الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
قسم الآلية
Ecole Nationale Polytechnique
Département d'Automatique

Laboratoire de Commande des Processus

End-of-studies project dissertation

for obtaining the State Engineer's degree in Automation and Control

Contribution to estimation-based nonlinear control design for multi-UAV systems

Realized by:

Ms. BOUHOUNALI Feriel Ms. KHEDACHE Kenza

Publicly presented and defended on the 3rd of July, 2023, in front of the jury composed of:

President	Pr. BOUKHETALA Djamel	ENP
Examiner	Dr. ACHOUR Hakim	ENP
Promoter	Pr. LADACI Samir	ENP
Co-promoter	Pr. BELKHATIR Zehor	U. Soton, UK
Guest	Dr. ILLOUL Rachid	ENP
Guest	Mr. MER ZIGHED Aziz	CRD, Algiers

ENP 2023

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
قسم الآلية
Ecole Nationale Polytechnique
Département d'Automatique

Laboratoire de Commande des Processus

End-of-studies project dissertation

for obtaining the State Engineer's degree in Automation and Control

Contribution to estimation-based nonlinear control design for multi-UAV systems

Realized by:

Ms. BOUHOUNALI Feriel Ms. KHEDACHE Kenza

Publicly presented and defended on the 3rd of July, 2023, in front of the jury composed of:

President	Pr. BOUKHETALA Djamel	ENP
Examiner	Dr. ACHOUR Hakim	ENP
Promoter	Pr. LADACI Samir	ENP
Co-promoter	Pr. BELKHATIR Zehor	U. Soton, UK
Guest	Dr. ILLOUL Rachid	ENP
Guest	Mr. MER ZIGHED Aziz	CRD, Algiers

ENP 2023

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
قسم الآلية
Ecole Nationale Polytechnique
Département Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'Ingénieur d'État en Automatique

Contribution à la conception de commandes non linéaires basées sur l'estimation pour les systèmes de multi-UAV

Réalisé par :

M^me^ BOUHOUNALI Feriel

M^me^ KHEDACHE Kenza

Présenté et soutenu publiquement le 3 Juillet 2023, devant le jury composé de :

Président	Pr. BOUKHETALA Djamel	ENP
Examinateur	Dr. ACHOUR Hakim	ENP
Promoteur	Pr. LADACI Samir	ENP
Co-promoteur	Pr. BELKHATIR Zehor	U. Soton, UK
Invité	Dr. ILLOUL Rachid	ENP
Invité	M. MER ZIGHED Aziz	CRD, Alger

ENP 2023

ملخص

يستكشف هذا العمل تقنيات التحكم والتقدير لمشكلة تتبع المسار مطبقة على نظام متعدد رباعي المحركات في سياق التحكم في تشكيل القائد-التابع. يتم التحكم بواسطة وحدة تحكم PID, واستراتيجية التحكم التنبئي غير الخطي (NMPC). يتم إجراء تقدير الحالة باستخدام مرشح كالمان الممتد (EKF). عمليات المحاكاة المكثفة تتحقق من صحة النهج المقترحة. توفر النتائج رؤى قيمة حول أداء و ملائمة أساليب التحكم هذه. تكمن مساهمتنا في ادماج النهج القائم على التنبؤ في NMPC لمشكلة التحكم في التشكيل واستخدام إطار تحسين CasADi للأنظمة متعددة رباعي المحركات.

كلمات مفتاحية : المركبات الجوية غير المأهولة، رباعي المحركات، أنظمة الطائرات بدون طيار المتعددة، التحكم في التشكيل، القائد-التابع، نموذج التحكم التنبؤي، مرشح كالمان الممتد.

Résumé

Ce travail explore les techniques de contrôle et d'estimation pour le problème de suivi de trajectoire appliqué à un système de multi-quadrirotor dans le contexte du contrôle de formation leader-suiveur. Nous introduisons un contrôleur PID, et une commande prédictive non linéaire de modèle (NMPC). L'estimation de l'état est réalisée à l'aide du Filtre de Kalman Étendu (KFE). Des simulations approfondies valident les approches proposées. Les résultats fournissent des indications précieuses sur les performances et l'adéquation de ces techniques de commande. Notre contribution réside dans l'intégration d'une approche basée sur les prédictions dans la commande NMPC pour le problème de contrôle de formation et l'utilisation de la bibliothèque d'optimisation CasADi pour les systèmes de multi-quadrirotor.

Mots clés : Véhicules aériens sans pilote, Quadrotor, Systèmes de multi-drones, Contrôle de formation, Leader-suiveur, Commande prédictive, Filtre de Kalman Étendu.

Abstract

This work explores control and estimation techniques for the trajectory tracking problem applied to a multi-quadrotor system in the context of leader-follower formation control. We introduce an optimized PID controller, and a Nonlinear Model Predictive Controller (NMPC). State estimation is done using the Extended Kalman Filter (EKF). Extensive simulations validate the proposed approaches. The results provide valuable insights into the performance and suitability of these control approaches. Our contribution lies in the integration of a prediction-based approach in NMPC for the formation control problem and the use of the CasADi optimization framework for multi-quadrotor systems.

Keywords: Unmanned Aerial Vehicles, Quadrotor, Multi-UAV systems, Formation control, Leader-follower, Model Predictive Control, Extended Kalman Filter.

Dedication

“

To my dear mother,
To my dear father,
To my grandmother,
To my brother Mounir and my sister Malak,
To all my family and friends,
To my dear Abdelghani,
To my dear friends Keltoum, Yousra, and Samy,
To my partner in this work Kenza,

”

- Feriel

“

To my dear father,
To my dear mother,
To my siblings,
To all my family and my friends,
To my dear friend and partner in this work Feriel,

”

- Kenza

Acknowledgments

We would like to express our sincere gratitude and appreciation to all those who have contributed to the completion of this thesis.

We are deeply grateful to our supervisors, Pr. LADACI and Pr. BELKHATIR for their guidance, expertise, and unwavering support throughout this research journey. Their invaluable insights, constructive feedback, and mentorship have been instrumental in shaping the direction of this work.

We would also like to extend our heartfelt thanks to the members of our thesis committee, Pr. BOUKHETALA Djamel and Dr. ACHOUR Hakim, for their evaluation and critical review of this thesis. Their expertise and feedback significantly enriches the quality of this research.

We would like to extend our heartfelt thanks to Dr. ILLOUL Rachid and Mr. MER ZIGHED Aziz, the representative of the CRD research center, for gracing us with their presence during this thesis defense. Their presence greatly enriches our discussion.

Our deepest appreciation goes to our families and friends for their unconditional love and support, and to all the people who have been by our side throughout this thesis journey.

Contents

List of Figures

List of Tables

List of Abbreviations

UAV	*Unmanned Aerial Vehicle*
GCS	*Ground Control Station*
VTOL	*Vertical Takeoff and Landing*
IMU	*Inertial Measurement Unit*
GPS	*Global Positioning System*
DCM	*Direction Cosine Matrix*
MIMO	*Multiple Input Multiple Output*
PD	*Proportional Derivative*
PID	*Proportional Integral Derivative*
FOPID	*Fractional Order Proportional Integral Derivative*
T2FNNs	*Type-2 Fuzzy Neural Networks*
LQR	*Linear Quadratic Regulator*
GA	*Genetic Algorithm*
PSO	*Particle Swarm Optimization*
ACO	*Ant Colony Optimization*
OPI	*Optimization Potential Index*
ISE	*Integral of Square Error*
IAE	*Integral Absolute Error*
ITAE	*Integral Time-weighted Absolute Error*
ISC	*Integral of Squared Control*
MPC	*Model Predictive Control*
NMPC	*Nonlinear Model Predictive Control*
OCP	*Optimal Control Problem*

NLP	*Nonlinear Programming Problem*
ODE	*Ordinary Differential Equation*
CasADi	*Computer Algebra Systems for Algorithmic Differentiation*
KF	*Kalman Filter*
EKF	*Extended Kalman Filter*
AHRS	*Attitude and Heading Reference System*
INS	*Inertial Navigation System*

Chapter 1

General introduction

1.1 Motivation and Challenges

In recent years, the field of aerial robotics has witnessed remarkable progress, with quadrotors emerging as agile and versatile Unmanned Aerial Vehicles (UAVs) [1]. These UAVs possess the potential not only to perform complex maneuvers and navigate intricate environments but also to collaborate and form coordinated formations. Considering the growing interest in utilizing UAVs for complex tasks, the development of effective control strategies for multi-UAV systems holds great importance. By coordinating the actions of multiple UAVs, we can achieve precise spatial relationships, adapt to changing environments, and execute tasks that require collective efforts. Therefore, the motivation behind investigating multi-UAV systems lies in their ability to accomplish tasks that would be challenging or impossible for a single UAV alone [2]. In the field of multi-UAV systems, significant advancements have been made in control and estimation techniques to address the unique challenges posed by these complex systems. Researchers have explored various approaches to achieve coordinated control and efficient operation of multiple UAVs. The formation control of UAVs, which involves orchestrating multiple UAVs to maintain specific spatial relationships, has become a fascinating and challenging area of research. Several control methods have been explored for UAV formation. These include the leader-follower method [3], virtual leader method [4], behavior-based method [5], and graph theory [6]. The leader-follower approach [7] is widely employed and has demonstrated effectiveness in various domains, including robotics, UAVs, and swarm robotics. Another important consideration in the context of multi-UAV systems is state estimation, which plays a crucial role in accurately determining the states of individual UAVs that are not directly measurable or to filter noisy measurements. In recent years, data fusion techniques have gained prominence in multi-UAV estimation [8], along with the Extended Kalman Filter (EKF) [9]. EKF enables accurate estimation of the states of individual UAVs, facilitating coordinated behavior and enhancing the overall performance of multi-UAV systems.

The cooperative control of UAVs presents significant challenges owing to factors such as complex, nonlinear, high order dynamic model of each UAV, interdependence between the agents, restricted information availability, and more. Addressing these complexities requires advanced control strategies. One promising control technique for the leader-follower formation problem is Nonlinear Model Predictive Control (NMPC) [10] [11]. NMPC offers several advantages, including the ability to handle nonlinear complex dynamics, such as the UAV model, incorporate constraints on the system's states and inputs, and generate optimal control actions based on a predictive model of the system. These features make NMPC a powerful tool for achieving accurate trajectory tracking of each UAV in the formation and ensure coordinated behavior of the agents. However, the interdependence between the leader and the followers poses a major challenge when incorporating the formation control approach with NMPC. This challenge arises because the NMPC algorithm requires having the references over an entire time horizon, meanwhile, the followers' references are generated based on the leader agent. We address this challenge by proposing a prediction-based leader-follower approach.

