0 \ \rightarrow \ \psi \neq \pm k \cdot \frac{\pi}{2} \ , k = 1, \dots, \mathbb{Z} \end{cases} \tag{3.26}$$

As it can be observed, equation (3.26) produces a constraint to ensure that the denominator is not equal to zero, since then it would result in an undefined solution. To avoid this constraint violation, in the general case where angle ψ is not equal to zero, it is necessary to ensure that $cos\varphi(k)cos\psi(k)$ is nonzero, which can be achieved by appropriately selecting the roll angle φ(k) and yaw angle ψ(k) within their valid ranges of $\pm k \cdot \frac{\pi}{2}$ during the control process. When these angles exceed 90 degrees, the system becomes highly unstable and may exhibit uncontrollable or unpredictable behavior, and that is why these constraints have already been imposed during the modelling of the quadrotor system.

Once the design variable has been defined, the next step is to construct the objective function for the MPC controller. Similar to the altitude control case, the objective function for the x position control can be formulated with three terms: the first term penalizes the error between the desired x position and the predicted x position, ensuring that the MPC controller drives the system with accuracy, the second term penalizes the derivative of the error, aiming to achieve smooth and stable control, by anticipating changes in the error, and the third term is designed to encourage the system to minimize unnecessary movements or control actions, thus promoting energy efficiency and reducing unnecessary adjustments.

$$J_x = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta u_x^T \cdot R \cdot \Delta u_x \tag{3.27}$$

The specific weights assigned to each term in the objective function depend on the control objectives and priorities of the system and they determine the relative importance of each term in shaping the control behavior.

The constraints for the MPC controller in the x position follow a similar logic as with the altitude control. So, firstly, there are upper and lower bounds set for the design variable, the virtual control variable (Δux), that limit the amount of control input that can be applied at each time step, preventing excessive or impractical control actions. Similarly, there are also upper and lower bounds defined for the virtual control variable, that restrict the magnitude of the control input, preventing control inputs from becoming too large or too small, which can lead to instability or insufficient control.

$$\Delta u_{x_{min}} \leq \Delta u_x(k+i) \leq \Delta u_{x_{max}}, \; i = 0, \dots, H_C \tag{3.28}$$

$$u_{x_{min}} \leq u_x(k+i) \leq u_{x_{max}}, \; i = 0, \dots, H_P \tag{3.29}$$

Finally, again, there is the constraint that enforces the condition that the change in total thrust beyond the control horizon is zero, thus helping guarantee the effectiveness and reliability of the MPC control strategy:

$$\Delta u_x(k+i) = 0, i = H_C + 1, \dots, H_P \tag{3.30}$$

So, by combining the objective function, with the constraints, the total optimization problem for the control of position in the x-axis is formulated as follows:

$$\begin{aligned} &\min_{\Delta u_x(k), \dots, \Delta u_x(k+H_C)} J_x = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta u_x^T \cdot R \cdot \Delta u_x \\ &subject\ to \quad \Delta u_{x_{min}} \leq \Delta u_x(k+i) \leq \Delta u_{x_{max}}, \quad i = 0, \dots, H_C \\ &\qquad u_{x_{min}} \leq u_x(k+i) \leq u_{x_{max}}, \quad i = 0, \dots, H_P \\ &\qquad \Delta u_x(k+i) = 0, \quad i = H_C + 1, \dots, H_P \end{aligned} \tag{3.31}$$

3.4..2.3 *The optimization problem for the MPC related to the position y*

The optimization problem for controlling the y position of the quadrotor can be formulated in a manner completely analogous to the one for the x position. That being said, the design variable is chosen to be the change in the virtual control variable, denoted as Δu_y, for indirect control of the y position through the roll angle φ:

$$\Delta u_y(k) = u_y(k) - u_y(k-1) \tag{3.32}$$

So, the MPC algorithm calculates the optimal sequence of Δu_y values, which are translated into the appropriate changes in the virtual control variable u_y, through equation (3.32), and since the MPC controller for the y position operates as the outer loop of the PID controller for φ angle regulation, it is responsible for providing the reference φ angle to the relevant PID controller.

After performing the necessary calculations to determine the expression for the reference angle θ, denoted as expression (3.36), it becomes feasible to incorporate this value into equation (3.23). By doing so, the expression that characterizes the reference angle φ is obtained, taking into account a non-zero angle ψ. Conversely, if ψ is chosen to be assumed as zero, primarily for the sake of practicality and simulation purposes, it is possible to directly acquire the expression for the reference angle φ from equation (3.23).

$$\varphi(k) = \sin^{-1}(u_y) \text{ , assuming that } \psi \cong 0$$

or

$$\varphi(k) = -\sin^{-1}(u_y(k) \cdot \cos\psi(k) - \sin\psi(k) \cdot u_x(k)) \text{ , for } \psi \neq 0 \tag{3.33}$$

Similarly to x, the objective function is constructed to minimize a combination of error terms, including the error in the y position, the derivative of the error, which is the velocity in the y axis, and a term that encourages smooth and efficient control actions. These terms are weighted by appropriate matrices to reflect the importance and desired behavior of each component.

$$J_y = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta u_x^T \cdot R \cdot \Delta u_x \tag{3.34}$$

Moreover, analogous constraints to those for x are imposed on the design variable Δu_x and its absolute value u_x to ensure that the control input remains within permissible bounds.

$$\Delta u_{y_{min}} \leq \Delta u_y(k+i) \leq \Delta u_{y_{max}}, \; i = 0, \dots, H_C \tag{3.35}$$

$$u_{y_{min}} \leq u_y(k+i) \leq u_{y_{max}}, \; i = 0, \dots, H_P \tag{3.36}$$

Also, there is the constraint that enforces the condition that the change in the control variable beyond the control horizon is zero, for adherence to algorithm-specific requirements:

$$\Delta u_y(k+i) = 0, \; i = H_C + 1, \dots, H_P \tag{3.37}$$

So, by combining the objective function, with the constraints, the total optimization problem for the control of position in the x-axis is formulated as follows:

$$\begin{aligned} &\min_{\Delta u_y(k), \dots, \Delta u_y(k+H_C)} J_y = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta u_y^T \cdot R \cdot \Delta u_y \\ &subject\ to \qquad \Delta u_{y_{min}} \leq \Delta u_y(k+i) \leq \Delta u_{y_{max}}, \qquad i = 0, \dots, H_C \\ &\qquad\qquad u_{y_{min}} \leq u_y(k+i) \leq u_{y_{max}}, \qquad i = 0, \dots, H_P \\ &\qquad\qquad \Delta u_y(k+i) = 0, \qquad i = H_C + 1, \dots, H_P \end{aligned} \tag{3.38}$$

3.4..3 The terminal constraint

In control systems theory, terminal constraint refers to a condition-constraint imposed on the system's state variables at the final time instant, also known as the terminal time, to specify a desired behavior at the end of a given time period. It adds an additional requirement that the system must satisfy at the final time, in addition to the usual constraints on the state and control variables throughout the time horizon.

In MPC, the terminal constraint plays a crucial role as it is an effective way to achieve closed-loop stability; it acts like an attractor towards the desired reference trajectory, ensuring that the MPC drives the system towards the reference setpoint and maintains it there. The assurance of stability

through the terminal constraint in the context of control systems can be attributed to its pivotal role in fostering convergence towards desired trajectories, while minimizing tracking errors [35].

More specifically, the terminal constraint ensures that the system matches the desired state at the final step of the prediction horizon, which, consequently, implies that the tracking error *e* at the terminal time instant is zero. Generally speaking, in control systems, the goal is to minimize the tracking error so a terminal constraint that enforces the aforementioned condition makes it possible to ensure that the error is minimized; as time approaches infinity ($t \to \infty$), the error asymptotically approaches zero, indicating convergence to the desired trajectory.

That being said, according to Lyapunov's theorem, when the error approaches zero, as it is the case with the employment of terminal constraint, the objective function is minimized, thus resulting in a decreasing function. Additionally, if the objective function is quadratic and positively defined, as it is the case for all of the objective functions in the examined quadrotor system, this allows for the establishment of a positive-definite objective function ($J \geq 0$). So, a positive-definite and increasing function has its lower bound at zero and thus the system adheres to the principles of asymptotic stability [36].

In the context of trajectory following in MPC, and as far as the quadrotor is concerned, three terminal constraints can be imposed, one for each of the three 3D coordinates, and they should be included in the optimization problem formulation for every timestep k within the MPC algorithm:

- Terminal constraint for altitude z: $\hat{z}(k + H_P) = z_r(k + H_P)$
- Terminal constraint for position x: $\hat{x}(k + H_P) = x_r(k + H_P)$
- Terminal constraint for position y: $\hat{y}(k + H_P) = y_r(k + H_P)$

(3.39)

where $\hat{z}$, $\hat{x}$, $\hat{y}$ represent the predicted states of the system at the end of the prediction horizon and z_r , x_r , y_r the desired system states at the same time step.

As it can be observed from (3.39), the satisfaction of the terminal constraint in MPC relies on ensuring an adequate prediction horizon (H_P) that allows the system sufficient time – an extended time window, to converge towards the reference trajectory. If the prediction horizon is too short, it restricts the system's ability to adjust its trajectory and may impede convergence to the desired state. In such cases, the system may not have ample time to respond to disturbances, adapt its control inputs, and adequately track the reference trajectory. Consequently, the terminal constraint may remain unsatisfied.

In contrast, a longer prediction horizon allows for a more comprehensive planning and adjustment of control actions, granting the system sufficient time to approach and align with the reference trajectory. By providing this extended duration, the system can effectively navigate towards the desired state and satisfy the terminal constraint within the specified time frame. Altogether, the complete optimization problems for x, y and z are presented in the set of equations (3.40):

$$\begin{aligned}
&\min_{\Delta f_t(k),\dots,\Delta f_t(k+H_C)} J = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta f_t^T \cdot R \cdot \Delta f_t \\
&subject\ to \quad \Delta f_{t_{min}} \le \Delta f_t(k+i) \le \Delta f_{t_{max}}, for\ i = 0,..,H_C \\
&\quad f_{t_{min}} \le f_t(k+i) \le f_{t_{max}}, for\ i = 0,..,H_P \\
&\quad \Delta f_t(k+i) = 0, for\ i = H_C+1,\dots,H_P \\
&\quad \hat{z}(k+H_P) = z_r(k+H_P)
\end{aligned}$$

$$\begin{aligned}
&\min_{\Delta u_x(k),\dots,\Delta u_x(k+H_C)} J_x = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta u_x^T \cdot R \cdot \Delta u_x \\
&subject\ to \quad \Delta u_{x_{min}} \le \Delta u_x(k+i) \le \Delta u_{x_{max}}, \quad i = 0,\dots,H_C \\
&\quad u_{x_{min}} \le u_x(k+i) \le u_{x_{max}}, \quad i = 0,\dots,H_P \\
&\quad \Delta u_x(k+i) = 0, \quad i = H_C+1,\dots,H_P \\
&\quad \hat{x}(k+H_P) = x_r(k+H_P)
\end{aligned}$$

$$\begin{aligned}
&\min_{\Delta u_y(k),\dots,\Delta u_y(k+H_C)} J_y = e^T \cdot Q_1 \cdot e + \dot{e}^T \cdot Q_2 \cdot \dot{e} + \Delta u_y^T \cdot R \cdot \Delta u_y \\
&subject\ to \quad \Delta u_{y_{min}} \le \Delta u_y(k+i) \le \Delta u_{y_{max}}, \quad i = 0,\dots,H_C \\
&\quad u_{y_{min}} \le u_y(k+i) \le u_{y_{max}}, \quad i = 0,\dots,H_P \\
&\quad \Delta u_y(k+i) = 0, \quad i = H_C+1,\dots,H_P \\
&\quad \hat{y}(k+H_P) = y_r(k+H_P)
\end{aligned}$$

(3.40)

However, it is important to remember that the utilization of a single MPC controller for both x and y axes involves formulating an optimization problem that combines the individual optimization problems for x and y and imposes a common prediction and control horizon.

Within this context, to implement the code of the MPC algorithm, it is possible to use the variable *u* in order to encompass both control variables u_x and u_y, for a unified problem. However, in

order to facilitate effective tuning, distinct bounds are assigned to the control variables for both x and y and penalties specific to each axis.

In this integrated framework, the individual objective functions for the x and y axes are combined into a single optimization problem through summation, as presented in set of equations (3.41). This means that the optimization process now seeks to optimize the combined objective, which considers both the x-axis and y-axis control objectives. By summing the separate objective functions, the coordination and balance of control inputs between the x and y axes are prioritized, thus providing a coherent and coordinated movement in both directions.

$$\begin{gathered}\min_{\Delta u(k),\dots,\Delta u(k+H_C)} \\ J = e_x^T \cdot Q_{1x} \cdot e_x + \dot{e}_x^T \cdot Q_{2x} \cdot \dot{e}_x + \Delta u_x^T \cdot R_x \cdot \Delta u_x + e_y^T \cdot Q_{1y} \cdot e_y + \dot{e}_y^T \cdot Q_{2y} \cdot \dot{e}_y + \Delta u_y^T \cdot R_y \cdot \Delta u_y \\ subject\ to \qquad \Delta u_{x_{min}}, \Delta u_{y_{min}} \le \Delta u(k+i) \le \Delta u_{x_{max}}, \Delta u_{y_{max}}, \qquad i = 0,\dots,H_C \\ u_{x_{min}}, u_{y_{min}} \le u(k+i) \le u_{x_{max}}, u_{y_{max}}, \qquad i = 0,\dots,H_P \\ \Delta u(k+i) = 0\,, \qquad i = H_C+1,\dots,H_P \\ \hat{x}(k+H_P) = x_r(k+H_P), \hat{y}(k+H_P) = y_r(k+H_P)\end{gathered} \tag{3.41}$$

3.5 Simulation analysis and results

For comparative analysis purposes, the same trajectories explored in Chapter 2., using only PIDs, will be re-evaluated with the new MPC control scheme design. It is anticipated that the predictive capability of MPC will yield superior results, enabling the precise tracking of even high-speed trajectories. The control and prediction horizons are presented in Table 7 :

Table 7. MPC parameters for z and x-y for a single quadcopter

MPC horizons	
H_c for z	3
H_p for z	30
H_c for x-y	6
H_p for x-y	20

The selection of the prediction and control horizons, as well as the decision to employ distinct horizons for altitude z and positions x-y, was made under careful consideration of the trajectory complexities in 3D space, as well as the trade-off between performance and computational burden.

More specifically, for the z-axis, the examined trajectory offers greater ease of following, as it doesn't involve changes in orientation. Consequently, a smaller control horizon was deemed sufficient, thus reducing computational cost. Conversely, the x-y trajectory is far more complex, necessitating a larger control horizon to enable responsiveness, maneuvers and improved control performance. However, since the control horizon impacts computational costs, a value of 6 was considered adequate to account for these factors without excessive computational burden.

Moreover, regarding Prediction Horizons, for z-axis trajectory, a substantial prediction horizon was chosen to facilitate the anticipation of future states, given its inherent simplicity. In contrast, the x-y trajectory necessitates a slightly shorter Prediction Horizon to ensure the system's ability to promptly adapt to changes in orientation and position. Nevertheless, it's important to acknowledge that in the general scenario, identical values are chosen for the prediction horizon in both the z-axis and x-y trajectories.

In summary, the selection of the specific values for control and prediction Horizons presented was a product of thorough trial and error, while also bearing in mind a pragmatic approach that optimizes the computational resources and addressing the complexities of the trajectory.

Table 8. The PID values for angle regulation for the MPC design for a single quadcopter

PID values			
	Proportional gain k_P	Integral gain k_I	Derivative gain k_D
PID controller for angle phi	0.15	0.0001	0.3
PID controller for angle theta	0.05	0.0001	0.25
PID controller for angle psi	0.005	0.0001	0.01

The choice of the PID values for angle regulation, as presented in Table **8**, follows a similar methodology to that employed in Chapter 2. More specifically, the process begun by selecting the proportional gain aimed to minimize the error between the reference angle generated by the MPC for positions x-y and the actual angle of the PID control. This choice necessitated a small value, given the sensitivity of angular control. Then, a larger derivative value was chosen to mitigate oscillations in the

system, and finally, a very small value for the integral gain was adopted to ensure that the steady state error converges to zero, while preventing the possibility of integral windup.

Table 9. The selected penalty matrices for the MPC design for a single quadcopter

Peanalty matrices			
	Error penalty matrix Q_1	Velocity penalty matrix Q_2	Design variable penalty matrix R
altitude z	$diag(1 \cdot ones(hp, 1))$	$diag(0.05 \cdot ones(hp, 1))$	$diag(0.5 \cdot ones(hc, 1))$
position x	$diag(1 \cdot ones(hp_{xy}, 1))$	$diag(0.01 \cdot ones(hp_{xy}, 1))$	$diag(0.75 \cdot ones(hc_{xy}, 1))$
position y	$diag(1 \cdot ones(hp_{xy}, 1))$	$diag(0.01 \cdot ones(hp_{xy}, 1))$	$diag(0.5 \cdot ones(hc_{xy}, 1))$

As described in sub-chapter 3.4.2., the penalty matrices, used to assign weights to the terms of the objective functions, are positive semi-definite matrices with elements along the main diagonal and dimensions relevant to the specified horizons. The selection of their values, as presented in Table **9**, is contingent upon the objectives established by the designer, as well as the specific attributes of the system.