1.2 Objectives and Contributions

In this work, our objective is to address the leader-follower formation control problem and the estimation problem for a multi-quadrotor system, ie. design effective control strategies in the context of the leader-follower configuration considering partial measurements information, we aim to maintain a specific formation pattern in space. For this, we consider the leader's trajectory tracking problem first. We will be focusing on two prominent methods, optimal PD control using the Genetic Algorithm (GA) and Nonlinear Model Predictive Control (NMPC) using the multiple shooting technique to transform an Optimal Control Problem (OCP) into a Nonlinear Programming Problem (NLP). Additionally, we implement the NMPC controller in the optimization framework CasADi to improve computational efficiency. Moreover, we aim to ensure the formation keeping by the followers using a Lyapunov-based formation controller. For this, we adapt the designed PD and NMPC controllers to the followers' case to maintain a desired 'V' shape. Additionally, we addressed the estimation problem for each quadrotor in the formation using the Extended Kalman Filter (EKF) to estimate the system states based on noisy sensor measurements.

Throughout this work, one major challenge when designing NMPC type of controller for the leader-follower formation control problem is that the NMPC algorithm requires having the references of each agent in the system over the whole horizon. Therefore, the generation of references for the followers' over the whole horizon at each instant posed a major challenge. This challenge arose because the followers' references solely depend on the state of the leader, meaning the problem required having the leader's state over the whole horizon, however, that information is not available at each instant. Our contribution to address this challenge involves the introduction of an original prediction-based formation control approach in the leader-follower configuration, where we used the leader's state predictions in the formation controller to generate the followers' references. Moreover, the prediction-based scheme is combined with EKF estimation for all agents in the formation. The proposed approach showcased its remarkable ability to generate coordinated control actions for maintaining desired formations. However, the implementation of NMPC for formation control posed notable challenges due to its inherent computational demands. To overcome this obstacle, we leveraged the capabilities of CasADi, a symbolic framework for dynamic optimization, which allowed us to effectively reduce the computational time while seamlessly integrating prediction-based techniques tailored specifically for formation control.

1.3 Organization of the thesis

This work is organized into several chapters to provide a systematic and comprehensive exploration of the proposed formation control problem. The following outlines the organization of the thesis:

This chapter serves as an introduction to the topic, providing an overview of the motivations, objectives, and contributions of this work. This chapter also presents the scope of the study and outlines the organization of the subsequent chapters.

Chapter 2 focuses on the background and state-of-the-art for UAV systems. It provides an exploration of the motivation behind UAV use and their various applications. The classification of UAV systems is discussed, along with an examination of multi-UAV systems and their different classifications. We also review the literature on control and estimation strategies for UAV systems. This chapter provides a comprehensive foundation for understanding the field of UAV and multi-UAV systems.

In chapter 3, the mathematical modeling of quadrotor systems takes center stage. The Newton-Euler representation and state-space representation for a single quadrotor are introduced, along with an analysis of the modeling challenges and structure of multi-quadrotor systems for formation control. This chapter lays the groundwork for understanding the dynamics and behavior of quadrotor systems.

Chapter 4 delves into the control and estimation for a single quadrotor. First we employ an optimized PD controller using the Genetic Algorithm. Then, Nonlinear Model Predictive Control (NMPC) is introduced, by formulating an Optimal Control Problem (OCP) and discretizing it using the multiple shooting technique, resulting in a Nonlinear Programming Problem (NLP), which is then solved using the CasADi optimization framework. Additionally state estimation is discussed using the Extended Kalman Filter (EKF). Moreover, a comparison between PD and NMPC control techniques is highlighted, showcasing their respective merits, trade-offs, and implications for tracking accuracy, response time, and computational complexity.

In chapter 5, the formation control of multi-quadrotor systems is investigated, particularly the leader-follower approach. The previously developed optimal PD and NMPC techniques are adapted to the followers' case. For the NMPC controller, we propose a prediction-based approach in the formation controller to ensure the formation keeping by the followers. The integration of formation control with EKF is explored for each quadrotor in the formation, leading to coordinated behavior among UAVs.

Finally, this work concludes with chapter 6, which presents a summary of the research findings. This section provides a concise yet thorough overview of the key outcomes obtained throughout the study. Additionally, the chapter delves into an exploration of potential future directions, paving the way for further investigations and advancements in the field.

Chapter 2

Background & state-of-the-art for UAV systems

2.1 Introduction

Unmanned Aerial Vehicles (UAVs) have attracted substantial interests from the control engineering research community in the recent years, this is due to their use in both military and civilian operations [2] and their various applications in many fields such as surveillance, search and rescue [12], environmental monitoring, mapping [13], and more. In some of these applications, it is advantageous to combine many UAVs to work together in a coordinated manner to achieve a common goal which leads to an improved efficiency, increased coverage, and greater flexibility in terms of task assignment compared to a single UAV, making it a more efficient solution for many applications.

An important amount of research has been done on the control of multi-agent systems which arise numerous challenges especially when it comes to the coordination between the different agents. Formation control, which is a key component in the domain of multi-agent systems, deals with the coordination of multiple autonomous agents, it aims to get these agents to form a specific pattern in space. To achieve this goal, numerous algorithms and control strategies have been studied in literature that enable agents to communicate, plan and execute their trajectories while minimizing communication delays and collision risks [14] [15], ensuring stability, robustness, and scalability of the overall system.

One widely used approach of formation control is the leader-follower approach [16]. In this approach, one or more agents are designated as leaders, while the rest are considered followers. This approach has been applied in various fields, such as robotics, UAVs, autonomous underwater vehicles, and swarm robotics. It has shown promising results in achieving complex tasks that are impossible to be completed by a single agent.

This chapter will cover the fundamentals of UAVs within the framework of multi-UAV systems. Particularly, our focus will be on motivations, applications, various classifications of both single UAVs and multi-UAV systems, as well as a literature review on UAV control and estimation. We will place particular emphasis on leader-follower formation control, which serves as the central topic of this work.

2.2 Unmanned Aerial Vehicles

UAVs are defined as type of aircraft that operates without a human pilot on board and that can be remotely controlled or can fly autonomously. Currently, a wide range of aircraft types are employed, such as fixed-wing aircraft, helicopters, and airships. However, the one platform that has received unmatched amount of attention is the multi-rotor helicopter, especially the quadrotor, because it is more advantageous over other aircrafts with a simple structure and a great flight capacity and maneuverability.

2.2.1 Motivation and applications

UAVs have captured the attention of researchers and garnered significant interest in recent years, owing to their potential to revolutionize a wide range of fields and their involvement in numerous real-world applications [2]. Their versatility has led to their use in various commercial purposes, including aerial photography and videography, delivery services,

and environmental monitoring. In the industrial sector, drones carry out missions in hostile environments that are inaccessible to humans, optimizing processes and ensuring safety. Additionally, drones serve multiple functions in transportation, contributing to traffic monitoring, package delivery, and many more.

Equipped with onboard sensors, UAVs facilitate efficient data collection across diverse environments. This capability enables access to a wide range of data types, including temperature, humidity, and geographic information, proving valuable in monitoring and tracking ecosystems, such as meteorological extreme events and climate change.

As UAV technology continues to advance, new applications are constantly being discovered and explored. From agricultural monitoring to disaster response, from search and rescue missions [12] to scientific research, drones are poised to reshape numerous industries and make a lasting impact on society. The popularity of UAVs can be attributed to their essential features, such as mobility, easy deployment, flexibility, and versatile usage. These characteristics make them a preferred choice for various applications and contribute to their continued growth and adoption across industries offering a wide range of benefits and paving the way for innovative solutions in the future.

2.2.2 Classification of UAVs

UAVs can be classified based on several factors such as shape, size, speed, reached altitude, operational range, flight endurance, among others [1]. We can summarize these different classifications as shown in Fig. 2.1.

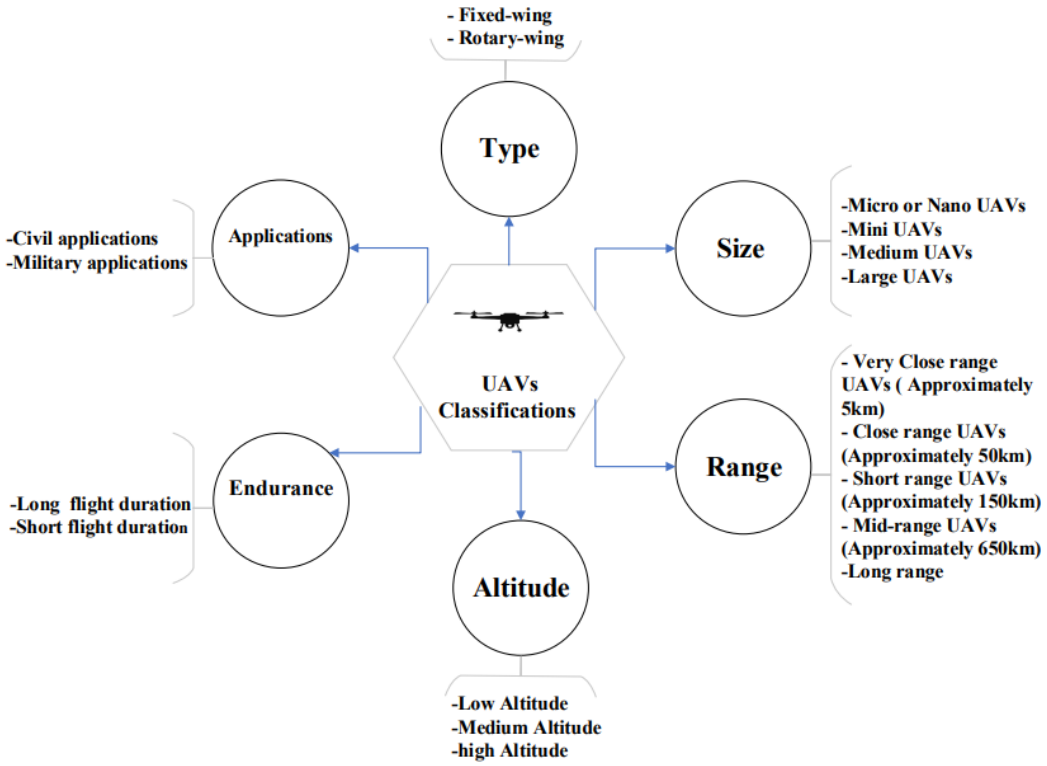

Figure 2.1: Single UAV classifications, from [1]

2.2.2.1 Classification based on the shape of the UAV

There are mainly four types of UAVs based on the shape:

- **Single-rotor UAVs**: also known as helicopters, have the capability to vertically take off and land (VTOL). They utilize a main rotor for controlling attitude and a tail rotor for directional control. Their key advantage lies in their ability to carry heavy payloads over extended flight durations. However, the mechanical complexity, large size, and high cost of the rotor system pose risks for unmanned versions.
- **Multi-rotor UAVs**: feature more than two rotors. This category includes birotors, trirotors, quadrotors, hexarotors, and octocopters. Multi-rotor UAVs enable vertical takeoff and landing. They possess high maneuverability and agility, allowing them to perform intricate maneuvers and navigate confined spaces. However, their main drawback is the limited flight time compared to other types of aircraft.
- **Fixed-wing UAVs**: are characterized by a simple structure consisting of a rigid and fixed wing. Fixed-wing UAVs have the capacity to carry heavier payloads compared to multi-rotor UAVs. However, they are less agile during flight, restricting their ability to perform complex maneuvers or operate in confined spaces. Additionally, they require a runway for takeoff and landing.
- **Hybrid UAVs**: combine features of both fixed-wing and multi-rotor UAVs. They offer a balance between agility and speed. They are capable of carrying larger payloads and do not necessitate a runway for operations. Their main drawbacks are their price, intricate mechanics, reduced flight stability, and limited speed range.

The aforementioned UAV classes are shown in Figures 2.2-2.5.

Figure 2.2: Single-rotor (helicopter)

Figure 2.3: Multi-rotor (quadrotor)

Figure 2.4: Fixed-wing UAV

Figure 2.5: Hybrid UAV

2.2.2.2 Classification based on the size of the UAV

We can break it up into the following categories:

- **Very small UAVs:** they include micro and nano-UAVs, are characterized by their resemblance to insects or birds, typically ranging between 5 and $50cm$, these UAVs are equipped with extremely small and lightweight components. They have a top speed of over $10km/h$.
- **Small UAVs:** often referred to as mini-UAVs, typically ranging from $50cm$ up to $2m$. These UAVs are commonly based on the fixed-wing model and are launched by propelling them into the air. They can carry a maximum payload of $9kg$ and fly at speeds of around $150km/h$, usually at altitudes below $400m$.
- **Medium UAVs:** are characterized by their weight, they are usually too heavy for a single person to handle but smaller than airplanes given their payload capacity limitation of $200kg$. Typically falling within the fixed-wing UAV category, these aircraft have wingspans ranging from 5 to $10m$. They can reach maximum velocities of $463km/h$ without exceeding altitudes of around $1km$.
- **Large UAVs:** primarily designed for military purposes, offer extensive range and endurance. They are typically based on fixed-wing structures, enabling them to carry heavy payloads over long distances while reaching maximum altitudes of up to $5.5km$.

Among the different cited types of UAVs, small size UAVs are the most popular for several reasons. Firstly, they are cost-effective, making them more accessible to a wider range of users. Secondly, their lightweight and compact design enhances portability, enabling easy transport and deployment. Additionally, small UAVs offer a higher level of safety, minimizing the potential harm to people or property in the event of accidents or malfunctions, which makes them suitable for indoor and urban environments. Lastly, their maneuverability and ease of maintenance make them particularly well-suited for scientific research and learning purposes.

2.2.3 Literature review on single UAV control and estimation

There is a wealthy literature on the control and estimation techniques for UAVs. A wide variety of control techniques and estimation algorithms have been applied to UAVs to stabilize these highly unstable systems, achieve satisfying performances, and accurate state estimation. We can classify some of the control and estimation strategies applied on UAVs as follows:

- PID control: from the control problem viewpoint, the most widely used technique is PID control due to its simplicity and efficiency [17]. For instance:
  - A PID and PD controllers have been implemented and compared in [18] to choose the proper controller.
  - In [19], a PID controller is combined with EKF to reject the measurement noise from the IMU sensor.

  - An adaptive neuro PID controller is presented in [20] to achieve stable performance of a quadcopter.
  - Numerous variations of PID controller have also been developed to enhance the transient performance, including inner-outer loop structures [21] and the PD^2 feedback structure in [22].
  - Another popular variation for enhancing PID performance and dealing with uncertainties is the Fractional Order PID (FOPID) in [23] [24].
  - The optimal tuning of PID parameters has also been widely discussed. Numerous heuristic approaches have been employed such as the Genetic Algorithm (GA) [25], Particle Swarm Optimization (PSO) [26], and Ant Colony Optimization (ACO).

- Nonlinear control techniques: the nonlinear nature of the control problem has resulted in significant advances in nonlinear control techniques [27] [28], we can cite:
  - Sliding mode control is considered a strong strategy due to its robustness in the face of parametric uncertainties and disturbances [29] [30] [31].
  - The authors of [32] propose a Lyapunov-based backstepping controller that guides a quadrotor along a predefined trajectory.
  - A comparison of sliding mode control and backstepping techniques can be found in [28] and [33]. On the other hand, [34] enhances the sliding mode approach by combining it with backstepping techniques.
  - The research [35] compares the accuracy of trajectory tracking between type-1 and type-2 fuzzy neural networks (T2FNNs).
  - Another interesting control method is the feedback linearization technique [36] [37] [38], which has demonstrated success in controlling nonlinear systems.
  - Reinforcement learning techniques have also been widely used for this purpose, neural controllers have proven their ability to control complex systems [39].

- Optimal control techniques: another type of the widely discussed control techniques in the literature is optimal control. There are different types of optimal control methods, such as:
  - Linear Quadratic Regulator (LQR) is a popular optimal control method employed in UAV control systems [40].
  - H^∞ is a robust control technique that aims to achieve desired performance in the presence of uncertainties, disturbances, and system modeling errors [41].
  - Model Predictive Control (MPC), which has emerged as a promising control technique for quadrotor systems in recent years [42] [43]. Its appeal stems from its ability to handle highly nonlinear MIMO systems. Extensive research has demonstrated that Nonlinear MPC (NMPC) outperforms linear MPC, particularly when dealing with nonlinear dynamics. This approach holds significant relevance for quadrotors due to their inherently complex and nonlinear dynamics. However, the implementation of NMPC in real-time applications presents computational challenges due to the demanding nature of the internal plant model's calculations. To overcome this issue, various approaches have been

developed, including the explicit MPC method [44], this technique involves solving the optimization problem offline and storing the resulting optimal parameters. As a result, solving the optimization problem at each sampling time is avoided, enabling real-time implementation of NMPC. To further enhance the real-time performance of NMPC, fast optimizers are also employed [45].

- To implement these different control strategies, state estimation is crucial to obtain the full information about the state vector of UAV systems from noisy sensor measurements. The Extended Kalman Filter (EKF) is one of the most popular and widely studied estimation techniques for nonlinear dynamic systems. Various works delve into this topic, offering valuable insights. For instance:
  - The authors of [46] focus on implementing an EKF-based on a drag force enhanced model, this EKF enables the estimation of all the states of a quadrotor, including the unknown drag coefficients.
  - Meanwhile, [47] concentrate on estimating only the attitude of the quadrotor using the IMU sensor.
  - The authors in [48] present a vision-based EKF approach for tracking a target.
  - EKF finds significant applications in sensor fusion scenarios where inaccurate measurements need to be effectively filtered. This technique combines multiple sensor inputs to enhance the accuracy of state estimation. In [49], EKF is combined with MPC, to fuse data obtained from an IMU, a sonar, and an optic-flow based vision system. By leveraging the EKF estimation, the study achieves improved estimation accuracy by accounting for the strengths and limitations of each sensor modality.
  - Another study that delves into sensor fusion is presented in [50], where the authors focus on developing different strategies for state vector estimation by fusing measurements from IMU, GPS, and distance measurement sensors using EKF estimation.

2.3 Multi-UAV systems

In nature, collective movements such as flocks of birds, swarms of bees, or colonies of bacteria are commonly seen. A fascinating occurrence is that several entities with limited intelligence and size can make spectacular coordinated movements, this phenomenon has attracted widespread attention from researchers in many areas. This kind of motion is not only visually satisfying but also possesses immense potential in military, industrial, and civilian applications [2]. From there comes the idea of a multiple UAV system involving the deployment of several UAVs in a coordinated manner to achieve a common goal. Moreover, it enables efficient utilization of resources, increased coverage, and improved safety compared to a single UAV system.

2.3.1 Motivation and applications

Despite the capabilities of a single UAV system, its operational tasks are often limited, hindering the system from achieving its full operational potential and widespread application. While a single UAV can perform certain functions, such as small payload transportation or localized surveillance, it may struggle with complex missions that require extensive coverage, large-scale object searching, or heavy payload transportation [51].

Recognizing the potential applications and challenges, researchers are increasingly drawn to explore the possibilities of utilizing multiple UAVs, commonly called swarms, in cooperative missions. The effective collaboration and synchronization of multiple UAVs can construct a more efficient system that harnesses the collective capabilities of the UAVs, surpassing the performance of a single UAV.

By working together, a system composed of multiple UAVs can achieve a higher level of operational functionality and provide broader applications. The cooperative efforts of multiple UAVs enable enhanced capabilities, such as coordinated surveillance over large areas, efficient distribution of heavier payloads, and the ability to tackle complex tasks that would be challenging for a single UAV. The synchronized operation of multiple UAVs not only improves efficiency but also enhances overall system reliability and robustness, making it an attractive approach for various applications ranging from disaster response and environmental monitoring [52] to military operations and search-and-rescue missions [12]. In agriculture, multi-UAV are utilized for tasks like crop mapping, plant health assessment, and fertilizer application. For delivery services, multi-drone systems are being developed to overcome urban traffic challenges and enable efficient package delivery. In the construction sector, swarm drones assist with tasks such as surveying [53], site inspections, and creating 3D maps. In the energy and utilities industry, multi-drone systems are employed for inspecting pipelines [54], power lines [55], and wind turbines. In entertainment, swarm drones are used to create captivating light shows and performances. Additionally, swarm UAVs find applications in military operations, scientific research for tasks such as data collection, wildlife monitoring [56], and oceanographic surveys. These examples merely scratch the surface of the vast potential and continuous development of swarm drones' applications. An example of multi-UAV cooperation is shown in Fig. 2.6.