More specifically, through a process of trial and error, the error penalty matrix denoted as Q_1, is chosen to have a relatively high value – being the largest among the matrices. This choice prioritizes the minimization of errors between the actual and the reference trajectories. Conversely, the velocity (the derivative of the error) penalty matrix Q_2 is selected to have a smaller value, allowing for the correction of initial errors stemming from the quadrotor's starting conditions.

Finally, the design variable penalty matrix, R, contains relatively small values in order to protect the system from exhibiting overly abrupt behavior. Consequently, the biggest value is assigned to x-position, which is associated to the particularly sensitive angle theta.

It is also important to highlight the fact that, penalty matrices are commonly assigned varying values along the diagonal, with larger values typically placed towards the end. This practice is related to the assumption that the values produced towards the end of the horizon tend to be less reliable or more prone to uncertainty. Consequently, by assigning to them larger penalty values, the control system penalizes them more to maintain better control or tracking performance. However, in this diploma thesis, for ease of tuning, a uniform value was applied throughout the entire diagonal.

Table 10. The minimum - maximum values for the variables for the MPC design for a single quad

Min and max values for the variables		
	Minimum value	Maximum value
f_t [N]	-20	20
u_x [rad]	-0.18	0.18
u_y [rad]	-0.2	0.2
df_t [N]	-10	10
du_x [rad]	-0.05	0.05
du_y [rad]	-0.06	0.06

The selection of values for the control variables $(\Delta f_t, \Delta u_x, \Delta u_y)$ and the variables from which they stem (f_t, u_x, u_y), as presented in Table **10**, involved a meticulous process of trial and error coupled with considerations of real-life implementation. In particular, the values were deliberately chosen within a narrow range since the quadrotor is a highly sensitive system that is prone to abrupt behavior. Moreover, for the sake of simplifying the tuning process, the decision was made to set the minimum and maximum values equal and opposite.

3.5..1 Simulation for the 1st trajectory set

The simulation graphs for the slow trajectory set and for a time span of half a period (approximately 1047 timesteps) for visual clarity are presented below:

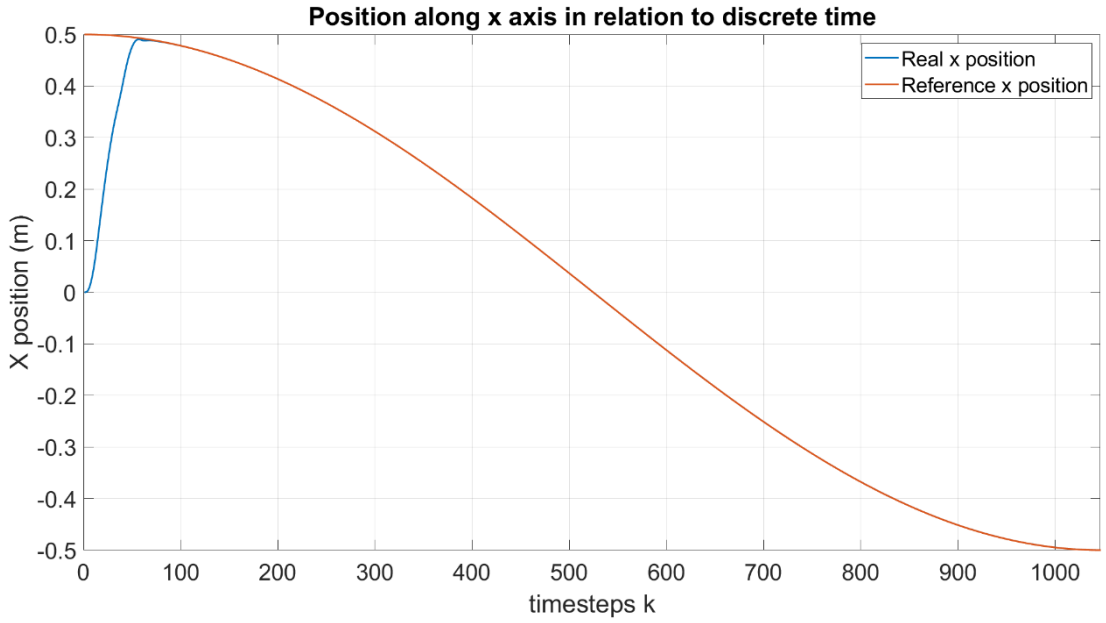


Figure 61. Position x and x_{ref} in relation to time for trajectory tracking (1st set) using the MPC design

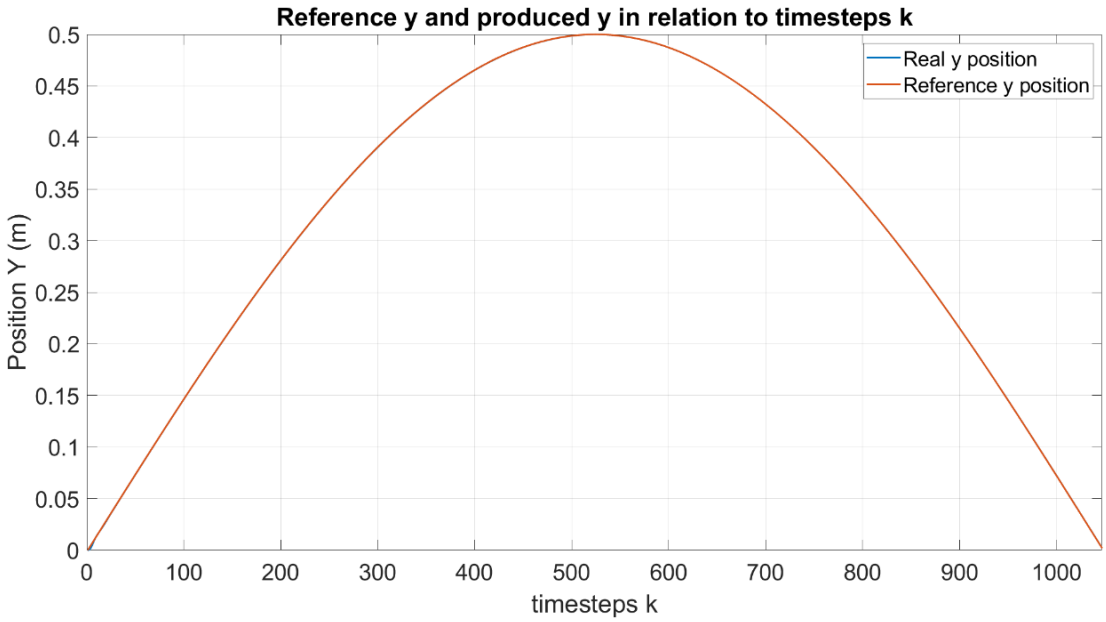


Figure 62. Position y and y_{ref} in relation to time for trajectory tracking (1st set) using the MPC design

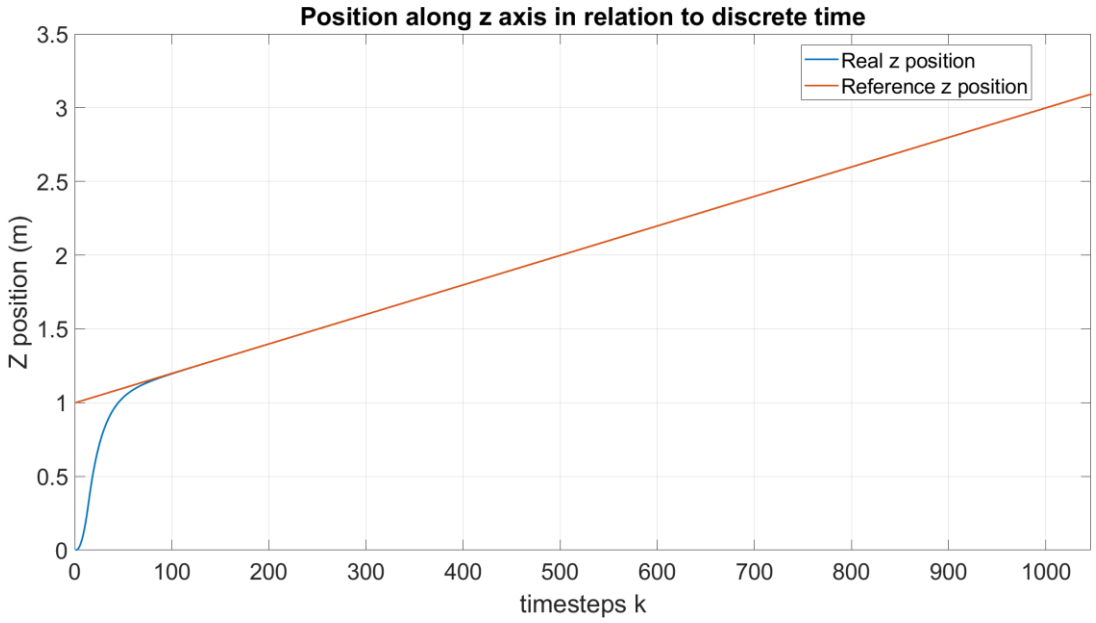


Figure 63. Position z and z_{ref} in relation to time for trajectory tracking (1st set) using the MPC design

As depicted in Figure **61**, Figure **62** and Figure **63**, the quadrotor demonstrates an exceptional ability to closely follow the prescribed x, y, and z trajectories. More specifically, initially, the quadrotor swiftly converges to the target trajectory, displaying a rapid 'catching' capability. Then, once aligned with the desired path, it exhibits remarkable stability, with a high level of precision. Notably, the y-position, characterized by a cosine trajectory, exhibits the best tracking performance, which is attributed to the favorable alignment of the initial condition with the trajectory. So, overall, these figures highlight the ability of the MPC contorllers to achieve great performace in terms of minimizing the tracking errors along all directions.

In addition to the precise tracking of positions, it is worth highlighting the behavior of the angles (Figure **64**, Figure **65** and Figure **66**), which are generated as references by the MPC controller for x (related to angle theta) and y (related to angle phi). The quadrotor exhibits a notably sharp response in regulating these angles, presenting rapid alignment with the desired orientations, and the same is true for the control inputs (Figure **67**, Figure **68**, Figure **69** and Figure **70**).

More specifically, during the initial stages of the trajectory tracking, both the angles and the associated control inputs exhibit sharp and dynamic behavior, but as the quadrotor approaches its desired path, there is a transition to a stable state characterized by minimal variations around zero.

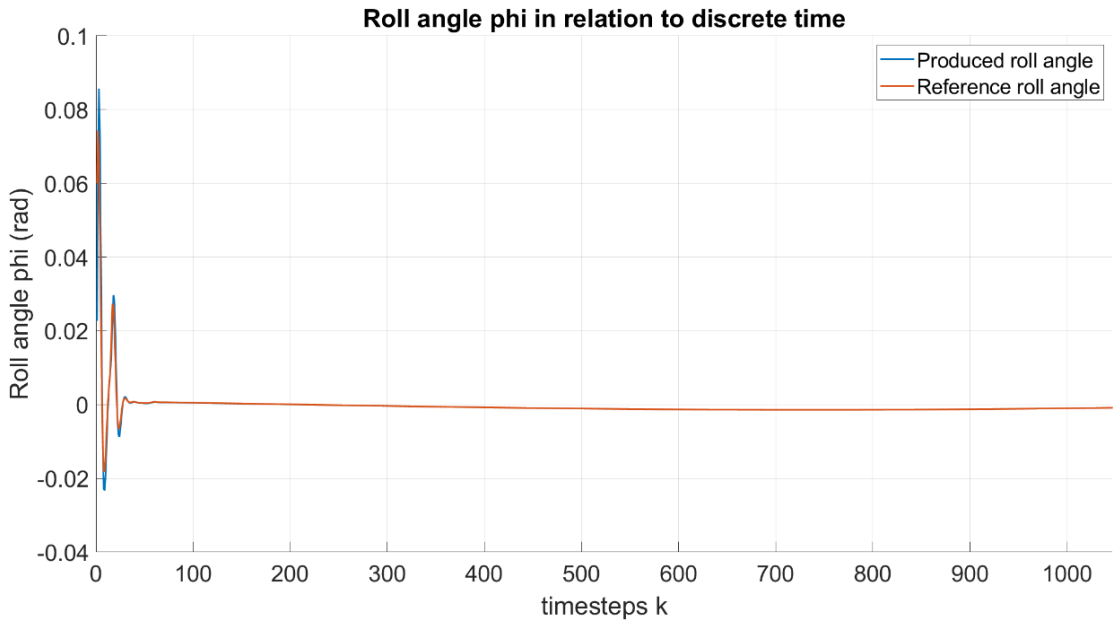


Figure 64. Roll angle φ in relation to time for trajectory tracking (1st set) using the MPC design

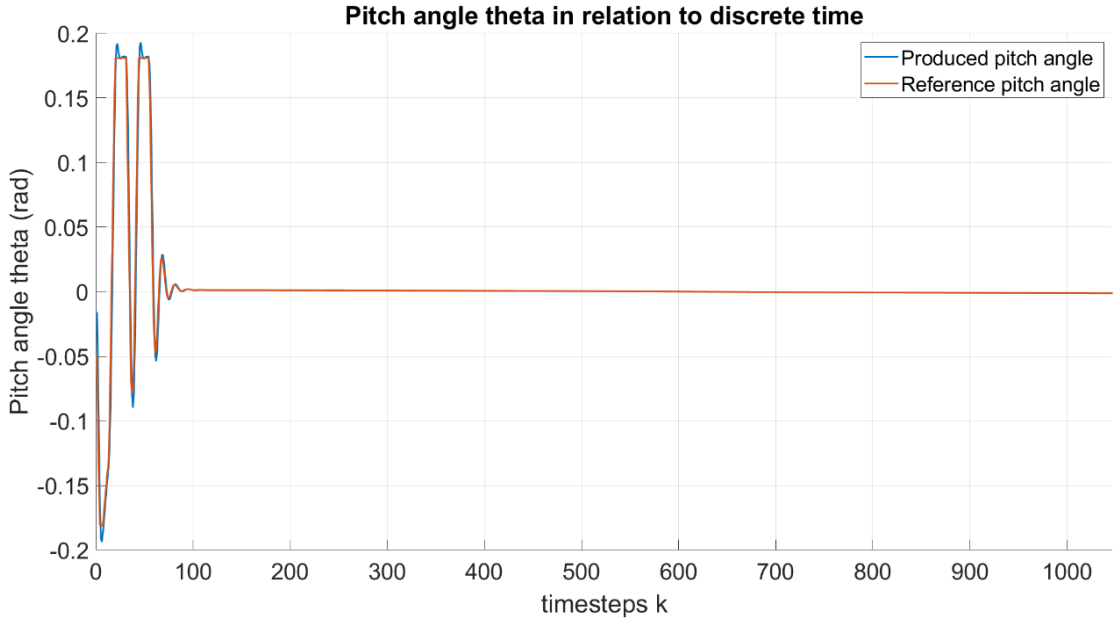


Figure 65. Pitch angle θ in relation to time for trajectory tracking (1st set) using the MPC design

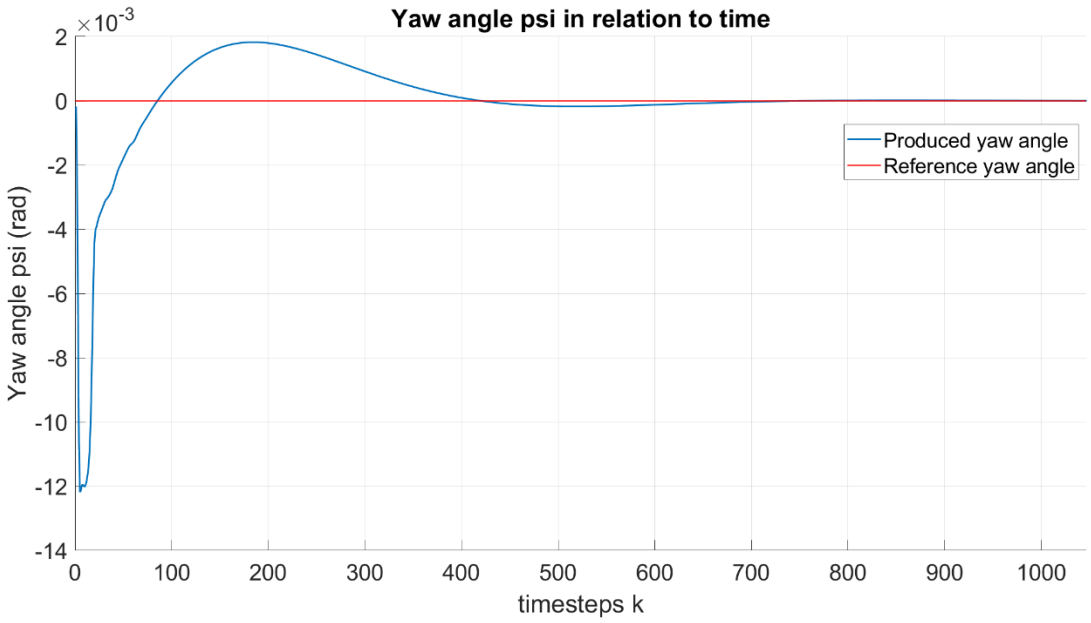


Figure 66. Yaw angle ψ in relation to time for trajectory tracking (1st set) using the MPC design