Figure 2.6: Multiple-UAV cooperation, from [7]

2.3.2 Classification of multi-UAV systems

2.3.2.1 Coordination and cooperation

In a platform involving multiple vehicles, coordination and cooperation are crucial. Coordination involves sharing resources and requires both temporal (synchronization) and spatial coordination (sharing of space). Spatial coordination is based on path planning algorithms but can become challenging when the number of vehicles is high. Cooperation refers to joint behavior directed towards a common goal and requires integration of sensing, control, and planning.

A coordinated multi-UAV system relies on several essential components to function effectively. Firstly, the system comprises the UAVs themselves, which form the swarm. Secondly, a Ground Control Station (GCS) serves as the central point of control, managing command transmission to the drones. The GCS can consist of either a single computer or a cluster of computing nodes. Thirdly, a communication system, equipped with devices and antennas, allows communication between the UAVs and the GCS using a common protocol. Various technologies such as Mobile Ad-Hoc Network (MANET) [57], Flying Ad-Hoc Network (FANET) [58], and Vehicular Ad-Hoc Network (VANET) [59] can be employed for this purpose. The system also requires a navigation system to enable the UAVs to fly and determine their position within the environment. This system incorporates sensors like GPS and Inertial Measurement Units (IMUs). Furthermore, a control system facilitates drone control and coordination from the ground station. It encompasses software for mission planning, decision making, and swarm behavior. Lastly, a data processing system enables real-time data processing at the GCS and facilitates feedback to the control system. This system may include tools for image processing, data analysis, and machine learning. Together, these components collaborate to enable the multi-UAV swarm to function as a unified entity. The GCS assumes overall command and control, while the UAVs work together to achieve shared objectives.

2.3.2.2 Classification based on spatial relations

In this section, we will focus on classification based on spatial relations as shown in Fig. 2.7, a fully detailed classification can be found in [60].

- **Physical coupling:** it concerns UAVs that are connected by physical links and have motions constrained by forces that depend on each other. The main focus is on motion-coordinated control while taking into account the force constraints.

- **Formations:** the UAVs that are not physically coupled but have their relative motions strongly constrained to maintain a formation. The motion planning problem can be addressed by considering the formation as a whole. Collision avoidance within the team can be incorporated into the formation control strategy.

- **Swarms:** it covers teams of many vehicles that display emerging collective behaviors due to their interactions. The motion of these vehicles may not necessarily result in formations. The main challenge of swarms is scalability, given the large number of vehicles involved.

- **Intentional cooperation:** in this scenario, the UAVs in the team move along individual task-defined trajectories to fulfill a global mission. These trajectories are not related geometrically as they are in the formation case.

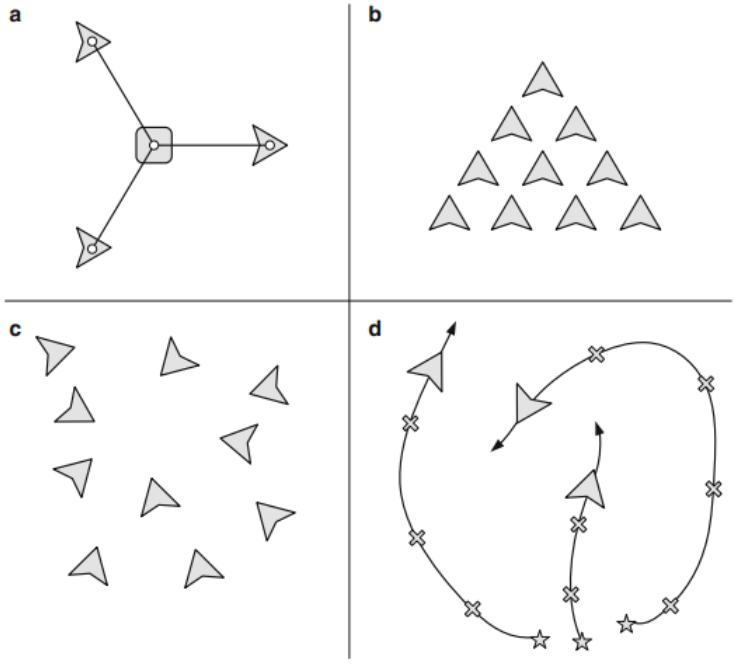

Figure 2.7: Multi-UAV spatial relations: (a) physical coupling, (b) formations, (c) swarms, and (d) intentional cooperation, from [61]

2.3.2.3 Classification of formation control

In formation control, multiple UAVs fly in a particular pattern, the aim of formation control is to ensure that each UAV follows a desired trajectory while maintaining a specified inter-vehicle distance and orientation. If the formation inter-distances are variable, its called a flexible formation. In contrast, if the inter-distances remain constant, it is known as a rigid formation. The control strategies for formation control are:

- **Behavioral approach:** several desired behaviors are defined for the agents. These may include behaviors such as maintaining cohesiveness, avoiding collisions, and avoiding obstacles, among others [5]. This approach is linked to the concept of amorphous formation control, as discussed further below.
- **Virtual structure approach:** the group of agents is treated as a single entity, referred to as a virtual structure. The desired movement of the virtual structure is specified, and the movements of the individual agents are then derived from it [4].
- **Leader–follower approach:** one or more agents acts as a leader and the remaining agents are considered as followers. The followers maintain a fixed positional offset from the leader, following its movements. Meanwhile, the leader is responsible for tracking its designated trajectory [3]. In this configuration, the states of the leader serve as the coordination variable, as the actions of the followers in the formation can be fully determined once the leader states are known. One major advantage of this

approach is its simplicity and scalability. The followers only need to communicate with the leader agents, which reduces the communication and computation burden. Moreover, this approach can handle a wide range of formation patterns by adjusting the relative positions and orientations of the followers with respect to the leaders.

The leader-follower approach mainly consists of two configurations, a standard and an interactive configuration. The standard configuration operates in a way that allows the leader to influence the followers when it is within their neighboring set. However, no feedback is considered from the followers to the leader [62], treating the leader as a special agent with independent motion. This approach offers advantages in terms of efficiency and energy saving. However, a drawback of the standard leader-follower approach is its lack of robustness, particularly in cases of leader failure. To address this limitation, in [63], a switchable multiple leaders concept is introduced to enable the formation to persist even in the event of leader failure. Additionally, in [64], a weighted neighbor-based formation method is presented, it exhibits greater robustness compared to the anonymous neighbor-based formation. For the interactive leader-follower configuration, in [65], [66] the leader has knowledge of the objective command for the group of agents, while the remaining agents are interconnected either with each other or with the leader through a predetermined topology. The followers having the capability to interact with the leader enables bidirectional communication.

Another classification of formation control can be based on the presence or absence of explicit prescriptions for desired formation shapes:

- **Morphous formation control:** the desired formations are established by explicitly specifying various factors, including the desired positions of the agents, inter-agent distances, and inter-agent displacements.
- **Amorphous formation control:** when desired formations are not explicitly defined, the desired behaviors, such as maintaining cohesiveness and avoiding collisions, are provided for the agents.

2.3.2.4 Classification based on communication architectures

In the leader-follower approach, centralized, decentralized, and distributed communication architectures can be employed. The choice of the adequate architecture depends on the system's complexity and the dynamic environment it operates in.

- **Centralized control strategy** [67] involves a central entity, such as a central computer or a GCS, that has complete knowledge and control over the leader and all the followers. The central processor is responsible for generating commands and transmitting them to the agents. It collects data from all agents, and determines the desired formation and actions for each agent. The centralized control strategy enables precise coordination and synchronization among the agents but relies heavily on the central entity for decision-making, which can introduce a single point of failure. The standard leader-follower formation configuration is perceived as a centralized method, where the leader operates independently from the followers. This architecture is illustrated in Fig. 2.8.

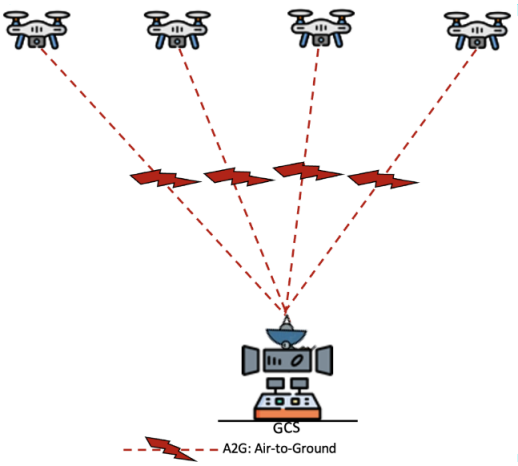

Figure 2.8: Multi-UAV centralized communication architecture, from [68]

- **Decentralized control strategy** [61] distributes the decision-making process among the leader and follower agents themselves. Each agent has its own local sensing, computation, and decision-making capabilities. The leader agent may provide high-level guidance or objectives to the followers, but the followers make their own decisions based on local information and interactions with neighboring agents. Decentralized control reduces the reliance on a central entity and allows for more autonomous behavior of the agents increasing the robustness of the overall system. However, achieving consensus and coordination among the agents can be more challenging in a decentralized control strategy due to the absence of a centralized authority.

- **Distributed control strategy** [68] it evolves from the decentralized control approach by incorporating the sharing of local information. It offers advantages over centralized control, particularly in situations where data delays occur. Distributed control is closely associated with decentralized control and the management of large-scale systems. Researchers have proposed distributed control strategies to address communication challenges within the decentralized control. By incorporating communication aspects, distributed control strategies enhance the coordination and performance of leader-follower formations in a more robust and efficient manner.

2.3.3 Literature review on formation control of multi-UAV systems

There are various research methods for UAV formation control, such as the leader-follower method [3], virtual leader method [4], behavior-based method [5], and graph theory [6]. We can cite the following works:

- In [14] and [15], different methods such as optimization control, graph theory, guidance, and potential field have been utilized to address formation control problems.

- In their work, the authors of [69] introduced a leader-follower formation control scheme that employs two different controllers. The first one is a PD controller utilized to maintain the desired formation shape. The second one is based on fuzzy logic for achieving the desired formation pattern.