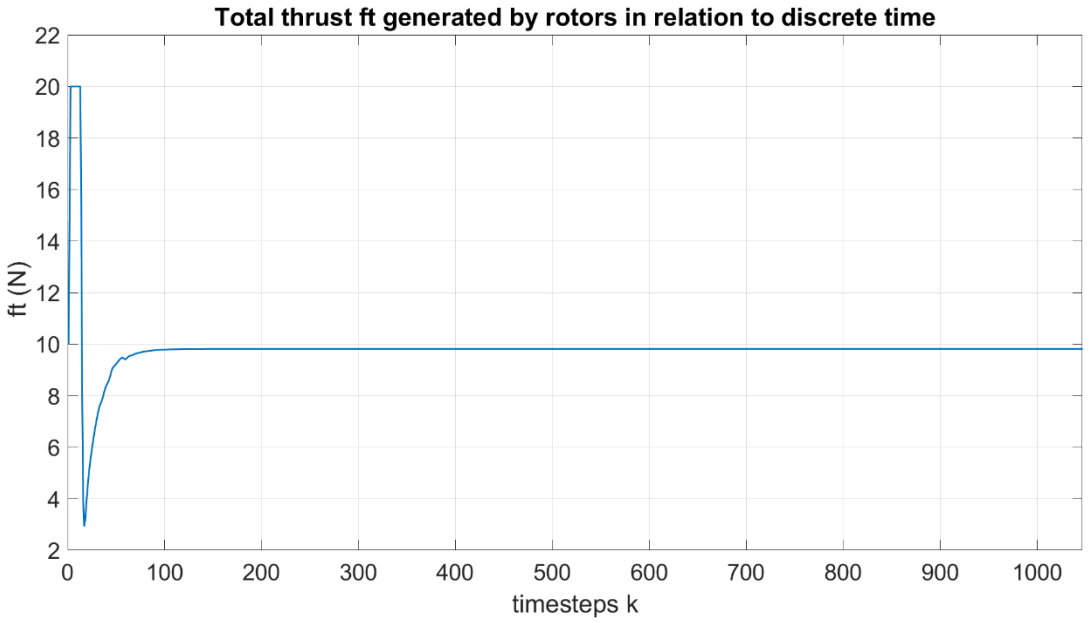


Figure 67. Thrust f_t in relation to time for trajectory tracking (1st set) using the MPC design

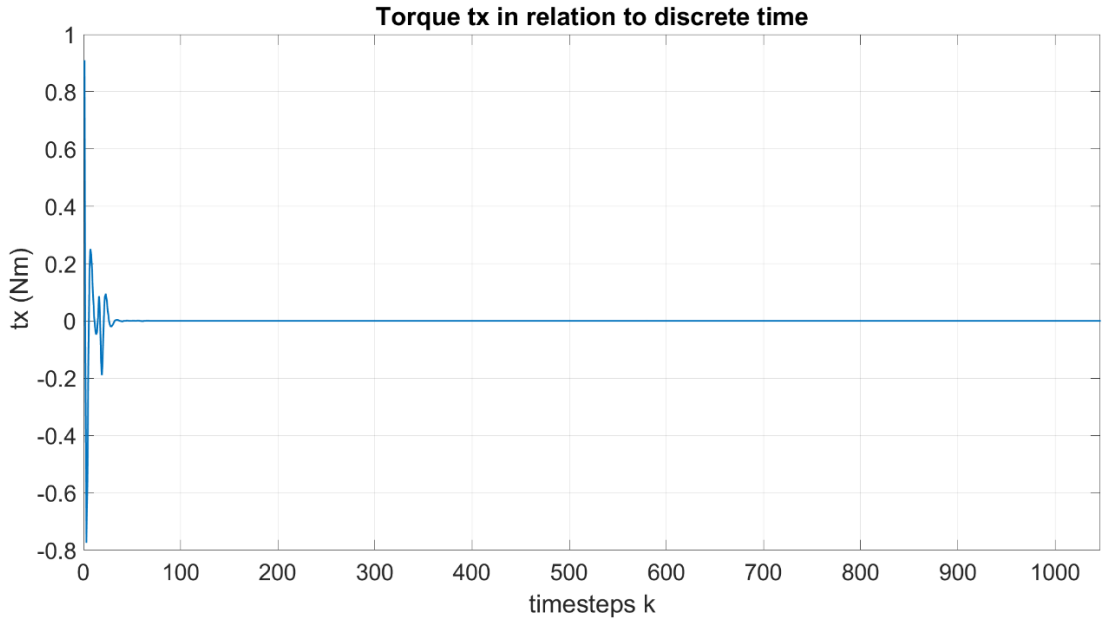


Figure 68. Torque τ_x in relation to time for trajectory tracking (1st set) using the MPC design

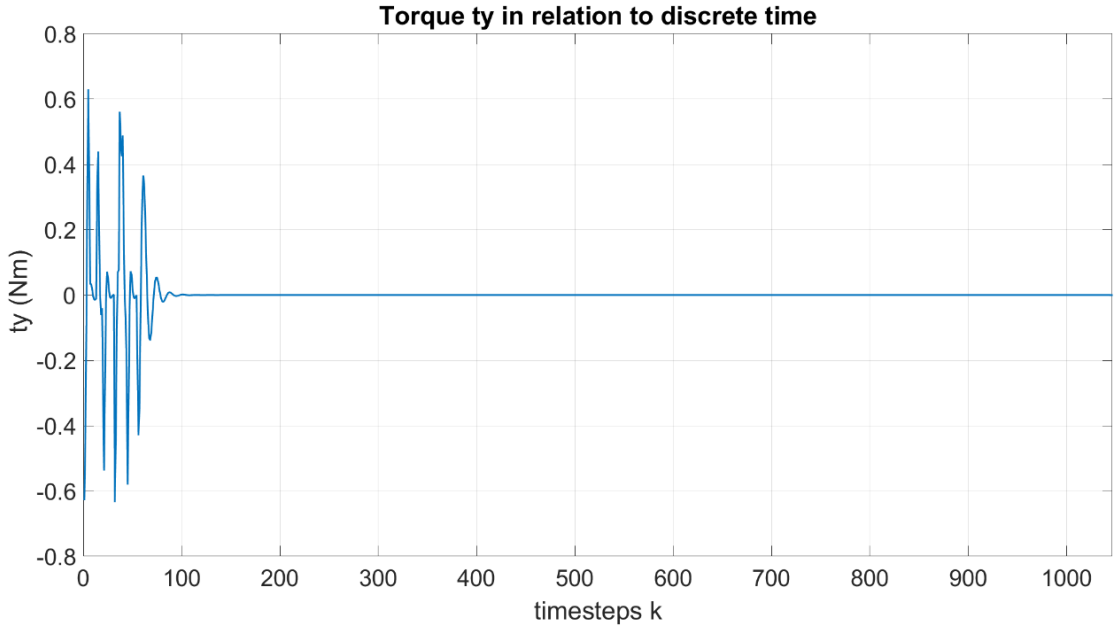


Figure 69. Torque τ_y in relation to time for trajectory tracking (1st set) using the MPC design

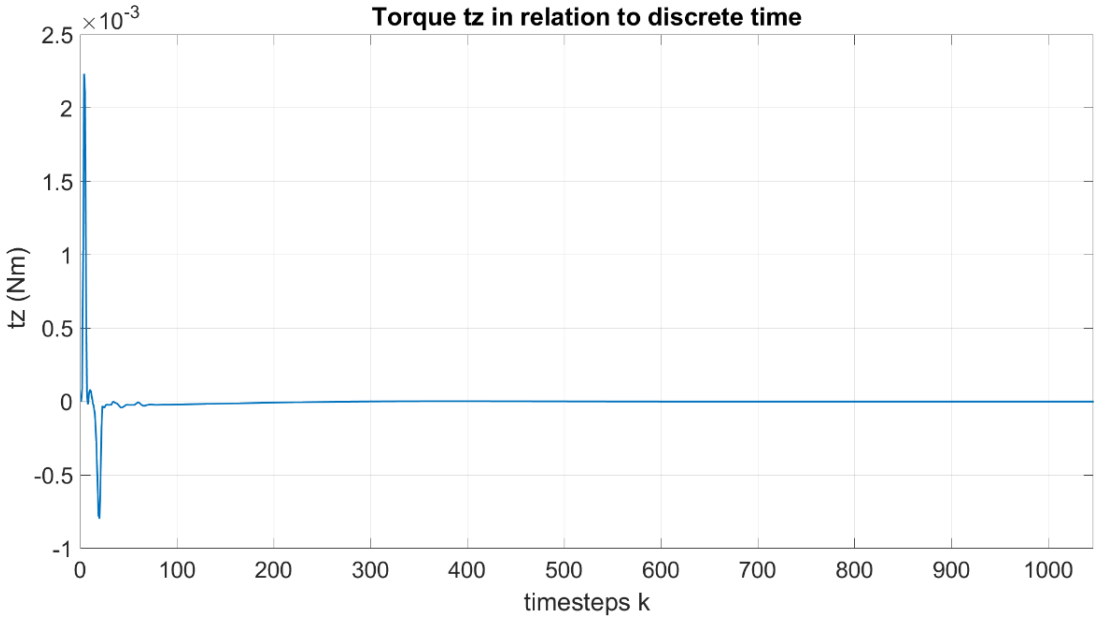


Figure 70. Torque τ_z in relation to time for trajectory tracking (1st set) using the MPC design

Furthermore, apart from the individual graphs for positions x, y, and z, it is essential to create a 3D visualization of the trajectory to observe the quadrotor's behavior in three-dimensional space. In Figure **71**, the spiral-like trajectory for three complete periods is presented for that purpose.

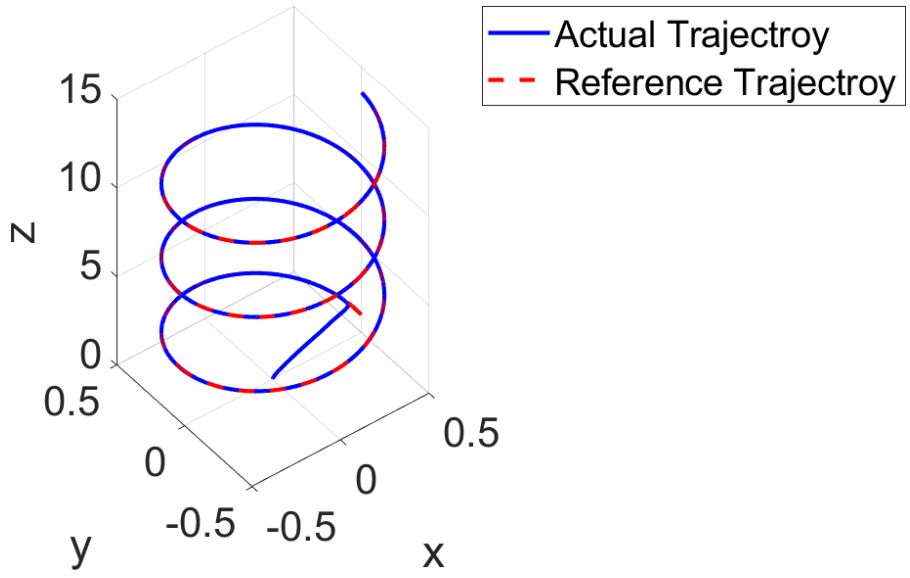


Figure 71. Actual trajectory compared to reference trajectory for the 1st set using the MPC design

Figure **71** reveals the quadrotor's exceptional ability to closely adhere to the trajectory, even during changes in orientation, and it does so with remarkable agility. This quantitative precision is

further elucidated by the calculation of the MSE values for each of the three separate axes and the overall system, as displayed in Table **11** for three time periods. The diminutive values are indicative of the system's exceptional performance, in all axes and overall.

Table 11. The MSE values for the three axes – 1st trajectory using the MPC design

	Mean Squared Error (MSE)
x-axis trajectory	0.0007
y-axis trajectory	0.0000
z-axis trajectory	0.0024
Mean Euclidean Distance	0.0043

3.5..2 Simulation for the 2nd trajectory set

Analogously to the simulation of the slow trajectory, the simulation graphs for the fast (the 2nd) trajectory will be held for a timespan of half a period (approximately 214 timesteps).

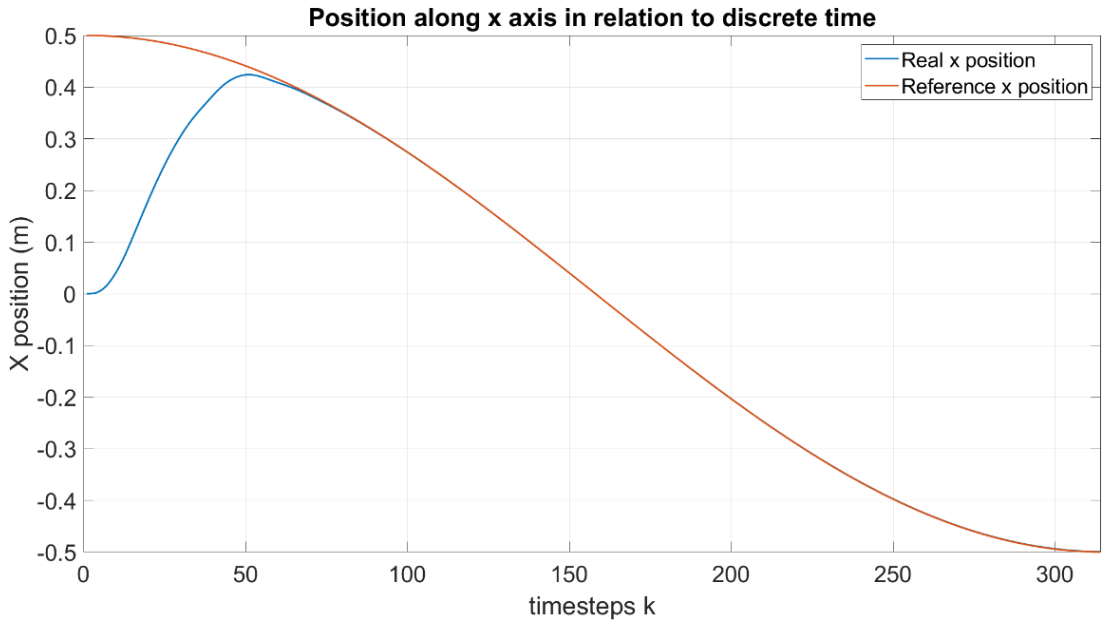


Figure 72. Position x and x_{ref} in relation to time for trajectory tracking (2nd set) using the MPC design

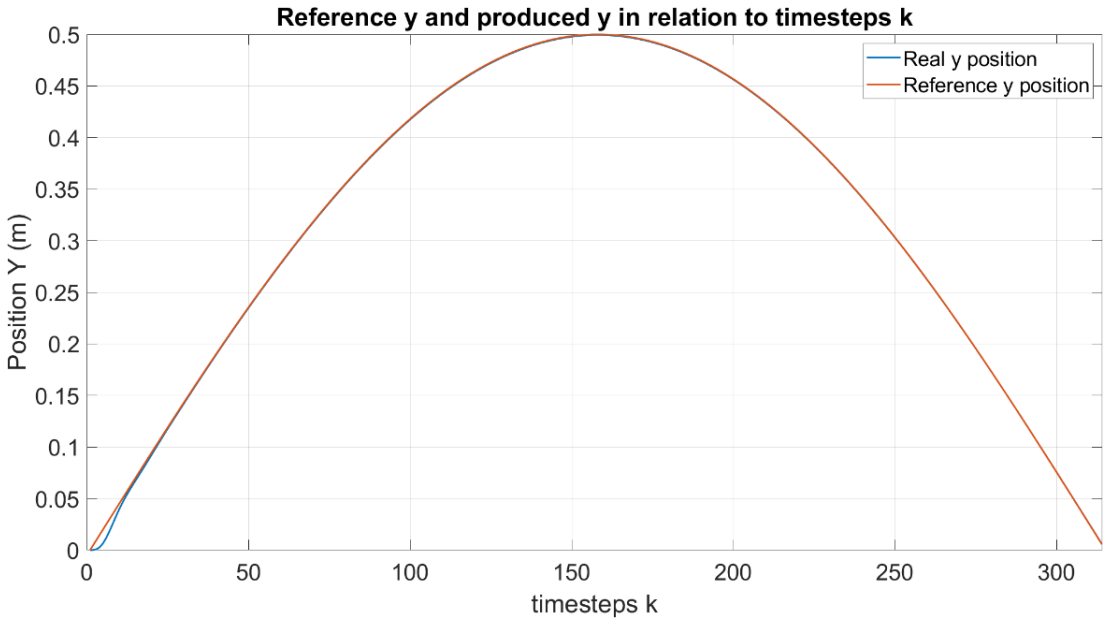


Figure 73. Position y and y_{ref} in relation to time for trajectory tracking (2nd set) using the MPC design