- The study [70] introduced a controller for synchronized position tracking based on a PI control law for a pair of UAVs. The proposed approach ensures the preservation of the formation shape by synchronizing the positions of the agents.
- Another study in [71] presented an alternative leader-follower formation control scheme for a quadcopter group. In their approach, the authors employed a prescribed performance control method to simultaneously achieve the desired formation pattern and formation trajectory.
- The author in [72] conducted research on formation keeping and reconstruction. A distributed feedback controller was created and its effectiveness was tested. The formation reconstruction problem was also transformed into a fuel optimal control.
- In [73], the authors designed a robotic formation-keeping strategy based on the leader-follower control method.
- A feedback controller is designed in [74] for each UAV and a distributed overlapping control technique is employed to shape the dispersed UAV formation into the target formation.
- The author in reference [75] introduced a multi-UAV formation control technique that uses a hierarchical mechanism and MPC. Although the method is highly precise in controlling the UAV formation, it has a weak real-time performance.
- The researchers in [76] aimed to preserve the formation of a high-order disturbed multi-UAV system. They proposed combining the sliding mode technique and the adaptive neural network method.
- In [77], a formation control strategy that utilizes deep reinforcement learning to avoid collision is presented.
- Another recent study in [78] utilized a distributed backstepping technique to control the formation of multiple UAV systems. In another study [79], the backstepping technique was used again to control a swarm formation of UAVs along a specific circular path. The designed technique takes into account parameters and input constraints, and adjusts itself adaptively.
- In [80], a predictive model was designed using the event-triggered method to control a multi-UAV system.
- Finally, [16] proposes a hybrid controller that is a combination of the PID and the adaptive fuzzy controller with integral feedback.

2.4 Conclusion

In conclusion, this chapter delved into the state-of-the-art research on UAVs, multi-UAV systems and formation control, with a particular focus on the leader-follower configuration. Throughout the chapter, several key aspects were discussed, including the motivation and applications of both single UAV and multi-UAV systems, their different classifications, as well as the different control and estimation strategies existing in literature. An analysis

of the limitations of single UAV systems was conducted, demonstrating the necessity for multi-UAV systems and emphasizing their crucial role in overcoming these limitations and unlocking new possibilities in various applications.

In the next chapter, the mathematical modeling of single UAV and multi-UAV systems is carried out, establishing a solid understanding of the underlying dynamics of these systems and gaining valuable insights into the behavior and interactions of UAVs, ultimately paving the way for advanced control and optimization techniques.

Chapter 3

Mathematical modeling of quadrotor systems

3.1 Introduction

The quadrotor has emerged as one of the most popular UAV platforms in recent years. Its unique design and maneuverability have made it a versatile choice for various applications. In this chapter, we will explore the modeling aspects and the underlying principles and equations that govern the movement of both single quadrotors and multi-quadrotor systems, laying the foundation for understanding their behavior and control in leader-follower formation scenarios.

To accurately represent and simulate the quadrotor's dynamics, various modeling approaches can be employed. These include Euler angles, Direction Cosine Matrix (DCM), and quaternions. Each representation has its advantages and trade-offs in terms of computational efficiency, singularity avoidance, and ease of implementation. In this study, we will adopt the Euler angles representation for its simplicity and intuitive understanding, allowing us to describe the quadrotor's attitude (pitch, roll, and yaw angles).

In the context of leader-follower formation control, modeling a multi-quadrotor system involves capturing the dynamics and interactions between the leader and follower UAVs. By considering the individual quadrotor models and incorporating coordination and constraints between the different agents, we can simulate the collective behavior and formation control of multiple quadrotors. This modeling approach enables the investigation of formation configuration, trajectory tracking, and cooperative tasks among the UAVs within a leader-follower framework.

In the subsequent sections, we will delve deeper into the mathematical modeling of single quadrotors using Euler angles representation. We will also explore the extension of these models to capture the dynamics of multi-quadrotor systems in the context of leader-follower formation control.

3.2 Mathematical modeling of a single quadrotor

A quadrotor is a type of UAV that features four vertically oriented rotors, each providing thrust to generate lift and control the vehicle's motion in the three-dimensional space. Any movement of the quadrotor can be achieved by changing the angular rates of its rotors [19], the quadrotor can generate the required lift and perform desired pitch, roll, and yaw maneuvers. Increasing and decreasing the speed of all rotors leads the quadrotor to ascend and descend vertically. Yaw rate is obtained by changing the velocity of the rotors (1,3) or (2,4) shown in Fig 3.1. Pitch rate is achieved by altering the speed balance of rotors 1 and 3. Change in pitch angle then leads to longitudinal acceleration. Similarly, roll rate is obtained like pitch rate, the only difference is that rotors 2 and 4 are used instead rotors 1 and 3. Changing the roll angle leads to lateral acceleration. When all of the rotors have same velocity, the overall moment produced is zero and the quadrotor is holding its attitude. When an appropriate velocity is set up, then the rotors generate counteracting thrust to balance the force of gravity, enabling the quadrotor to maintain its altitude. The position when attitude and altitude are kept unchanged is called hover.

3.2.1 Newton-Euler representation

To model the 6-DOF quadcopter, Newton-Euler formalism will be used. We need to consider two reference frames: an earth inertial frame ($Oxyz$) and a body fixed frame ($O_b x_b y_b z_b$) shown in Fig 3.1.

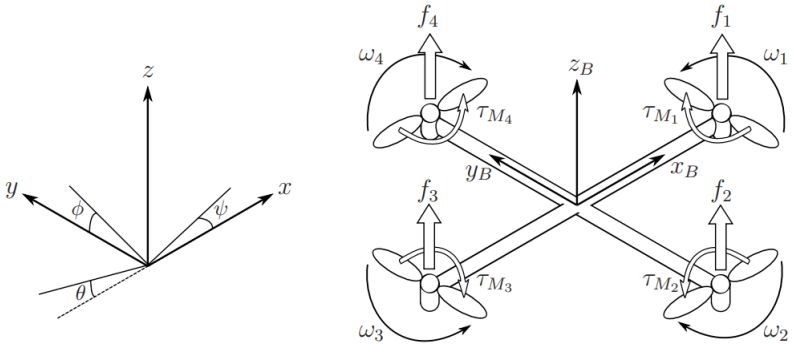

Figure 3.1: Reference frames for the quadrotor, from [81]

The quadrotor operates based on six degrees of freedom (6-DOF), encompassing translational and rotational motion [19]. therefore twelve states are needed to describe it. The first six states represents the attitude and its change. These are the roll ϕ, pitch θ and yaw ψ Euler angles $\eta = [\phi\, \theta\, \psi]^T$ (between body-fixed frame and the earth-fixed frame) and the angular velocities in the body frame $w = [p\, q\, r]^T$. The remaining six states are the coordinates of the center of mass of the quadrotor in the earth frame expressed as $r_{xyz} = [x\, y\, z]^T$ and the linear velocities in the earth frame $V = \dot{r}_{xyz} = [\dot{x}\, \dot{y}\, \dot{z}]^T$.

To simplify the modeling and analysis of the quadrotor system, the following assumptions are made [7] [82]:

- The structure and the propellers are rigid.
- The structure is symmetrical about the axes.
- The body frame origin coincides with the center of gravity of the quadrotor.
- We do not take into account the dynamics of rotors and propellers.
- The aerodynamic forces and gyroscopic effects are not taken into account.
- The parameters of the quadrotor are supposed to be known and constant.

The rotation matrices with respect to the yaw, pitch and roll angles are given by:

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -sin\psi & 0 \\ \sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}, \tag{3.2}$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}. \tag{3.3}$$

The rotation matrix from the body frame to the inertial frame is given by the sequence roll-pitch-yaw [7]:

$$R = R_z(\psi)R_y(\theta)R_x(\phi), \tag{3.4}$$

$$R = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - c\phi s\psi & s\psi s\phi + c\psi c\phi s\theta \\ c\theta s\psi & c\psi c\phi + s\psi s\theta s\phi & c\phi s\psi s\theta - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}, \tag{3.5}$$

where: $c \triangleq cos$ and $s \triangleq sin$.

Note that $R^{-1} = R^T = R_x(\phi)R_y(\theta)R_z(\psi)$ is the matrix defining the transform from the inertial frame to the body frame. The importance of the R matrix stems from the fact that certain states are measured in the body frame while others are measured in the earth frame, thus a transform is needed.

The quadrotor's motion can be broken into two subsystems, a rotational subsystem with respect to the Euler angles that is fully actuated and a translational subsystem with respect to the x, y and z positions that is underactuated.

Newton's second law and dynamics equations of the quadrotor can be described in vector form as following:

$$F = m\frac{dV}{dt}, \tag{3.6}$$

$$M = \frac{dH}{dt}, \tag{3.7}$$

where m is the quadrotor's mass, V is the linear velocity vector in the earth frame and M is the moment the quadrotor. H is the angular momentum of the quadrotor relative to the inertial frame. It is essential to emphasize that the translational dynamics of the quadrotor are represented in the earth frame, while the rotational dynamics are modeled in the body frame. In the body-fixed frame, the inertia matrix I is diagonal, simplifying the calculation of rotational dynamics compared to the earth frame, where I is non-diagonal and time-variant.

3.2.1.1 Translational subsystem equation

Starting with the translational subsystem, the equation (3.6) in the earth frame becomes:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}. \tag{3.8}$$

The total thrust is the sum of the forces generated by the four propellers $T = T_1 + T_2 + T_3 + T_4$. If we neglect the aerodynamic forces, the external forces acting on the quadrotor's body consist of the total thrust T exerted by the propellers and the gravitational force. Thrust always acts along the body's z axis therefore it is projected according to

quadrotor's attitude while the gravitational force is always acting in the earth's frame upward-pointing z axis [7]. The acceleration due to gravity is g.

$$\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}, \tag{3.9}$$

$$\begin{aligned} \ddot{x} &= (\sin\psi \sin\phi + \cos\psi \cos\phi \sin\theta)\frac{T}{m}, \\ \ddot{y} &= (\cos\phi \sin\psi \sin\theta - \cos\psi \sin\phi)\frac{T}{m}, \\ \ddot{z} &= (\cos\theta \cos\phi)\frac{T}{m} - g. \end{aligned} \tag{3.10}$$

3.2.1.2 Rotational subsystem equation

For the rotational subsystem, in equation (3.7), since the angular momentum vector alters its direction, we need to apply the total derivative of the vector H.

$$M = \dot{H} + w \times H, \tag{3.11}$$

$$H = Iw, \tag{3.12}$$

where w represents the quadrotor's attitude change, while I denotes its moment of inertia. Since the quadrotor is a symmetric rigid body about its xz and yz planes, with rotation axes coinciding with the principal axes, its moment of inertia matrix is diagonal and is given by:

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}, \tag{3.13}$$

where I_x, I_y and I_z are the moments of inertia with respect to the principle axes in the body frame.