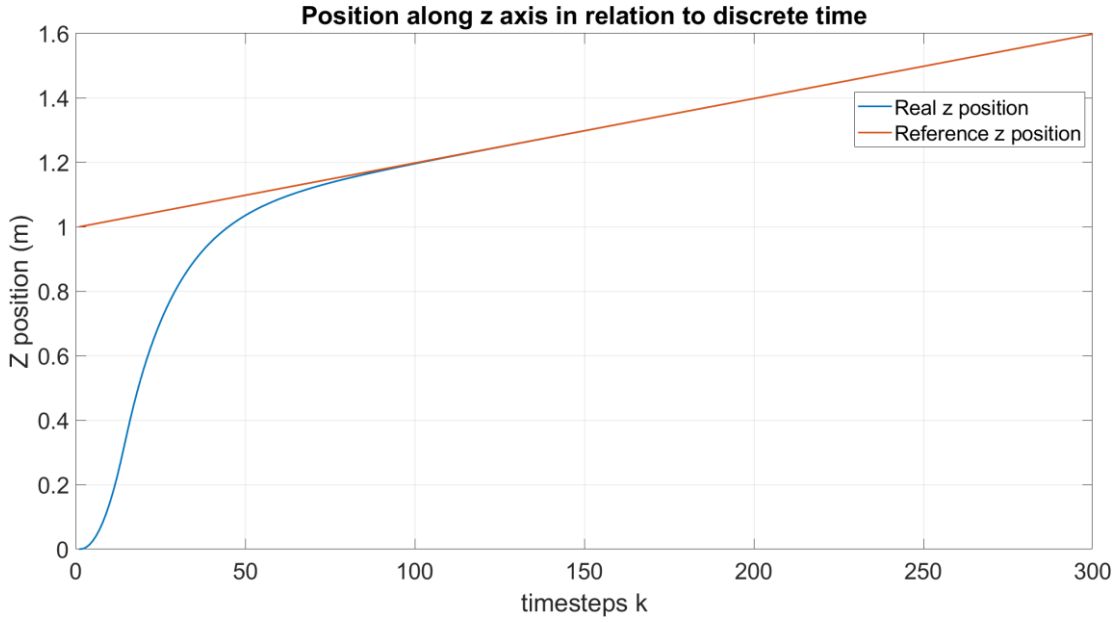


Figure 74. Altitude z and z_{ref} in relation to time for trajectory tracking (2nd set) using the MPC design

As evident from Figure **72**, Figure **73** and Figure **74**, which illustrate the tracking performance along the X-Y and Z axes, respectively, through the use of MPC design, the quadrotor exhibits exceptional performance. Despite the faster nature of this trajectory set, there is demonstrated remarkable agility, swiftness in adapting to changes in orientation and prompt aligning with the desired

trajectory. In the following graphs (Figures 75 to 81) the angles and control inputs that make for this tracking performance are presented.

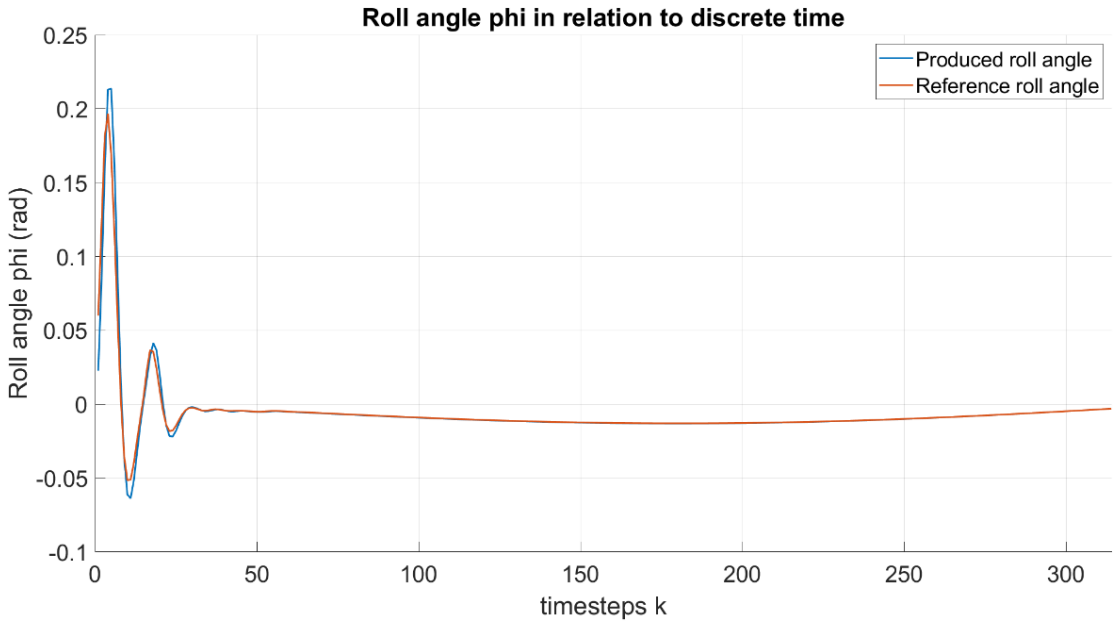


Figure 75. Roll angle φ in relation to time for trajectory tracking (2nd set) using the MPC design

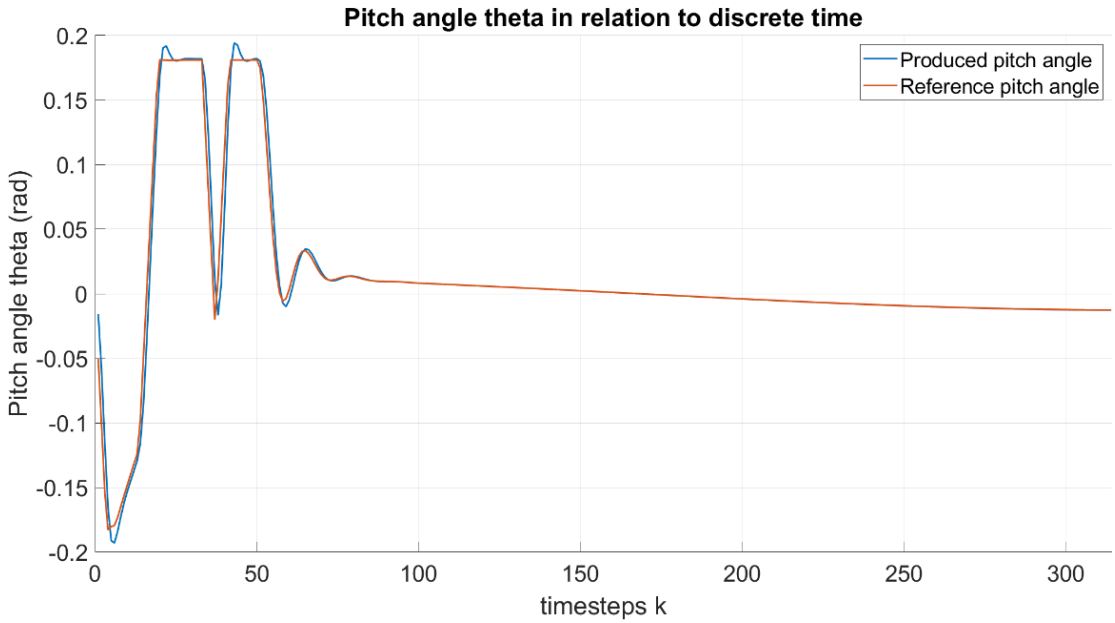


Figure 76. Pitch angle θ in relation to time for trajectory tracking (2nd set) using the MPC design

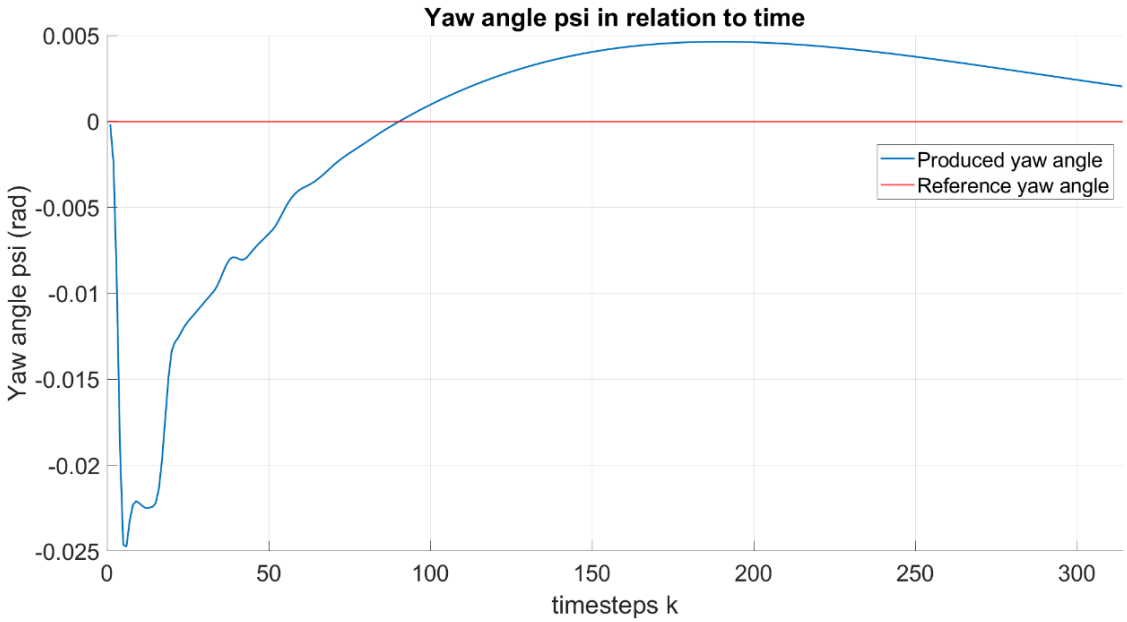


Figure 77. Yaw angle ψ in relation to time for trajectory tracking (2nd set) using the MPC design

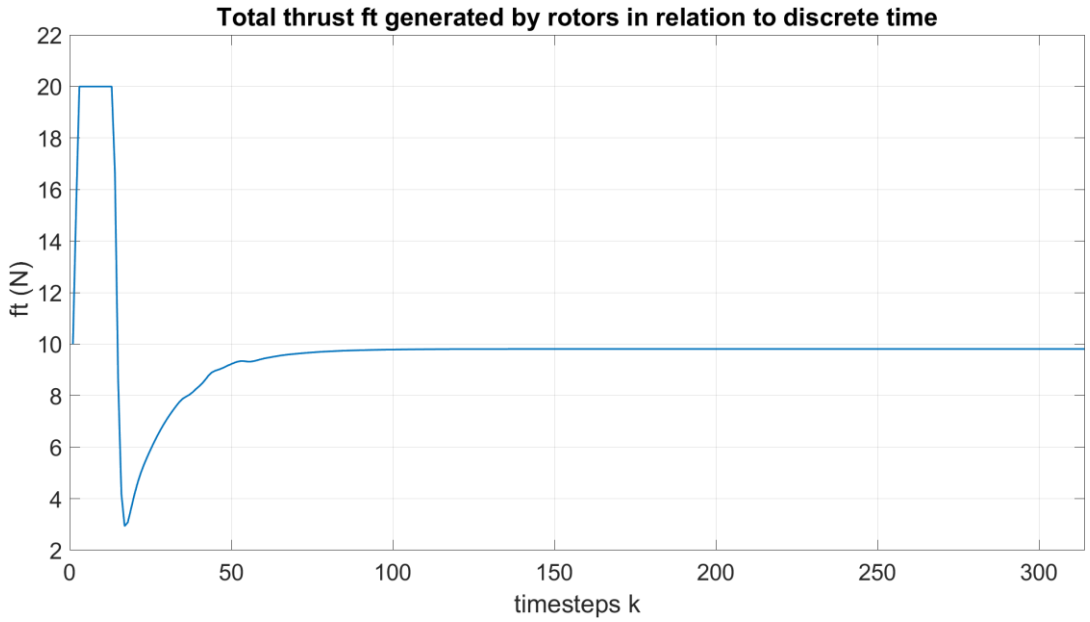


Figure 78. Thrust f_t in relation to time for trajectory tracking (2nd set) using the MPC design

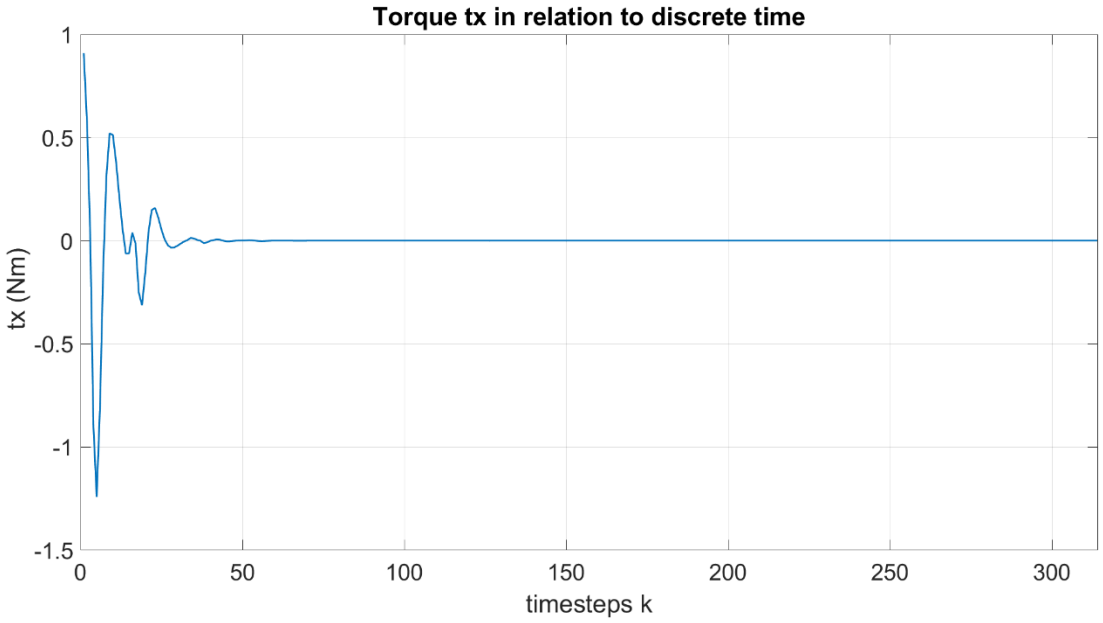


Figure 79. Torque τ_x in relation to time for trajectory tracking (2nd set) using the MPC design

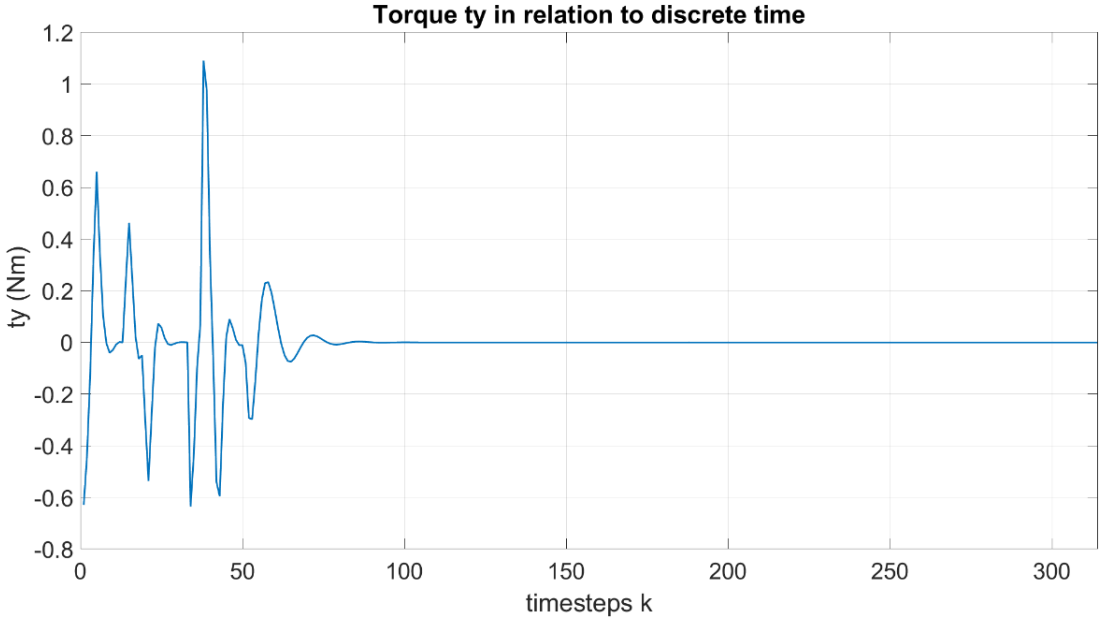


Figure 80. Torque τ_y in relation to time for trajectory tracking (2nd set) using the MPC design

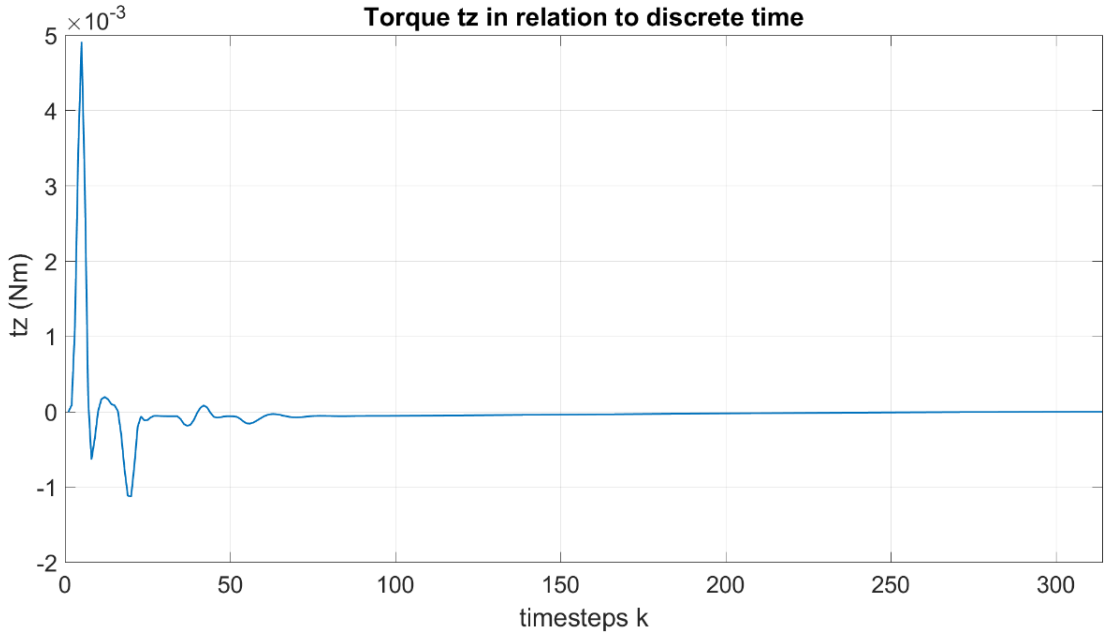


Figure 81. Torque τ_z in relation to time for trajectory tracking (2nd set) using the MPC design

The angles and control inputs display sharp and dynamic responses, underscoring the quadrotor's ability to execute rapid and agile maneuvers, thus resulting to precise tracking and control. The trajectory for the 2nd reference set in 3D space is presented in Figure **82** as a clear evidence of the quadrotor's exceptional performance thanks to the MPC design employed.