$$M = I\dot{w} + w \times (Iw), \tag{3.14}$$

after replacing the angular velocities in the body frame $w = [p\, q\, r]^T$ and expanding the cross product we get:

$$\begin{aligned} M_\phi &= I_x\dot{p} + qr(I_z - I_y), \\ M_\theta &= I_y\dot{q} + pr(I_x - I_z), \\ M_\psi &= I_z\dot{r} + pq(I_y - I_x), \end{aligned} \tag{3.15}$$

where M_ϕ, M_θ and M_ψ are the roll, pitch and yaw moments [82]. Therefore, the Euler rates in the body frame are given by:

$$\begin{aligned} \dot{p} &= \frac{M_\phi}{I_x} - \frac{qr}{I_x}(I_z - I_y), \\ \dot{q} &= \frac{M_\theta}{I_y} - \frac{pr}{I_y}(I_x - I_z), \\ \dot{r} &= \frac{M_\psi}{I_z} - \frac{pq}{I_z}(I_y - I_x). \end{aligned} \tag{3.16}$$

The transform between angular rates in the earth frame to body frame is given by:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = E \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \tag{3.17}$$

$$E = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}, \tag{3.18}$$

the change of attitude in the earth frame (for $\theta \neq \pm\frac{\pi}{2}$ so that E is invertible) is given by:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = E^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \tag{3.19}$$

$$E^{-1} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix}. \tag{3.20}$$

Considering four identical motors and no motor dynamics, for each propeller, the total thrust T is proportional to the square of its rotating velocity Ω_i with respect to the coefficient k_T [82]. The three moments M_ϕ, M_θ, M_ψ are also proportional to the square of Ω_i with respect to the coefficient k_τ as follows:

$$\begin{bmatrix} T \\ M_\phi \\ M_\theta \\ M_\psi \end{bmatrix} = \begin{bmatrix} k_T & k_T & k_T & k_T \\ 0 & -k_T l & 0 & k_T l \\ -k_T l & 0 & k_T l & 0 \\ k_\tau & -k_\tau & k_\tau & -k_\tau \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \tag{3.21}$$

where l is the length of the arm of the quadrotor. The inverse rotational velocities inputs are:

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4k_T} & 0 & \frac{-1}{2k_T l} & \frac{1}{4k_\tau} \\ \frac{1}{4k_T} & \frac{-1}{2k_T l} & 0 & \frac{-1}{4k_\tau} \\ \frac{1}{4k_T} & 0 & \frac{1}{2k_T l} & \frac{1}{4k_\tau} \\ \frac{1}{4k_T} & \frac{1}{2k_T l} & 0 & \frac{-1}{4k_\tau} \end{bmatrix} \begin{bmatrix} T \\ M_\phi \\ M_\theta \\ M_\psi \end{bmatrix}, \tag{3.22}$$

we know that the inputs of the quadrotor are the four angular velocities of its propellers Ω_i. However, in this work, we don't transform the thrust and moment inputs into angular velocities.

Finally we get:

$$\begin{aligned}
\ddot{x} &= (\sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta)\frac{T}{m},\\
\ddot{y} &= (\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi)\frac{T}{m},\\
\ddot{z} &= (\cos\theta\cos\phi)\frac{T}{m} - g,\\
\dot{\phi} &= p + \sin\phi\tan\theta q + \cos\phi tan\theta r,\\
\dot{\theta} &= \cos\phi q - \sin\phi r,\\
\dot{\psi} &= \frac{\sin\phi}{\cos\theta}q + \frac{\cos\phi}{\cos\theta}r,\\
\dot{p} &= \frac{M_\phi}{I_x} - \frac{qr}{I_x}(I_z - I_y),\\
\dot{q} &= \frac{M_\theta}{I_y} - \frac{pr}{I_y}(I_x - I_z),\\
\dot{r} &= \frac{M_\psi}{I_z} - \frac{pq}{I_z}(I_y - I_x).
\end{aligned} \tag{3.23}$$

3.2.2 State-space representation

In this section, the state-space representation of the quadrotor's nonlinear model is presented with respect to the earth frame. This representation serves the purpose of designing controllers for the quadrotor system.

We consider the state vector $X = [x_1, ..., x_{12}]^T \in \mathbb{R}^{12}$ given as follows:

$$X = [x \quad y \quad z \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad \phi \quad \theta \quad \psi \quad p \quad q \quad r]^T, \tag{3.24}$$

which contains the linear positions and velocities in the earth-fixed frame, the angular positions in the earth-fixed frame and angular velocities in the body-fixed frame.

We also consider the input vector $U \in \mathbb{R}^4$:

$$U = [u_1 \quad u_2 \quad u_3 \quad u_4]^T = [T \quad M_\phi \quad M_\theta \quad M_\psi]^T. \tag{3.25}$$

Thus, the quadrotor model can be written as follows:

$$\dot{X} = F(X, U), \tag{3.26}$$

where the nonlinear function $F(X, U) : \mathbb{R}^{12} \times \mathbb{R}^4 \to \mathbb{R}^{12}$ is locally Lipschitz.

The desired motion of a quadrotor can be completely specified by the vector Y containing the $3D$ positions x, y and z and the yaw angle ψ. As a result, the objective of the trajectory planning problem is to determine the vector Y^{ref} that corresponds to this desired motion [7].

$$Y = h(X) = [x \quad y \quad z \quad \psi]^T = [x_1 \quad x_2 \quad x_3 \quad x_9]^T, \tag{3.27}$$

$$Y^{ref}(t) = [x^{ref}(t) \quad y^{ref}(t) \quad z^{ref}(t) \quad \psi^{ref}(t)]^T. \tag{3.28}$$

On the other hand, the measurement vector Z contains the measured states using sensors [83], its gonna be used with the Extended Kalman Filter (EKF) for state estimation. More details about this can be found in section 4.4.3.

$$Z = h^{mes}(X) = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T. \tag{3.29}$$

Finally, the state-space representation is given by the following set of equations:

$$
\begin{aligned}
\dot{x_1} &= x_4, \\
\dot{x_2} &= x_5, \\
\dot{x_3} &= x_6, \\
\dot{x_4} &= (\sin x_9 \sin x_7 + \cos x_9 \cos x_7 \sin x_8)\frac{u_1}{m}, \\
\dot{x_5} &= (\cos x_7 \sin x_9 \sin x_8 - \cos x_9 \sin x_7)\frac{u_1}{m}, \\
\dot{x_6} &= (\cos x_8 \cos x_7)\frac{u_1}{m} - g, \\
\dot{x_7} &= x_{10} + \sin x_7 \tan x_8 x_{11} + \cos x_7 tan x_8 x_{12}, \\
\dot{x_8} &= \cos x_7 x_{11} - \sin x_7 x_{12}, \\
\dot{x_9} &= \frac{\sin x_7}{\cos x_8} x_{11} + \frac{\cos x_7}{\cos x_8} x_{12}, \\
\dot{x}_{10} &= \frac{u_2}{I_x} - \frac{x_{11}x_{12}}{I_x}(I_z - I_y), \\
\dot{x}_{11} &= \frac{u_3}{I_y} - \frac{x_{10}x_{12}}{I_y}(I_x - I_z), \\
\dot{x}_{12} &= \frac{u_4}{I_z} - \frac{x_{10}x_{11}}{I_z}(I_y - I_x).
\end{aligned}
\tag{3.30}
$$

This model serves as the fundamental entity for the subsequent section focusing on multi-quadrotor systems. It establishes the basis upon which the analysis and understanding of the multi-quadrotor system will be built.

3.3 Mathematical modeling of multi-quadrotor system for formation control

Formation control refers to cooperative control where agents are required to maintain a specific topology and relative distances to their neighbors. As part of a team, each action executed by a drone can impact the overall performance of the team. We will be focus on leader-follower formation control as described in subsection 2.3.2.3. In this work, we consider a rigid formation where the desired pattern is fixed. For comprehending the fundamentals of multi-quadrotor systems modeling, the subsequent definition elucidates the concept of leader-follower formation control.

Definition 3.1 (Leader-follower formation task)
Consider a group of n quadrotors, where each quadrotor ith in the group is described by the dynamical model in equation (3.30). Within this group, certain quadrotors are designated as leaders, while the remaining ones are followers. A formation task in the context of a multi-quadrotor system with a leader-follower configuration is characterized by a designated formation trajectory provided to the leader quadrotor and a desired geometric pattern that defines the inter-distance and orientation between neighboring quadrotors within the quadrotor group.

The objective is to design the control vector $U_i = [u_{1i} \quad u_{2i} \quad u_{3i} \quad u_{4i}]^T$ for each quadrotor i that result in a stable formation. With one leader $i = L$ and $n-1$ followers, $i \in \{F_1, ...F_{n-1}\}$, the i^{th} quadrotor is described by the set of equations (3.31).

$$\begin{cases} \dot{X}_i = F(X_i, U_i), \quad i \in \{L, F_1, F_2, ...F_{n-1}\}, \\ Y_i = h(X_i), \\ Z_i = h^{mes}(X_i), \end{cases} \tag{3.31}$$

we assume that all the quadrotors in the formation have the same mass m and inertia matrix I. We also assume that the i^{th} follower knows the current state of the leader at each instant t [7].

The position and yaw angle of the leader are denoted as $(x_L(t), y_L(t), z_L(t), \psi_L(t))$ and the followers' position and yaw angle are denoted as $(x_i(t), y_i(t), z_i(t), \psi_i(t))$, for $i \in \{F_1, ..., F_{n-1}\}$. For the leader, the output vector is $Y_L = h(X_L)$ and it tracks its desired trajectory $Y_L^{ref}(t)$, where:

$$Y_L = h(X_L) = [x_L \quad y_L \quad z_L \quad \psi_L]^T, \tag{3.32}$$

$$Y_L^{ref}(t) = [x_L^{ref}(t) \quad y_L^{ref}(t) \quad z_L^{ref}(t) \quad \psi_L^{ref}(t)]^T. \tag{3.33}$$

The formation configuration is specified by a desired inter-distance and orientation between the leader and each follower. Therefore, the leader's states are used to generated reference velocities for the followers to keep the desired formation pattern.

For the i^{th} follower, the output vector is $Y_i = h(X_i)$, for $i \in \{F_1, ..., F_{n-1}\}$, and it tracks its desired trajectory $Y_i^{ref}(t)$, where:

$$Y_i = h(X_i) = [x_i \quad y_i \quad z_i \quad \psi_i]^T, \tag{3.34}$$

$$Y_i^{ref}(t) = [x_i^{ref}(t) \quad y_i^{ref}(t) \quad z_i^{ref}(t) \quad \psi_i^{ref}(t)]^T. \tag{3.35}$$

The reference $Y_i^{ref}(t)$ is solely generated based on the leader's states. The followers are then controlled to keep the formation pattern. More details about this are discussed in section 5.2.1.