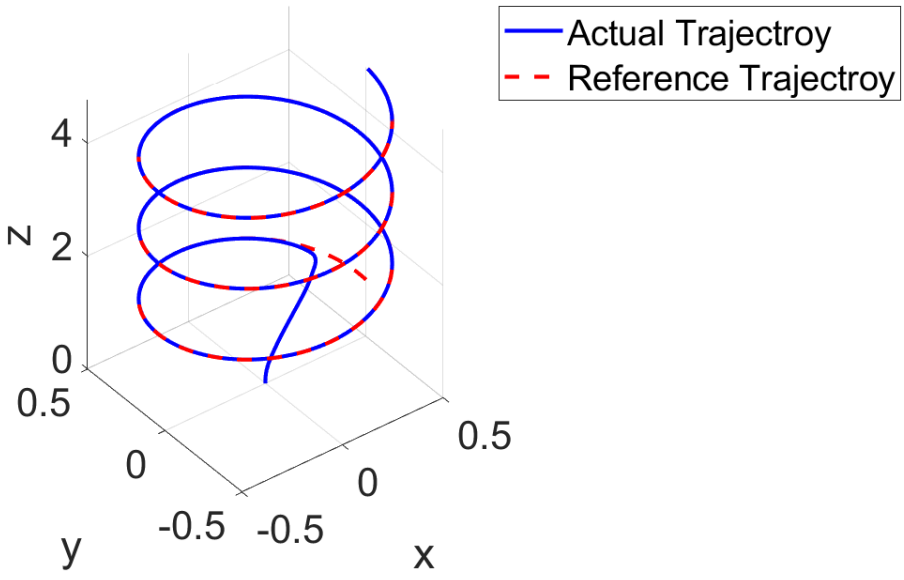


Figure 82. Actual trajectory compared to reference trajectory for the 2nd set using the MPC design

The MSE values for the 2nd trajectory are presented in Table **12**:

Table 12. The MSE values for the three axes – 2nd trajectory using the MPC design

	Mean Squared Error (MSE)
x-axis trajectory	0.0024
y-axis trajectory	0.0000
z-axis trajectory	0.0081
Mean Euclidean Distance	0.0145

3.5..3 Comparison with the PID design

From the presented graphs and the evaluation index MSE calculations, it becomes evident that the MPC design, which incorporates two MPCs for position and three PIDs for angle regulation respectively, outperforms the PID design, which utilizes a total of six PIDs for both position and angle regulation. This compelling evidence highlights the advanced capabilities of MPC, which offers a superior tracking performance, making it a very suitable choice for complex systems where precision and adaptability are paramount.

More specifically, the MSE, which is a key indicator of tracking accuracy, showcases a substantial difference; the MSE for the MPC design is approximately five times smaller than that of the PID design (1st set: 0.0043 vs 0.0205, 2nd set: 0.0145 vs 0.0741), underscoring the superior performance of the former.

Furthermore, when examining the angles, as well as the control inputs, intriguing dynamics are unveiled. More precisely, in the MPC design these parameters exhibit a more dynamic behavior, with steeper and more pronounced changes. While tuning plays an important role in this behavior, it also highlights that the MPC design employs more aggressive control actions, that makes for swift and adaptive responses, ultimately contributing to a superior tracking performance. This, of course, comes with more energy consumption – computational load.

In Table **13**, a comprehensive comparison of the PID and the MPC design for the quadrotor system examined is presented, highlighting their distinct characteristics and performance attributes. These observations align with the established theoretical principles and advantages of MPC presented in section 3.1.

Table 13. Comparison between the PID design and the MPC design

	Quadrotor PID design	**Quadrotor MPC design**
Mean Squared Error	higher due to reactive control	lower due to predictive control
Adaptability	limited	high
Control Precision	limited	high
Constraint Handling	none	efficient handling
Computational load	lower	higher
Response Time	slower	faster
Multi-variable Control	separate PIDs for each variable	unified control with consideration of the interdependence between x and y

Overall, the simulations conducted provide clear evidence that the Model Predictive Control (MPC) strategy employing two MPCs for position control and three inner-loop PID controllers for angle regulation, outperforms the design that employs only PID controllers, by a significant margin. This exceptional performance can be attributed to the multiple advantages and predictive nature of MPC, which allows for precise control and trajectory tracking. That being said, while the tuning process may be intricate, it offers the flexibility to fine-tune numerous parameters in order to achieve the desired performance.

So, after conducting a thorough evaluation and assessment of the performance of MPC in the context of trajectory tracking, spanning two distinctly paced scenarios, it becomes apparent that this advanced control strategy has not only established its efficacy, but also reveals a wealth of potential for application in even more intricate systems. More specifically, the promising outcomes observed in this chapter inspire confidence in its versatility and adaptability, and thus, in its application to address the unique challenges posed by a swarm of quadrotors, as opposed to the control of a single entity.

The transition from the realm of single-agent control to orchestrating the collective behavior of a swarm is an example of harnessing the full spectrum of MPC's capabilities and underscores the potential of this control strategy in real-world scenarios, ranging from autonomous surveillance and precision agriculture to disaster response and environmental monitoring, where the autonomous coordination of multiple agents is paramount.

It is within this context that the next chapter is explored, where the application of MPC to swarm robotics takes center stage, offering a promising glimpse into the future of autonomous multi-agent systems.

CHAPTER 4: Swarm of quadcopters

In the final chapter of this diploma thesis, the proposed control framework presented in Chapter 3, with two separately working Model Predictive Controllers, is utilized for the management of multiple quadcopter agents, rather than a single quadcopter, also known as a swarm of quadcopters. In this context, the chapter begins by providing some introductory information about the swarms, as well as the distributed MPC (DMPC) framework implemented. Finally, in order to establish the efficiency and functionality of the designed control system, simulations are conducted for trajectory tracking, with emphasis on guaranteeing not only collision avoidance between the agents of the swarm, but obstacle avoidance, as well.

4.1 Swarm

A swarm of quadrotors, commonly referred to as a quadcopter swarm or a drone swarm, embodies the collective intelligence of multiple autonomous or semi-autonomous quadrotors that work collaboratively to achieve a common goal. By leveraging the power of cooperation and coordination, quadrotor swarms unlock unprecedented capabilities, enabling them to perform complex tasks that would be beyond the reach of a single drone. As a result, quadrotor swarms find applications in many industries or domains, like surveillance, search and rescue in disaster scenarios, precision agriculture, environmental monitoring, communication relay and more [4]. Figure **83** presents a swarm of five quadcopter agents.

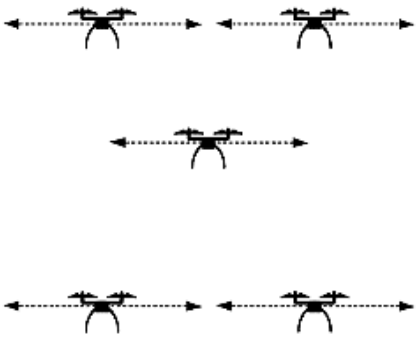


Figure 83. Visual representation of a swarm of quadcopters with 5 agents

In the context of swarms, *formation* [37] refers to the spatial arrangement of the drones and how they maintain specific spatial configuration while performing tasks collectively, which can change depending on the specific application, mission objectives, and the complexity of the swarm task. Whatever the case, orchestrating a swarm of quadrotors is a challenging task that requires the

combination of advanced control algorithms, reliable communication systems, and well-coordinated design and planning. This is because, there are several factors that need to be addressed for an effective and safe operation, including:

- *Communication and Coordination*: the quadcopters must be able to share information to work collaboratively and maintain cohesion, under a robust and low-latency communication network.
- *Collision Avoidance*: avoiding collisions between quadrotors is critical for ensuring the safety and integrity of the swarm, as it navigates through complex environments or potential hazards. Thus, the relevant mechanisms must be robust enough and able to adjust trajectories in real-time.
- *Path Planning*: efficient path planning is necessary to optimize the trajectories of individual quadrotors and the overall swarm, while also considering factors such as energy consumption, mission objectives, obstacle avoidance, formation control and dynamic changes in the environment.
- *Scalability*: as the number of drones in the swarm increases, the complexity of communication and coordination grows exponentially, thus requiring sophisticated algorithms to ensure efficient cooperation.

Finally, depending on various factors like the swarm's size, the nature of the task, the available computational resources or the communication capabilities, different control strategies can be chosen, combined, or even customized to suit specific applications. In the context of swarm, a *control strategy* is the approach or methodology used for coordination and control of autonomous agents, to achieve a collective goal or perform a task. Some common control strategies used in swarm robotics are:

- *Distributed Control*: this strategy involves giving each agent a (certain) degree of autonomy and decision-making capabilities. The communication between the agents and, ultimately, the coordinated behavior, is achieved through information sharing. [29]
- *Centralized Control*: in contrast to distributed control, this strategy involves a single central controller that makes the decisions and sends the commands to all agents. This approach is common in scenarios when real-time coordination is critical. [38]
- *Hierarchical Control*: with this strategy, there is a centralized controller placed somewhere that determines high-level actions, as well as distributed controllers at each agent to implement these actions. [39]

- *Leader-Follower Control*: some agents within the swarm act as leaders, which make decisions and guide the behavior of the swarm, while the rest act as followers of the leader. [40]
- *Learning-Based Control*: the agents learn and adapt their behavior over time, by using machine learning techniques, such as reinforcement learning or neural networks. [41]

4.2 Design of the quadrotor swarm

The control architecture employed for the swarm in the scenario of trajectory tracking is the same as that for Chapter 3., which incorporates two MPC controllers, one devoted to managing altitude (z-axis), and the other for the x-y plane, as well as three Proportional-Integral-Derivative (PID) controllers for the regulation of the inner-loop angles.

This MPC control strategy is used for the developing of a comprehensive distributed MPC (DMPC) framework, that orchestrates the collective behavior of the swarm. In that context, the different agents are controlled autonomously, eliminating the necessity of a single - central controller, which could serve as a potential point of failure. Moreover, the proposed framework doesn't predetermine the trajectories for each agent, but each one of them adjusts its behavior to closely follow the real-time desired trajectory, allowing for more agile responses to changing conditions and unforeseen obstacles.

Furthermore, in the context of a swarm, as already mentioned in sub-chapter 4.1., a mechanism is required to facilitate communication among the agents to prevent collisions. When using the aforementioned control framework, it is important to remember that there are two distinct controllers utilized for position control that solve two separate optimization problems, with the z-axis MPC taking precedence.

That being said, a straightforward approach to address this is to initiate the procedure by executing the z-axis MPC controller, while maintaining the existing framework intact and without incorporating any collision avoidance provisions at this stage. Subsequently, upon generating the solution, the z-axis control output f_t is transmitted to the x-y MPC controller, accompanied by the x-y-z triad data predictions from the prior time instant k-1. So, within the x-y controller's MPC scheme, a collision avoidance constraint is introduced, as a nonlinear constraint function in the optimization problem, which is designed to ascertain the euclidean distance between the focal agent and all other agents throughout the prediction horizon. In the general case, this 3D distance must be greater than the agents' radius, which essentially defines a safety margin as an imaginary sphere around them.

$$Collision\ avoidance\ constraint: Euclidean\ Distance\left(Agent_i, Agent_j\right) =$$
$$= \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2} > agent's\ radius\ , for\ all\ of\ Hp \qquad (4.1)$$

where (x_i, y_i, z_i) represent the position of the current agent $Agent_i$ and (x_j, y_j, z_j) represents the position of the other agents $Agent_j$, in the three-dimensional space.

Concurrently, the x-y-z triad for the current agent is accounted for, where z-component information stems from the completed z-axis MPC's predictions and the x-y components emanate from the ongoing x-y controller's prognostications at that specific point in discrete time k. This strategy is presented in Figure **84**.

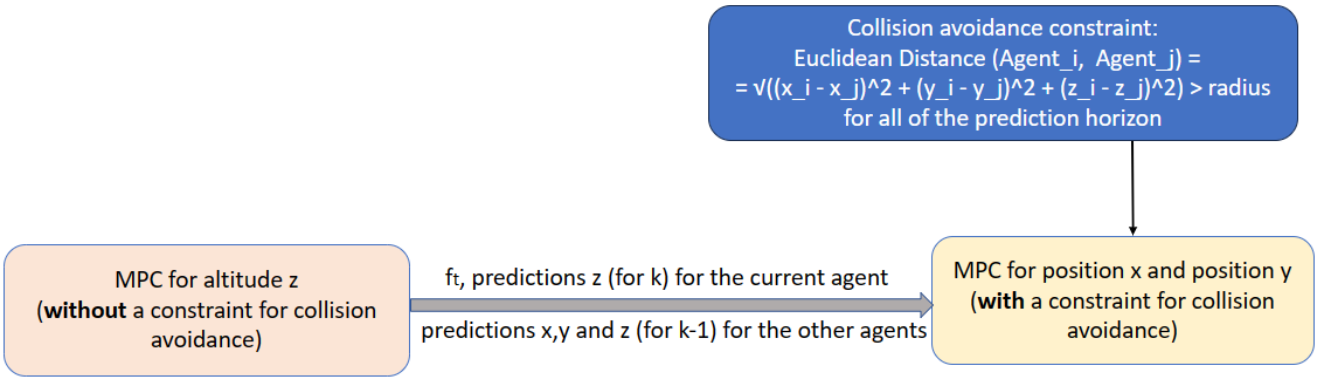


Figure 84. The Distributed MPC framework communication strategy for collision avoidance

This inter-agent communication allows the quadrotors to make decisions based on not only their own state but also the predictions regarding the states of the other agents, thus being able to adjust their trajectories and avoid potential collisions while also working together to achieve their individual and overall optimization objectives.

It is important to note that this proposed strategy is attuned to prioritizing control for altitude z, since it is arbitrarily assumed that the z-axis will present a better trajectory (i.e., a smaller error) than that of x-y. However, while this compromise, necessitated by the dual-controller framework, is not an optimized way of approaching the swarm case, it offers a pragmatic means to address collision avoidance concerns within the given context.

Finally, this diploma thesis does not employ a predefined formation control strategy for the swarm; instead, the quadrotors autonomously and dynamically determine their positions and formations during operation (online). This means that since the quadrotors are able to communicate

and coordinate their movements, through the aforementioned strategy, they make decisions collectively to achieve the desired formation, without the absolute need for a fixed strategy set beforehand.

4.3 Simulation for trajectory tracking with collision avoidance

For this simulation, a multi-agent swarm composing of three identical quadcopters commencing from an initial configuration where each agent occupies an adjacent position in the x axis is being considered (Agent 1: x = -1 [m], Agent 2: x = 0 [m], Agent 3: x = 1 [m]). The reference trajectory is:

$$x_{ref(t)} = 5 \cdot \cos(0.5 \cdot t)$$
$$y_{ref(t)} = 5 \cdot \sin(0.5 \cdot t)$$
$$z_{ref(t)} = -1 - 0.1 \cdot t$$

The tuning parameters are presented in the following Tables. More specifically, Table **14** presents the chosen horizons for the swarm. The Control Horizons have remained the same as those chosen in Chapter 3, due to their excellent performance, while the Prediction Horizons have been extended slightly compared to the previous chapter, maintaining uniformity for both z and x-y dimensions, in pursuit of a more universally applicable approach. Table **15** presents the PID values for the swarm, which are generally basic values for angle regulation PID control without excessive tuning. Contrary, the values of the penalty matrices, as depicted in Table **16**, are carefully and intentionally chosen to be relatively small, especially for the error penalty matrix Q_1. Generally basic values were also chosen for the maximum and minimum parameter values in Table **17**.

Moreover, in the MPC framework designed for an individual quadcopter, terminal constraints were implemented for both the z-axis and the x-y plane. However, in the context of swarm control, a strategic decision was made to simplify the system while preserving its efficiency. Thus, the terminal constraint was retained exclusively for the z-axis; a simplification driven by the need for enhanced maneuverability and adaptability required for such complex task.

Finally, given that the quadrotors examined have an arm length of 0.24 m, a safety radius of 0.3 m around each agent is deemed acceptable. At this point it is important to highlight the fact that the tuning parameters and variables remain consistent for all agents, as it is assumed that they are geometrically identical quadcopters.

Table 14. The selected swarm Control and Prediction Horizons for z and x-y

MPC horizons	
H_c for z	3
H_p for z	40
H_c for x-y	6
H_p for x-y	40

Table 15. The PID values for angle regulation for the Distributed MPC design for a swarm of quads

PID values			
	Proportional gain k_P	Integral gain k_I	Derivative gain k_D
PID controller for angle phi	0.05	0.0001	0.15
PID controller for angle theta	0.05	0.0001	0.15
PID controller for angle psi	0.005	0.0001	0.05

Table 16. The selected Penalty Matrices for the Distributed MPC design for a swarm of quadcopters

Peanalty matrices			
	Error penalty matrix Q_1	Velocity penalty matrix Q_2	Design variable penalty matrix R
altitude z	$diag(2 \cdot ones(hp, 1))$	$diag(0.05 \cdot ones(hp, 1))$	$diag(0.01 \cdot ones(hc, 1))$
position x	$diag(1.5 \cdot ones(hp_{xy}, 1))$	$diag(0.5 \cdot ones(hp_{xy}, 1))$	$diag(0.01 \cdot ones(hc_{xy}, 1))$
position y	$diag(1.5 \cdot ones(hp_{xy}, 1))$	$diag(0.5 \cdot ones(hp_{xy}, 1))$	$diag(0.01 \cdot ones(hc_{xy}, 1))$

Table 17. The minimum-maximum swarm values for the variables for the Distributed MPC design

Min and max values for the variables		
	Minimum value	Maximum value
f_t [N]	-20	20
u_x [rad]	-0.2	0.2
u_y [rad]	-0.2	0.2
df_t [N]	-5	5
du_x [rad]	-0.05	0.05
du_y [rad]	-0.05	0.05

At this point, it is important to highlight the fact that, while the collision avoidance constraint (specified by equation 4.1) dictates that the Euclidean distance between the quadcopters should exceed the certain chosen radius, different simulation scenarios conducted witnessed violations of this constraint. One possible explanation is that the solver responsible for evaluating these distances, i.e., SQP, relies on the predictions produced by the MPC models at the previous timesteps and not on the current positions. That being said, if these predictions are significantly wrong, the solver might erroneously conclude that the real constraint is satisfied and that there aren't any collisions detected between the agents of the swarm.

In light of these observations, one potential solution turned out to be increasing the radius value used in this constraint, thus making the constraint bound stricter, to account for uncertainties and inaccuracies. After several trial-and-error attempts, it was determined that setting the constraint radius to three (3) times its original value effectively addressed the issue (Table **18**).

Table 18. The chosen radiuses for collision avoidance in trajectory tracking of the swarm

Chosen radiuses	
Agents' safe radius	0.3 [m]
Collision avoidance constraint radius	3·agents' safe radius [m]

Figure **85**, Figure **86** and Figure **87** present the trajectory tracking of the agents of the swarm with respect to the reference trajectories for x, y, and z axes, respectively, while Figure **88** presents the trajectory of the swarm in three-dimensional space. Since all agents share the same reference trajectory

rather than a predefined one, it is expected that there will be deviations for the trajectory tracking in the x-y plane, due to the imperative need to avoid collisions. On the contrary, with regard to trajectory tracking along the z-axis, since there are no constraints related to collision avoidance in the MPC for the z-axis, it is expected that the behavior of the agents will maintain the same or relevant altitude.

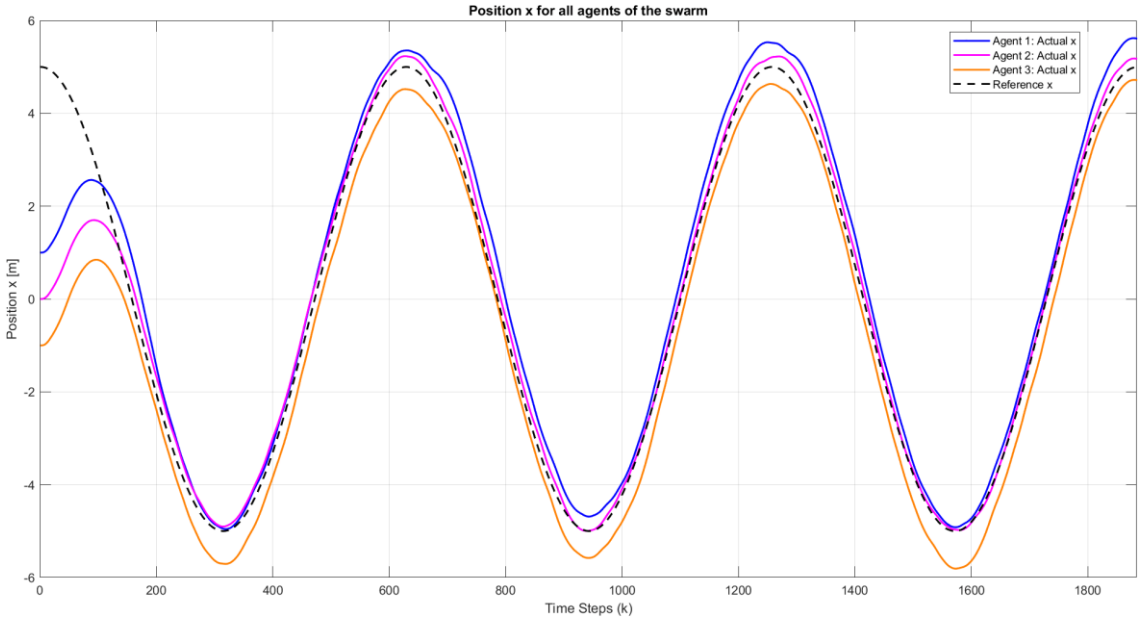


Figure 85. Position x tracking for the agents of the swarm using Distributed MPC

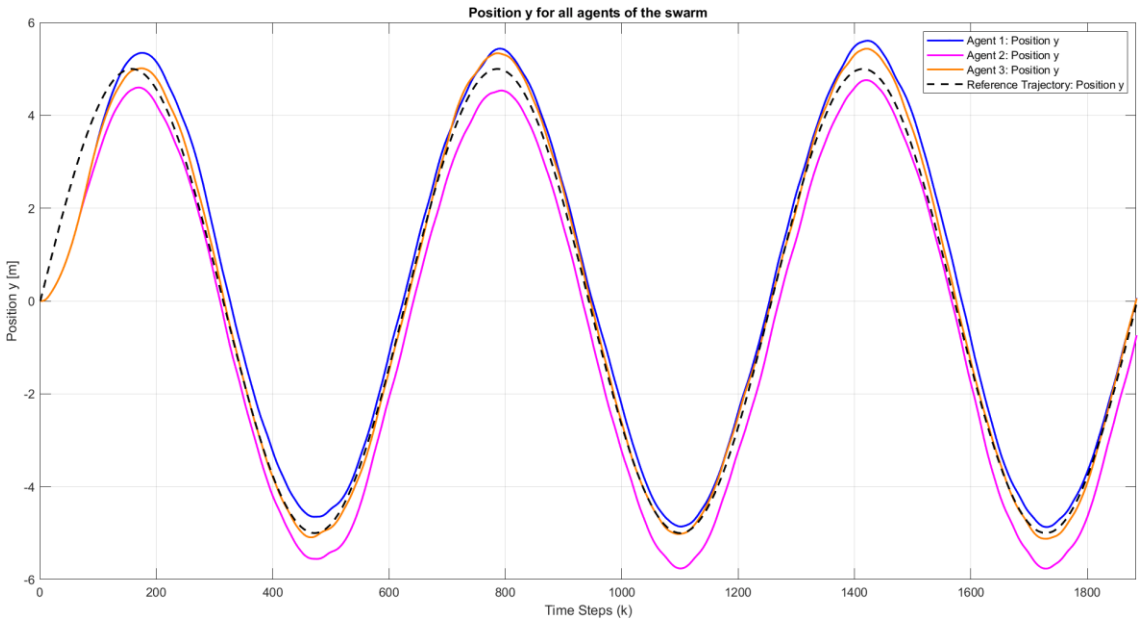


Figure 86. Position y tracking for the agents of the swarm using Distributed MPC

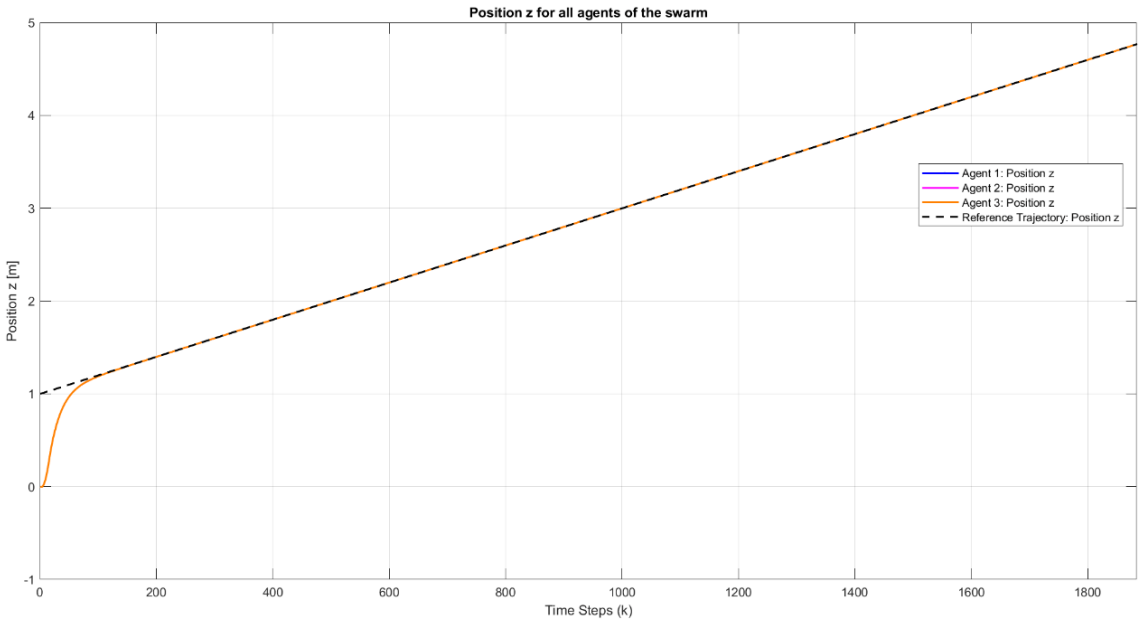


Figure 87. Position z tracking for the agents of the swarm using Distributed MPC

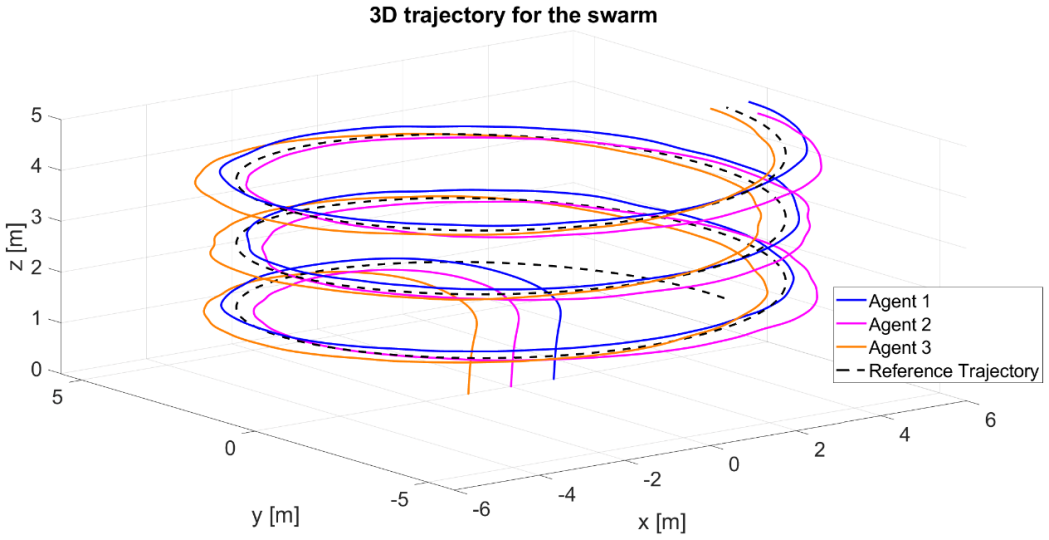


Figure 88. The 3D trajectory tracking for the agents of the swarm using Distributed MPC

As anticipated, the agents display minor deviations from the actual trajectory in the x and y axes, which primarily stem from the need to steer clear of potential collisions among themselves. However, the overarching observation is that the agents effectively adhere to the reference trajectory, without the imposition of a separate and predetermined reference trajectory for each one of the agents or a predefined formation. In other words, they collectively navigate while maintaining a certain degree of flexibility in their individual paths; a behavior that underscores their ability to autonomously and intelligently adapt their movements to ensure collision-free operation and precise trajectory tracking.

The efficacy of the designed DMPC methodology employed can be further evaluated through the calculation of an evaluation index, such as the average sum of euclidean distances (obtained by the calculation of the 3D distances of all agents at every timestep divided by the total number of timesteps k).

This calculation is highly dependent on the chosen constraint radius value, since opting for a smaller value effectively relaxes the constraint, making it less stringent. Consequently, this adjustment allows for the agents' trajectories to be closer in proximity, while still be considered compliant with the constraint, which ultimately results in a more favorable evaluation index. Contrary, increasing the constraint radius leads to the tightening of bounds, since the agents must adhere to a greater constraint value. This interdependence between the tracking evaluation index and the constraint radius is presented in Table **19**.

Table 19. Comparison between the constraint radius and tracking evaluation index using Distributed MPC

Chosen constraint radius	Average Sum of Euclidean Distance for three time periods
$3 \cdot radius$	2.19
$2.7 \cdot radius$	2.04
$2.5 \cdot radius$	1.95
$2.3 \cdot radius$	1.85
$2 \cdot radius$	Collision constraint violation

4.4 Trajectory tracking with obstacle evasion

To incorporate obstacle avoidance within the Distributed MPC framework, a distinct constraint is essential to be included into the optimization problem alongside the collision avoidance constraint. More specifically, considering an obstacle represented by a sphere along the reference path, with its center on such path at coordinates (O_x, O_y, O_z) and a known radius, it is possible to introduce a constraint that enforces that the Euclidean distance between each agent of the swarm and the center of the obstacle to be greater than the agent's radius and the obstacle's radius for all of the prediction horizon. Mathematically, this can be expressed as:

$$Obstacle\ evasion\ constraint: Euclidean\ Distance(Agent_i, Obstacle) =$$
$$= \sqrt{(x_i - O_x)^2 + \left(y_i - O_y\right)^2 + (z_i - O_z)^2} > (agent's\ radius + obstacle's\ radius)$$
$$for\ all\ of\ Hp \tag{4.2}$$

where (x_i, y_i, z_i) represents the position of the agent and (O_x, O_y, O_z) represents the center of the obstacle, in the three-dimensional space. This constraint ensures that the agents maintain a safe distance from the obstacle at all times.