The measurement vector is the same for all quadrotors:

$$Z_i = h^{mes}(X_i) = [x_i \quad y_i \quad z_i \quad \phi_i \quad \theta_i \quad \psi_i]^T \quad i \in \{L, F_1, F_2, ...F_{n-1}\}. \tag{3.36}$$

3.4 Analysis of models' structure and challenges

3.4.1 Single quadrotor system case

The quadrotor model is a high-order dyanmic model, inherently unstable [84], severely nonlinear, has multiple inputs and multiple outputs (MIMO), is under-actuated with six degrees of freedom and only four control inputs, its state variables are strongly coupled which makes the control of this system a difficult task. An analysis of the quadrotor's model stability, controllability and observability can be found in [84] [85] [86]. The quadrotor model is controllable and observable considering the measurement vector described in equation (3.29).

It is worth noting that the control of the z dynamic is less challenging compared to the x and y dynamics, the z position can be directly controlled using one loop that generates the thrust input, however, x and y positions are coupled with the roll ϕ and pitch θ angles.

3.4.2 Multi-quadrotor system case

Besides the complex dynamics of a multi-quadrotor system, it also presents several other difficulties and challenges that need to be addressed for successful operation. Firstly, coordinating the movements and behaviors of multiple quadrotors is a complex task. It involves managing various aspects, including maintaining the desired formation, avoiding collisions and obstacles, and executing synchronized maneuvers. Achieving these objectives necessitates the implementation of advanced control algorithms and communication protocols to handle the complexity of the task effectively. Secondly, the interplay between the quadrotors' dynamics and their interaction with the environment poses another challenge. Each quadrotor's motion can be influenced by external disturbances and model uncertainties. These effects can affect the stability and overall system performance, demanding robust control strategies and sensor fusion techniques for accurate state estimation. Another challenge lies in designing efficient and scalable communication systems for inter-quadrotor coordination. Establishing reliable and low-latency communication links among the quadrotors is crucial for sharing information, coordinating actions, and achieving cooperative tasks.

3.5 Conclusion

Throughout this chapter, we presented the single quadrotor model using Euler angles representation, as well as the multi-quadrotor system model in the context of leader-follower formation control. Furthermore, a full analysis of the models' structures and difficulties is given for the single and multi-quadrotor case which gives us insights into their dynamics, kinematics, and control challenges. In the next chapter, we will present controllers design for single quadrotor systems.

Chapter 4

Control and estimation for a single quadrotor

4.1 Introduction

The main objective of this chapter is to synthesize control strategies for trajectory tracking in 3D space of a single quadrotor. Specifically, we focus on the design of an optimized Proportional-Derivative (PD) controller using the Genetic Algorithm (GA) and a Nonlinear Model Predictive Controller (NMPC), coupled with the Extended Kalman Filter (EKF) for state estimation. We validate these approaches on the model of the quadrotor developped in section 3.2.

The quadrotor system represents a complex and nonlinear dynamic platform, posing significant challenges in control design. Traditional PD controllers, while widely used, may struggle to yield optimal performance due to the quadrotor's intricate dynamics. To address these limitations, we explore the application of the Genetic Algorithm for optimizing the PD controller parameters to improve its performance. Additionally, we investigate the application of NMPC, which allows for the explicit consideration of system dynamics and constraints in the control formulation, enabling real-time optimization for trajectory tracking and stability. Furthermore, accurate estimation of the quadrotor's state variables is crucial for effective control implementation. In this regard, we incorporate EKF to estimate the quadrotor's states based on noisy sensor measurements.

Moreover, to assess the effectiveness of the proposed control approaches, we conduct a comprehensive comparison between the two control techniques, namely the optimized PD and the NMPC controllers. This comparative analysis aims to evaluate the strengths, weaknesses, and overall performance of each control approach. By examining their respective control capabilities, tracking accuracy, and computational efficiency, we can gain valuable insights into the suitability of these control strategies for quadrotor applications.

4.2 Optimal PD control of a quadrotor

In this section, we employ a PD controller for trajectory tracking of a desired reference of the 6-DOF quadrotor. For successful maneuvering of the quadrotor, six PD controllers are designed to control the attitude (ϕ, θ, and ψ) and position (x, y, and z) of the quadrotor. For this quadrotor model, we avoid the integrator part of the PID controller due to the presence of a double integrator within the system [87]. The PD gains are tuned using the Genetic Algorithm in order to minimize an objective function and optimize the performance of the closed-loop system [25].

4.2.1 Cascade control

A quadrotor consists of six degrees of freedom $(x, y, z, \phi, \theta, \psi)$ with only four control inputs (u_1, u_2, u_3, u_4). Due to this under-actuation, the x and y positions can't be directly controlled. This problem is resolved using two cascaded loops [7] which rely on the inherent coupling between the (ϕ, θ) attitude and the (x, y) positions, as we can observe in the quadrotor's model, given in equation (3.30), the translational subsystem depends on the rotational subsystem. Hence, to move in the x direction, the quadrotor needs to adjust its orientation by pitching downward, creating a horizontal force through the propellers' thrust, while keeping its altitude constant. Likewise, to move horizontally in

the y direction, the quadrotor must modify its attitude by rolling either to the right or left. The outer loop generates references for rotational variables based on translational variables [82]. Then, we control the attitude variables in the inner loop. Therefore, the inner loop dynamics need to be faster than the outer loop. The control scheme is shown in Fig. 4.1.

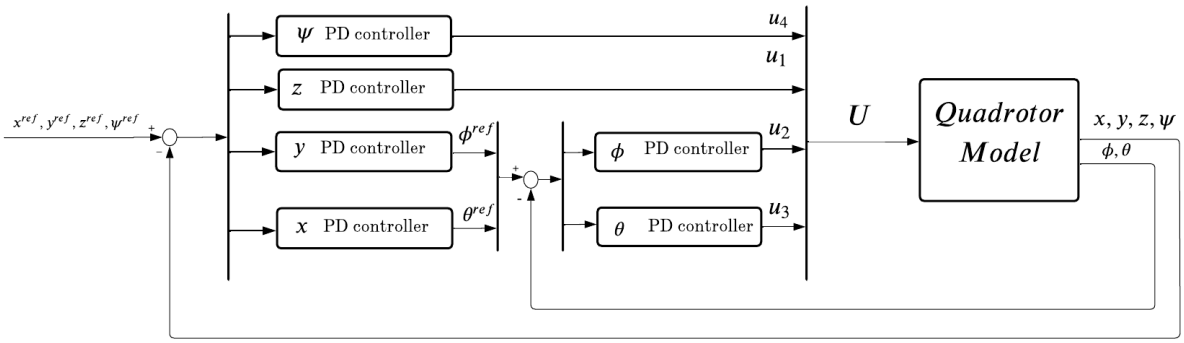

Figure 4.1: Quadrotor PD control scheme

4.2.1.1 Outer loop for translational subsystem

The outer loop determines the desired pitch and roll angles $(\theta^{ref}, \phi^{ref})$ based on the desired x and y positions (x^{ref}, y^{ref}). Considering the symmetry of the quadrotor, the control design for x and y positions is identical.

We first compute the x and y position's tracking errors in earth frame then we transform them into the body frame because the input vector U is defined in the body frame, this is shown in Fig 4.2.

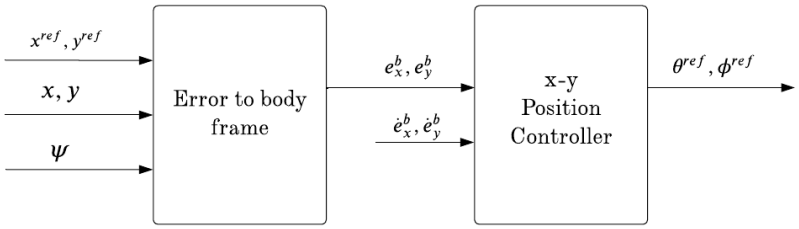

Figure 4.2: $x - y$ position controller

$$e_x(t) = x^{ref}(t) - x(t), \tag{4.1}$$

$$e_y(t) = y^{ref}(t) - y(t), \tag{4.2}$$

$$\begin{bmatrix} e_x^b \\ e_y^b \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix}, \tag{4.3}$$

where e_x, e_y are the x and y tracking errors in the earth frame while e_x^b, e_y^b are the tracking errors in the body frame. This outer loop position controller consists of two PD controllers that take the e_x^b, e_y^b errors and their derivatives, and generates the ϕ and θ references ($\phi^{ref}(t)$ and $\theta^{ref}(t)$ respectively).

$$\phi^{ref}(t) = k_{Py}e_y^b(t) + k_{Dy}\dot{e}_y^b(t), \tag{4.4}$$

$$\theta^{ref}(t) = k_{Px}e_x^b(t) + k_{Dx}\dot{e}_x^b(t), \tag{4.5}$$

where k_{Py} and k_{Dy} are the PD gains of the y axis while k_{Px} and k_{Dx} are the PD gains of the x axis.

4.2.1.2 Inner loop for rotational subsystem

The inner loop controls the quadrotor's rotational subsystem (ϕ, θ), it has two PD controllers that generate the roll and pitch moments u_2 and u_3.

$$u_2 = k_{P\phi}e_\phi(t) + k_{D\phi}\dot{e}_\phi(t), \tag{4.6}$$

$$e_\phi(t) = \phi^{ref}(t) - \phi(t), \tag{4.7}$$

$$u_3 = k_{P\theta}e_\theta(t) + k_{D\theta}\dot{e}_\theta(t), \tag{4.8}$$

$$e_\theta(t) = \theta^{ref}(t) - \theta(t), \tag{4.9}$$

where $k_{P\phi}$ and $k_{D\phi}$ are the PD gains of the roll controller while $k_{P\theta}$ and $k_{D\theta}$ are the PD gains of the pitch controller. Due to the symmetry of the quadrotor, we choose $k_{Px} = k_{Py}$, $k_{Dx} = k_{Dy}$, $k_{P\phi} = k_{P\theta}$, $k_{D\phi} = k_{D\theta}$.

4.2.1.3 Altitude and yaw control

The altitude PD controller takes the z tracking error and its derivative and generates the necessary thrust input after compensating the gravitational effect. The resulted thrust is in the earth frame, the necessary conversions are done to get the thrust in the body frame.

$$u_1 = \frac{mg + k_{Pz}e_z(t) + k_{Dz}\dot{e}_z(t)}{\cos\phi\cos\theta}, \tag{4.10}$$

$$e_z(t) = z^{ref}(t) - z(t), \tag{4.11}$$

similarly for the yaw controller:

$$u_4 = k_{P\psi}e_\psi(t) + k_{D\psi}\dot{e}_\psi(t), \tag{4.12}$$

$$e_\psi(t) = \psi^{ref}(t) - \psi(t), \tag{4.13}$$

where k_{Pz} and k_{Dz} are the PD gains of the z axis while $k_{P\psi}$ and $k_{D\psi}$ are the PD gains of the yaw controller. It is very convenient to put limitations on some control signals like pitch and roll angles limit ($|\phi| < \frac{\pi}{2}$, $|\theta| < \frac{\pi}{2}$) as well as saturations on the input effort U.