It is important to note that, in order to simulate a scenario that closely resembles real-life situations and practical swarm applications, it's essential to ensure that the swarm becomes aware of the obstacle only when the agents are within a close proximity to it. That being said, the obstacle avoidance constraint characterized by equation 4.2, should only be activated when the Euclidean distance between the current-real position of the agent and the center of the obstacle exceeds a predetermined threshold distance. In practical terms, this mimics how a swarm of quadcopter agents would encounter obstacles in real world, using cameras, LiDAR or other sensor systems for proximity awareness and obstacle perception. Figure **89** presents the updated strategy employed, that incorporates obstacle avoidance:

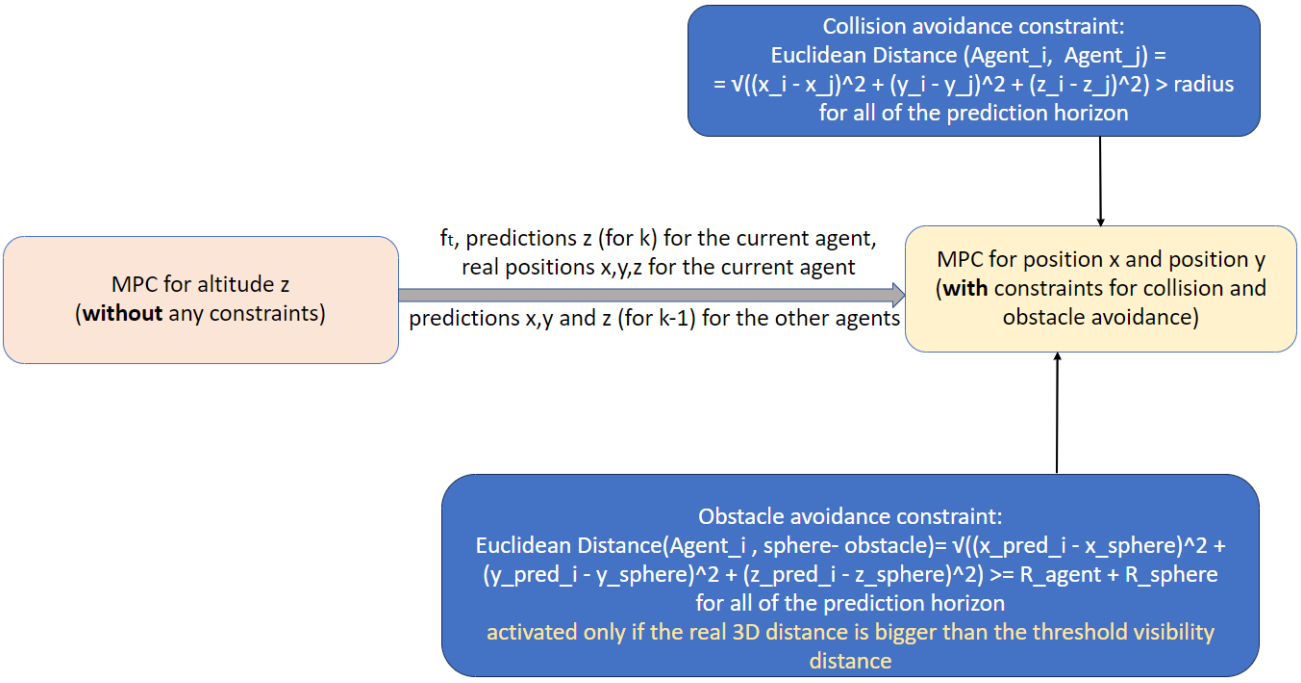


Figure 89. The Distributed MPC framework with collision and obstacle avoidance

4.4..1 Simulation analysis and results

For this simulation, there are three identical quadcopter agents in the swarm, which start from an initial configuration of adjacent positions in the y axis (Agent 1: x = 2 [m], y = 2 [m], z = -2 [m] , Agent 2: x = 2 [m], y = 3 [m], z = -2 [m] and Agent 3: x = 2 [m], y = 1 [m], z = -2 [m]). The reference trajectory examined is:

$$x_{ref(t)} = 2 + 1.5t$$
$$y_{ref(t)} = 2 + 0.7t$$
$$z_{ref(t)} = -2 - 0.1t$$

and the obstacle (its center) appears suddenly at coordinates:

$$O_x = 9.5\ [m]$$
$$O_y = 5.5\ [m]$$
$$O_z = -2.3\ [m]$$

The tuning parameters for obstacle evasion largely mirror those used for simple trajectory tracking, presented in sub-chapter 4.3. However, given the challenging task faced by the swarm agents — following a trajectory amidst sudden obstacle appearances while adhering to the terminal constraint for the z-axis — specific modifications were considered necessary.

More specifically, the PID controllers' settings, as well as the maximum and minimum values for the variables of the optimization problems remained consistent. However, the prediction horizons were deliberately extended, both in the z-axis and the x-y plane, as presented in Table **20** providing the agents with a broader prediction scope to effectively capture the dynamics of the environment.

Table 20 . The updated swarm control and prediction Horizons for z and x-y for obstacle evasion

MPC horizons	
H_c for z	3
H_p for z	50
H_c for x-y	6
H_p for x-y	50

Additionally, particular emphasis was placed on selecting appropriate values for the penalty matrices (Table **21**), specifically targeting Q_1 and Q_2, corresponding to the error and velocity error matrices. In this specific configuration, smaller values were chosen for Q_1 to enable accurate tracking without veering so closely as to risk collision, while the Q_2 matrices were intentionally set to zero for

all three dimensions. This deliberate adjustment nullifies the influence of velocity errors between the actual and reference velocities within the objective functions. Consequently, the agents' task becomes more manageable, and the swarm can navigate through complex scenarios, such as sudden obstacle appearances, with enhanced robustness. The rationale behind setting the Q_2 matrices to zero is followed by the exclusion of the velocity component in the z axis terminal constraint; a decision that grants the system with additional flexibility.

Moreover, The R matrix, serving as the penalty matrix for motion economy, was specifically augmented within the framework. This adjustment was crucial as it emphasizes the need for swarm agents to navigate both left and right of obstacles while ensuring they do not deviate excessively from their designated trajectory. This delicate balance in lateral movement is vital for maintaining precision in obstacle evasion scenarios. Finally, Table **22** presents the chosen radius for the swarm collision and obstacle avoidance simulation.

Table 21. The updated warm Penalty Matrices for the Distributed MPC design with obstacle evasion

Peanalty matrices			
	Error penalty matrix Q_1	Velocity penalty matrix Q_2	Design variable penalty matrix R
altitude z	$diag(1 \cdot ones(hp, 1))$	$diag(0 \cdot ones(hp, 1))$	$diag(0.1 \cdot ones(hc, 1))$
position x	$diag(0.75 \cdot ones(hp_{xy}, 1))$	$diag(0 \cdot ones(hp_{xy}, 1))$	$diag(0.5 \cdot ones(hc_{xy}, 1))$
position y	$diag(0.75 \cdot ones(hp_{xy}, 1))$	$diag(0 \cdot ones(hp_{xy}, 1))$	$diag(0.5 \cdot ones(hc_{xy}, 1))$

Table 22. The chosen radiuses for swarm collision and obstacle avoidance in trajectory tracking

Chosen radiuses	
Agents' safe radius	0.3 [m]
Obstacle's radius	1 [m]
Threshold visibility distance	3 [m]
Collision avoidance constraint radius	3 ·agents' safe radius [m]
Obstacle avoidance constraint radius	(obstacle's radius + 2.5·agents' radius) [m]

Figure **90**, Figure **91** and Figure **92** depict the swarm's trajectory tracking with obstacle evasion in the x, y and z dimensions, respectively, while Figure **93** illustrates the trajectory in three-dimensional space.

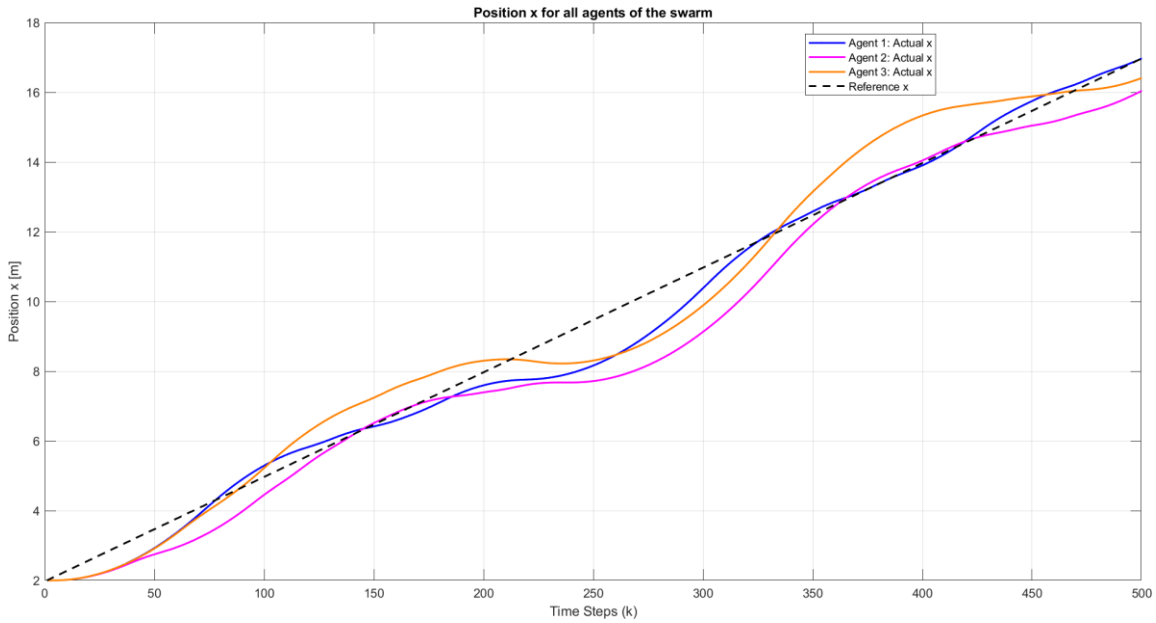


Figure 90. Position x tracking for the agents of the swarm using Distributed MPC

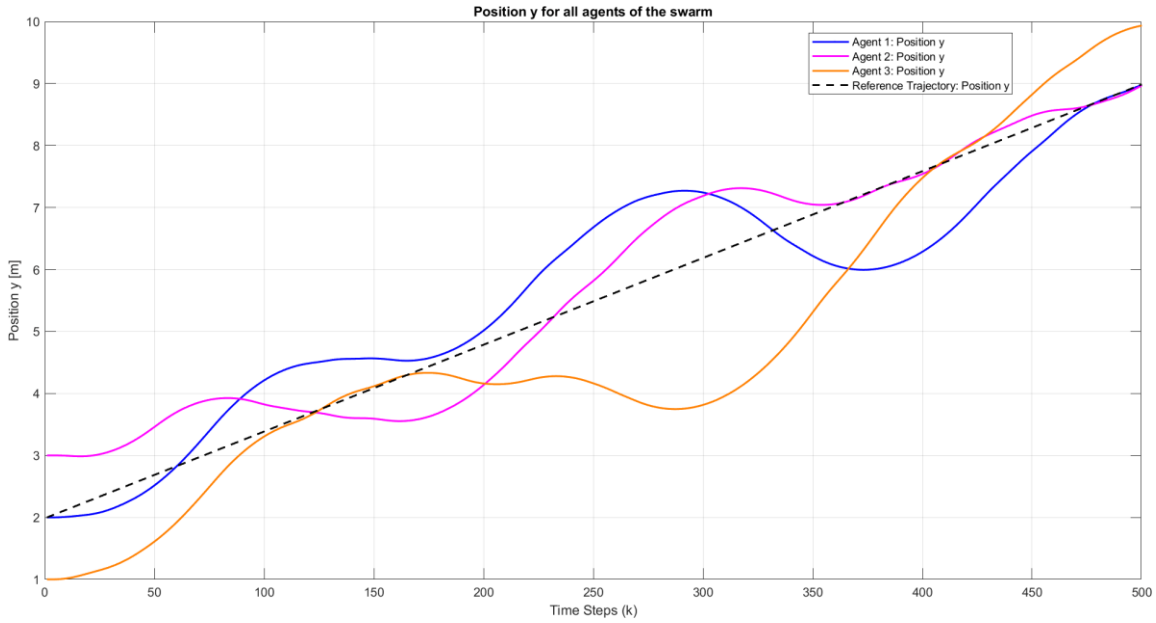


Figure 91. Position y tracking for the agents of the swarm using Distributed MPC

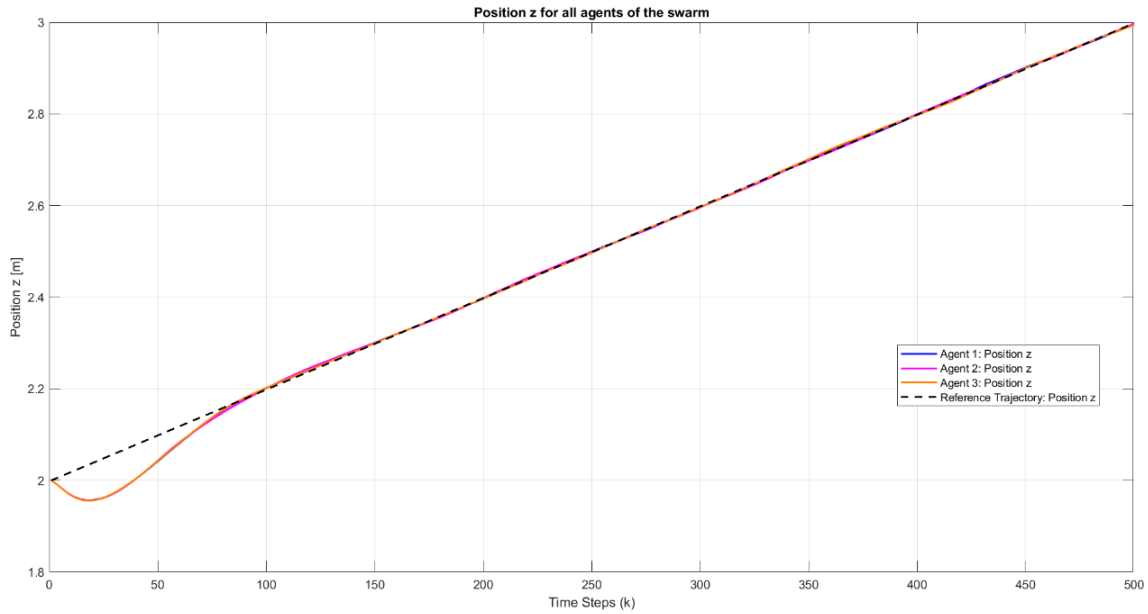


Figure 92. Position z tracking for the agents of the swarm using Distributed MPC

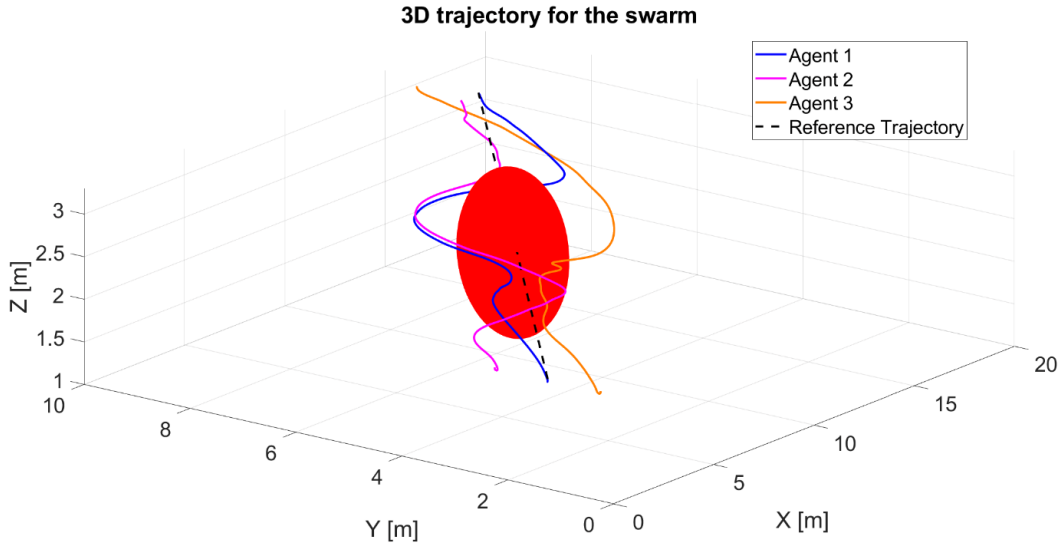


Figure 93. The 3D swarm trajectory tracking with obstacle avoidance using Distributed MPC

The graphs presented offer a comprehensive visual representation of the swarm's ability to adapt its path dynamically. This adaptation ensures accurate trajectory tracking while effectively avoiding obstacles and maintaining safe distances between agents to prevent collisions. Moreover, the Average Sum of the Euclidean Distances among the agents throughout the simulation (500 timesteps) stands at 2.88, providing quantitative confirmation of the swarm's precise tracking capabilities, as well as the Distributed Model Predictive Control (DMPC) framework's effectiveness in managing challenging swarm tasks.

CHAPTER 5: Conclusions

5.1 Summary, results and challenges

This diploma thesis embarked on a journey aiming to address the complex challenges associated with controlling a swarm of quadcopters in the scenario of trajectory tracking in unstructured environments. For this purpose, it proposed a comprehensive distributed model predictive control (DMPC) framework, capable of orchestrating the collective behavior of a quadcopter swarm, with not only precise trajectory tracking but intelligent collision and obstacle avoidance, as well; a task crucial for real-world applications.

The research began with a thorough exploration of quadcopter dynamics. This involved constructing a mathematical model to describe its complex and nonlinear behavior, followed by the design of a PID cascaded control system. This endeavor not only deepened the understanding of the control challenges associated with an individual quadcopter, but also revealed that relying solely on PID control is insufficient for satisfactorily addressing the complexities of path following, especially for faster trajectories in 3D-space.

In the subsequent phase, following the assessment of PID's subpar performance, a model predictive control (MPC) design was adopted for an individual quadcopter. This approach involved integrating the benefits of the sophisticated MPC method with that of PID control. Specifically, MPCs were utilized for position regulation, while PID controllers were employed for angle regulation in a cascaded design, for trajectory tracking scenarios. The simulations demonstrated the system's remarkable ability to handle path following tasks, instilling confidence in its potential for application in a quadcopter swarm.

In the final phase, the MPC control design for the quadcopter was extended to a swarm scenario. In this setup, each agent was autonomously controlled through individual MPCs within a distributed MPC framework (DMPC). Through inter-agent communication, involving their predicted positions in three-dimensional space, the system aimed to ensure trajectory tracking, collision avoidance and obstacle evasion. The simulations presented and the evaluation indexes calculated stand as a testament to the success of this method and the swarm’s ability to navigate in environments, while avoiding obstacles and collisions between the agents, thus affirming its effectiveness in challenging scenarios.

The proposed framework boasts several strengths. First, after thorough evaluation in a single quadcopter scenario, it consciously opted for a hybrid approach that employs both MPC for certain aspects of control and PID for others. This decision was made to harness the unique advantages of each approach, while striking a balance between optimal performance and computational efficiency.

Moreover, the design of a linear prediction model further simplifies future real-life implementation, thus enhancing the project's practical viability.

However, the framework's strongest point lies primarily in its utilization of distributed control. This design choice enhances the system's reliability significantly, eliminating the dependency on a central controller, that traditionally decides for other agents and sends commands. This absence of centralization ensures a more robust system operation with no single point of failure. Furthermore, considering its applicability in various real-world trajectory tracking missions, the method can easily accommodate different numbers of agents tailored to specific applications. Another advantage of the distributed approach is the considerably simpler optimization problem that needs to be solved by each agent, compared to the more intricate problem in a centralized approach.

Additionally, a key innovative aspect lies in the absence of predefined trajectories for individual agents. Instead, each agent dynamically navigates its path, avoiding obstacles and maintaining a safe distance from other agents without adhering to any specific formation. This dynamic adaptability sets it apart from traditional control methods and highlights the framework's robustness and agility especially in unpredictable scenarios.

The research journey encountered several significant challenges. Understanding the intricate behavior of both individual quadcopters and the collaborative dynamics within a quadcopter swarm was a formidable task. This complexity escalated when designing a framework capable of addressing these intricacies, especially in the context of complex tasks such as real-time trajectory tracking involving dynamic obstacles. Tackling these challenges necessitated meticulous tuning, experimentation, careful consideration of the system's complexities and innovative solutions. Among those, the communication strategy's approach stands out as it involves a necessary compromise; favoring the z-axis, as the MPC for altitude must be implemented first. Nevertheless, this strategy proves to be effective and functional within the context of the study.

5.2 Discussion and expansions

While this thesis offers a comprehensive and efficient examination of the control of quadcopter swarms, open matters and avenues for extension persist. The translation of simulations into tangible real-world applications of a Distributed Model Predictive Control (DMPC) framework stands as an important challenge, concerning hardware integration, scalability for scenarios demanding extensive coverage and real-time obstacle detection technologies.

Furthermore, the inclusion of terminal region constraints instead of terminal constraints within the swarm's behavior presents an intriguing avenue of exploration, as it simplifies the optimization

problem and unveils possibilities for diverse applications. Moreover, another viable path for investigation is the design of a singular MPC for the three axes, aimed to mitigate the perceived bias towards the z axis.

Finally, exploring the advantages of machine learning techniques, such as neural networks[42]–[47], presents a compelling avenue for extension. Integrating these techniques in control schemes [48][49][50] could enhance the swarm's decision-making capabilities, allowing it to adapt more effectively in unforeseen scenarios, thereby broadening its scope of application.

Bibliography – Citations – Online Sources

[1] U. Ukaegbu, L. Tartibu, and M. Okwu, "Unmanned Aerial Vehicles for the Future: Classification, Challenges, and Opportunities," in *2021 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, 2021, pp. 1–7. doi: 10.1109/icABCD51485.2021.9519367.

[2] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," 2015.

[3] M. Stogiannos, A. Alexandridis, and H. Sarimveis, "An enhanced decentralized artificial immune-based strategy formulation algorithm for swarms of autonomous vehicles," *Appl Soft Comput*, vol. 89, p. 106135, Apr. 2020, doi: 10.1016/J.ASOC.2020.106135.

[4] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of Unmanned Aerial Vehicles — A Survey," *J Ind Inf Integr*, vol. 16, p. 100106, 2019, doi: https://doi.org/10.1016/j.jii.2019.100106.

[5] I. Lopez-Sanchez and J. Moreno-Valenzuela, "PID control of quadrotor UAVs: A survey," *Annu Rev Control*, vol. 56, p. 100900, Jan. 2023, doi: 10.1016/j.arcontrol.2023.100900.

[6] E. Joelianto, D. Christian, and A. Samsi, "Swarm control of an unmanned quadrotor model with LQR weighting matrix optimization using genetic algorithm," *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, vol. 11, no. 1, pp. 1–10, 2020, doi: 10.14203/j.mev.2020.v11.1-10.

[7] Y. Kartal, K. Subbarao, N. R. Gans, A. Dogan, and F. Lewis, "Distributed backstepping based control of multiple UAV formation flight subject to time delays," 2020, doi: 10.1049/iet-cta.2019.1151.

[8] D. Yu, C. L. P. Chen, and H. Xu, "Fuzzy Swarm Control Based on Sliding-Mode Strategy With Self-Organized Omnidirectional Mobile Robots System," *IEEE Trans Syst Man Cybern Syst*, vol. 52, no. 4, pp. 2262–2274, 2022, doi: 10.1109/TSMC.2020.3048733.

[9] D. Yu, J. Long, C. L. P. Chen, and Z. Wang, "Adaptive Swarm Control Within Saturated Input Based on Nonlinear Coupling Degree," *IEEE Trans Syst Man Cybern Syst*, vol. 52, no. 8, pp. 4900–4911, Aug. 2022, doi: 10.1109/TSMC.2021.3102587.

[10] M. Papadimitrakis and A. Alexandridis, "Active vehicle suspension control using road preview model predictive control and radial basis function networks," *Appl Soft Comput*, vol. 120, p. 108646, 2022, doi: https://doi.org/10.1016/j.asoc.2022.108646.

[11] M. Papadimitrakis, M. Stogiannos, H. Sarimveis, and A. Alexandridis, "Multi-Ship Control and Collision Avoidance Using MPC and RBF-Based Trajectory Predictions," *Sensors*, vol. 21, no. 21, 2021, doi: 10.3390/s21216959.

[12] A. Alexandridis, H. Sarimveis, and K. Ninos, "A Radial Basis Function network training algorithm using a non-symmetric partition of the input space – Application to a Model Predictive Control configuration," *Advances in Engineering Software*, vol. 42, no. 10, pp. 830–837, 2011, doi: https://doi.org/10.1016/j.advengsoft.2011.05.026.

[13] A. Alexandridis and H. Sarimveis, "Nonlinear adaptive model predictive control based on self-correcting neural network models," *AIChE Journal*, vol. 51, no. 9, pp. 2495–2506, Sep. 2005, doi: 10.1002/AIC.10505.

[14] A. Kapnopoulos and A. Alexandridis, "A cooperative particle swarm optimization approach for tuning an MPC-based quadrotor trajectory tracking scheme," *Aerosp Sci Technol*, vol. 127, p. 107725, 2022, doi: https://doi.org/10.1016/j.ast.2022.107725.

[15] S. Vargas, H. M. Becerra, and J. B. Hayet, "MPC-based distributed formation control of multiple quadcopters with obstacle avoidance and connectivity maintenance," *Control Eng Pract*, vol. 121, Apr. 2022, doi: 10.1016/j.conengprac.2021.105054.

[16] G. Tartaglione, M. Ariola, E. D'Amato, and P. S. Rossi, "A 3D Decentralized Guidance and Control System for a Swarm of Multi-Copters," in *IFAC-PapersOnLine*, Elsevier B.V., Jul. 2017, pp. 5788–5793. doi: 10.1016/j.ifacol.2017.08.424.

[17] G. Franze, G. Fedele, A. Bono, and L. D'alfonso, "Reference Tracking for Multiagent Systems Using Model Predictive Control," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 4, pp. 1884–1891, Jul. 2023, doi: 10.1109/TCST.2022.3226326.

[18] F. Ahmed, J. C. Mohanta, A. Keshari, and P. S. Yadav, "Recent Advances in Unmanned Aerial Vehicles: A Review," *Arab J Sci Eng*, vol. 47, no. 7, pp. 7963–7984, Jul. 2022, doi: 10.1007/S13369-022-06738-0/FIGURES/22.

[19] T. Bresciani, "Modelling, Identification and Control of a Quadrotor Helicopter," 2008, [Online]. Available: http://www.control.lth.se/publications/

[20] K. J. Åström and T. Hägglund, "Advanced PID Control," 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:110923981

[21] W. Bolton, "Instrumentation and Control Systems, Third Edition," *Instrumentation and Control Systems, Third Edition*, pp. 1–370, Jan. 2021, doi: 10.1016/C2020-0-00286-0.

[22] E. Camacho and C. Bordons, *Model Predictive Control*, vol. 13. 2004. doi: 10.1007/978-0-85729-398-5.

[23] N. Lanzetti, Y. Z. Lian, A. Cortinovis, L. Dominguez, M. Mercangoz, and C. Jones, "Recurrent neural network based MPC for process industries," *2019 18th European Control Conference, ECC 2019*, pp. 1005–1010, Jun. 2019, doi: 10.23919/ECC.2019.8795809.

[24] A. M. Ersdal, D. Fabozzi, L. Imsland, and N. F. Thornhill, "Model Predictive Control for Power System Frequency Control Taking into Account Imbalance Uncertainty," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 981–986, Jan. 2014, doi: 10.3182/20140824-6-ZA-1003.01631.

[25] C. Diaz, A. Mazza, F. Ruiz, D. Patino, and G. Chicco, "Understanding Model Predictive Control for Electric Vehicle Charging Dispatch," *Proceedings - 2018 53rd International Universities Power Engineering Conference, UPEC 2018*, Nov. 2018, doi: 10.1109/UPEC.2018.8542050.

[26] T. Gold, A. Völz, and K. Graichen, "Model Predictive Interaction Control for Industrial Robots," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9891–9898, Jan. 2020, doi: 10.1016/J.IFACOL.2020.12.2696.

[27] M. Tiwari, E. Coyle, and R. J. Prazenica, "Direct-Adaptive Nonlinear MPC for Spacecraft Near Asteroids," *Aerospace 2022, Vol. 9, Page 159*, vol. 9, no. 3, p. 159, Mar. 2022, doi: 10.3390/AEROSPACE9030159.

[28] T. A. Johansen and A. Grancharova, "Approximate explicit constrained linear model predictive control via orthogonal search tree," *IEEE Trans Automat Contr*, vol. 48, no. 5, pp. 810–815, 2003, doi: 10.1109/TAC.2003.811259.

[29] I. Kalogeropoulos, A. Alexandridis, and H. Sarimveis, "Economic Oriented Dynamic Matrix Control of Wastewater Treatment Plants," *J Process Control*, vol. 118, pp. 202–217, 2022, doi: https://doi.org/10.1016/j.jprocont.2022.08.006.

[30] B. Li, Y. Tan, A.-G. Wu, and G.-R. Duan, "A Distributionally Robust Optimization Based Method for Stochastic Model Predictive Control," *IEEE Trans Automat Contr*, vol. 67, no. 11, pp. 5762–5776, 2022, doi: 10.1109/TAC.2021.3124750.

[31] R. Wang, I. R. Manchester, and J. Bao, "Distributed Economic MPC With Separable Control Contraction Metrics," *IEEE Control Syst Lett*, vol. 1, no. 1, pp. 104–109, 2017, doi: 10.1109/LCSYS.2017.2708660.

[32] T. Besselmann, J. Lofberg, and M. Morari, "Explicit MPC for LPV Systems: Stability and Optimality," *IEEE Trans Automat Contr*, vol. 57, no. 9, pp. 2322–2332, 2012, doi: 10.1109/TAC.2012.2187400.

[33] "What is Model Predictive Control? - MATLAB & Simulink - MathWorks Benelux." Accessed: Oct. 03, 2023. [Online]. Available: https://nl.mathworks.com/help/mpc/gs/what-is-mpc.html

[34] H. Lee, J. Lee, S.-H. Kim, S. Jung, and Y. Kim, "Adaptive Control Allocation-based Control Design for Non-affine Morphing Aircraft System," 2019, doi: 10.13009/EUCASS2019-163.

[35] D. Q. Mayne, J. B. Rawlings, C. V Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000, doi: https://doi.org/10.1016/S0005-1098(99)00214-9.

[36] R. Kalman and J. Bertram, "Control system analysis and design via the second method of lyapunov: (I) continuous-time systems (II) discrete time systems," *IRE Transactions on Automatic Control*, vol. 4, no. 3, p. 112, 1959, doi: 10.1109/TAC.1959.1104895.

[37] L. Barnes, M. Fields, and K. Valavanis, "Unmanned ground vehicle swarm formation control using potential fields," in *2007 Mediterranean Conference on Control & Automation*, 2007, pp. 1–8. doi: 10.1109/MED.2007.4433724.

[38] K. Loayza, P. Lucas, and E. Peláez, "A centralized control of movements using a collision avoidance algorithm for a swarm of autonomous agents," in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, 2017, pp. 1–6. doi: 10.1109/ETCM.2017.8247496.

[39] Z. Qin, Z. Lin, D. Yang, and P. Li, "A task-based hierarchical control strategy for autonomous motion of an unmanned surface vehicle swarm," *Applied Ocean Research*, vol. 65, pp. 251–261, 2017, doi: https://doi.org/10.1016/j.apor.2017.04.013.

[40] A. Loria, J. Dasdemir, and N. Alvarez Jarquin, "Leader–Follower Formation and Tracking Control of Mobile Robots Along Straight Paths," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 2, pp. 727–732, 2016, doi: 10.1109/TCST.2015.2437328.

[41] L. Yang, J. Jiang, X. Gao, Q. Wang, Q. Dou, and L. Zhang, "Autonomous environment-adaptive microrobot swarm navigation enabled by deep learning-based real-time distribution planning," *Nat Mach Intell*, vol. 4, no. 5, pp. 480–493, 2022, doi: 10.1038/s42256-022-00482-8.

[42] A. P. Alexandridis, C. I. Siettos, H. K. Sarimveis, A. G. Boudouvis, and G. V Bafas, "Modelling of nonlinear process dynamics using Kohonen's neural networks, fuzzy systems and Chebyshev series," *Comput Chem Eng*, vol. 26, no. 4, pp. 479–486, 2002, doi: https://doi.org/10.1016/S0098-1354(01)00785-2.

[43] D. Karamichailidou, S. Koletsios, and A. Alexandridis, "An RBF online learning scheme for non-stationary environments based on fuzzy means and Givens rotations," *Neurocomputing*, vol. 501, pp. 370–386, 2022, doi: https://doi.org/10.1016/j.neucom.2022.06.016.

[44] D. Karamichailidou, A. Alexandridis, G. Anagnostopoulos, G. Syriopoulos, and O. Sekkas, "Modeling biogas production from anaerobic wastewater treatment plants using radial basis function networks and differential evolution," *Comput Chem Eng*, vol. 157, p. 107629, 2022, doi: https://doi.org/10.1016/j.compchemeng.2021.107629.

[45] D. Kagkas, D. Karamichailidou, and A. Alexandridis, "Chess Position Evaluation Using Radial Basis Function Neural Networks," *Complexity*, vol. 2023, p. 7143943, 2023, doi: 10.1155/2023/7143943.

[46] D. Karamichailidou, V. Kaloutsa, and A. Alexandridis, "Wind turbine power curve modeling using radial basis function neural networks and tabu search," *Renew Energy*, vol. 163, pp. 2137–2152, 2021, doi: https://doi.org/10.1016/j.renene.2020.10.020.

[47] A. Alexandridis, E. Chondrodima, N. Giannopoulos, and H. Sarimveis, "A Fast and Efficient Method for Training Categorical Radial Basis Function Networks," *IEEE Trans Neural Netw Learn Syst*, vol. 28, no. 11, pp. 2831–2836, 2017, doi: 10.1109/TNNLS.2016.2598722.

[48] Alexandridis, A., M. Stogiannos, N. Papaioannou, E. Zois, H. Sarimveis, "An Inverse Neural Controller Based on the Applicability Domain of RBF Network Models", Sensors, 18(1) (2018), 315.

[49] Stogiannos, M., A. Alexandridis, H. Sarimveis, "Model predictive control for systems with fast dynamics using inverse neural models", ISA Transactions, 72 (2018), pp. 161-177.

[50] Alexandridis, A., M. Stogiannos, A. Kyriou, H. Sarimveis, "An offset-free neural controller based on approximating the inverse process dynamics" Journal of Process Control, 23(7) (2013), pp. 968–979.