Despite the PD controller effectiveness and ability to achieve satisfying results, its performance depends mainly on tuning the gains. Tuning its constants manually can be laborious, time consuming, and requires a very good understanding of the system dynamics. To solve this issue, the Genetic Algorithm is employed to tune the gains of the six PD controllers and find their optimal values.

4.2.2 Background on the Genetic Algorithm

Genetic Algorithms (GAs) are a type of optimization and search algorithms used to solve complex constrained and unconstrained optimization problems [25], they are heuristic search algorithms that are categorized under the broader family of evolutionary algorithms, drawing inspiration from the principles of genetics and natural selection observed in biological evolution, introduced by John Holland at the University of Michigan in the United States in the 1970s. GAs involve iteratively modifying a population of individual solutions. In each iteration, certain individuals are selected from the current population as parents and are utilized to generate the offspring for the subsequent generation. Across multiple successive generations, the population "evolves" to converge towards an optimal solution.

GAs start without any prior knowledge of the optimal solution and rely solely on feedback from their environment, as well as evolutionary operators like reproduction, crossover, and mutation, to reach the best solution [81]. By initiating the search from multiple independent points and conducting parallel search, the algorithm converges towards suboptimal solutions and avoids local minimas. GAs prove particularly useful in solving optimization problems that are not well suited for standard algorithms such as problems with objective functions that are discontinuous, nondifferentiable, highly nonlinear or stochastic.

4.2.2.1 Initial population

The population can be viewed as a collection of points within the search space. Each individual within the population is represented by a chromosome which can be a sequence of bits. The degree of adaptation or fitness of an individual is determined by an objective function. In a GA, the population size is typically set to include several hundreds of potential solutions. The initial population is usually generated randomly, encompassing the entire range of possible solutions within the search space. In some cases, the solutions may be strategically "seeded" in specific regions where optimal solutions are more likely to be found.

4.2.2.2 Fitness score

Each individual is assigned a fitness score, which reflects their competitive capability. The goal is to seek individuals with optimal or near-optimal fitness scores. The GA maintains a population consisting of a fixed number of individuals (chromosomes or solutions) along with their corresponding fitness scores [81]. The selection process favors individuals with better fitness scores, who mate and produce offspring with improved genetic combinations inherited from their parents. As the population size remains constant, space needs to be created for new individuals. Consequently, some individuals die and are replaced by newcomers, eventually forming the next generation once all mating opportunities in the previous population are exhausted. It is anticipated that over successive generations, better solutions will emerge while the least fit individuals die.

4.2.2.3 Operators of Genetic Algorithms

GAs employ three main sets of rules at each iteration to generate the subsequent generation from the current population [88]:

- **Selection rules:** these rules select the individuals with the best qualities, referred to as parents, who will contribute to the population in the next generation. Selection is typically stochastic and may take into account the scores or fitness of the individuals.
- **Crossover rules:** these rules involve combining the chromosomes of two parents to create children for the next generation. The crossover process often involves exchanging genetic information between parents to generate offspring with a combination of traits from both parents.
- **Mutation rules:** they introduce random changes to individual parents to generate children. These random alterations are meant to introduce diversity and explore new areas of the solution space.

The different variations of GAs can be identified based on the type of codification utilized for the chromosomes and the specific genetic operators employed. The Algorithm can be summarized as follows:

Algorithm 1 Genetic Algorithm

Input: Initial *population*
Output: Best *individuals*
1 Evaluate *fitness* of the *individuals*
for *population size* **do**
2 Select best *individuals* from *population* as *parents*
Generate new *individuals* using *mutation* and *crossover*
Evaluate *fitness* of the new *individuals*
Replace the worst *individuals* of the *population* by the new *individuals*
3 **end**

4.2.3 Genetic Algorithm for PD tuning

Discussing the differences between the GA and traditional PD tuning methods such as Ziegler–Nichols [89] can help us understand why GA is often considered more efficient. GA differ from traditional search and optimization techniques in the following main aspects:

- **Parallel population search:** unlike traditional methods that start from a single point and iteratively explore the solution space, GAs simultaneously search a population of points which allows for a broader exploration of the solution space.
- **No derivative or auxiliary knowledge requirement:** GAs rely solely on the objective function and corresponding fitness levels to guide the search process.

- **Probabilistic transition rules:** GAs employ probabilistic transition rules instead of deterministic rules [88]. These rules, such as selection, crossover, and mutation, introduce randomness into the search process.
- **Multiple potential solutions:** GAs often provide multiple potential solutions to a given problem instead of producing a single optimal solution. The final choice of the solution is left to the user.

4.2.3.1 Multi-objective optimization

We conduct an unconstrained multi-objective optimization problem. Each PD controller has two tuning parameters; they are tuned using GA to optimize a multi-objective Optimization Potential Index (OPI). Different performance indexes have been developed to evaluate dynamic performances of controllers based on factors like the tracking error of the closed-loop system or the control energy. These indexes are time-integral criteria that are used in the objective function of optimization algorithms. Many criterion can be used such as Integral Squared Error (ISE), Integral Absolute Error (IAE), Integral Time-weighted Absolute Error (ITAE), and the Integral Squared Control (ISC). It's worth noting that the choice of the performance index depends on the specific control objectives and system requirements. In our application, we chose the the ISE and ISC indices to satify our control objectives which are minimizing the tracking error and control energy, they are given by:

Performance index	Description	Expression
ISE	Integral Squared Error	$J_{ISE} = \int_0^\infty e^2(t)dt$
ISC	Integral Squared Control	$J_{ISC} = \int_0^\infty u^2(t)dt$

Table 4.1: Performance indices

We run the GA using a multi-objective function that is a weighted sum of the ISE and ISC indices for each PD controller. We aim to make the quadrotor follow a desired trajectory subject to minimum tracking error and minimum control energy specifications. We employ a stop criterion based on the maximum number of generations. The PD tuning scheme using GA is shown in Fig 4.3.

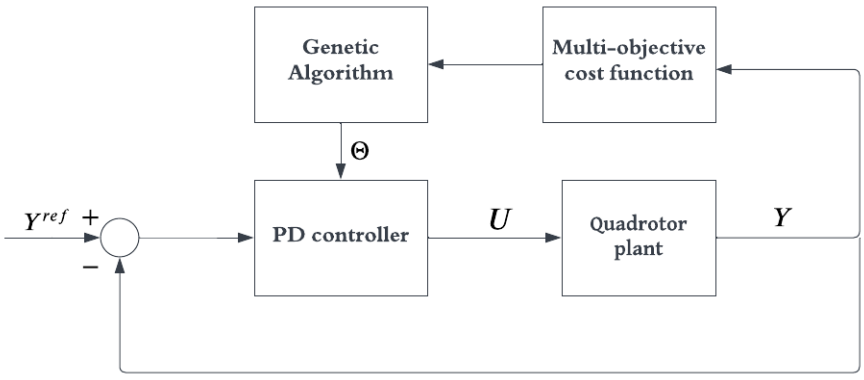

Figure 4.3: GA-based PD control

The GA is run offline, therefore, there are no computational restrictions. The total cost function in equation (4.14) is given by the sum of the cost functions of each PD controller. The GA is initialised randomly, but it is possible to choose its initial population. The resulted parameters represent a local optimum of the cost function $J_G(\Theta)$.

$$\underset{\Theta}{minimize}\ J_G(\Theta) = J_x + J_y + J_z + J_\phi + J_\theta + J_\psi, \tag{4.14}$$

where $\Theta = [k_{Px}\ k_{Dx}\ k_{Py}\ k_{Dy}\ k_{Pz}\ k_{Dz}\ k_{P\phi}\ k_{D\phi}\ k_{P\theta}\ k_{D\theta}\ k_{P\psi}\ k_{D\psi}]^T$ is the vector of parameters to be optimized. The optimization problem is twelve-dimensional, therefore, there is no guarantee that it is a convex problem (non-linear model, saturation).

The cost functions for each axis are a weighted sum of ISE and ISC indices:

$$J_x = \int_0^{t_f} w_1 (e_x^b(t))^2 dt, \tag{4.15}$$

$$J_y = \int_0^{t_f} w_1 (e_y^b(t))^2 dt, \tag{4.16}$$

$$J_z = \int_0^{t_f} [w_1 e_z^2(t) + w_2 u_1^2(t)] dt, \tag{4.17}$$

$$J_\phi = \int_0^{t_f} [w_1 e_\phi^2(t) + w_2 u_2^2(t)] dt, \tag{4.18}$$

$$J_\theta = \int_0^{t_f} [w_1 e_\theta^2(t) + w_2 u_3^2(t)] dt, \tag{4.19}$$

$$J_\psi = \int_0^{t_f} [w_1 e_\psi^2(t) + w_2 u_4^2(t)] dt, \tag{4.20}$$

where t_f is the final simulation time. w_1, w_2 are the ISE and ISC optimization weights respectively, they must satisfy $w_1 + w_2 = 1$, their values are selected to increase the emphasis on each control objective. We take the output states (x, y, z and, ψ) errors in the optimization process because the position of the quadrotor is specified by these variables in 3D space. Then, we add ϕ and θ errors to enhance the performance of the controller.

4.3 Nonlinear Model Predictive Control of a quadrotor

In this section, we apply Nonlinear Model Predictive Control (NMPC) for trajectory tracking of the 6-DOF quadrotor. We begin by providing an overview of optimal control theory and the different approaches to solving an Optimal Control Problem (OCP). Specifically, we focus on the direct multiple shooting method, which enables the reformulation of an OCP into a Nonlinear Programming Problem (NLP). Then, we introduce the principles of NMPC and outline the formulation of the NMPC optimization problem, which falls within the category of OCPs. Given the highly nonlinear nature of the quadrotor model, the NMPC-based controller demands significant computational resources. To address this, we introduce the CasADi optimization framework, which offers the advantage of faster computational time.

4.3.1 Background on Optimal Control

In dynamic optimization problems, referred to as Optimal Control Problems (OCPs), the primary objective is to find an optimal control input that minimizes a specified cost function or performance measure while adhering to a set of constraints. These constraints encompass both system dynamics and limitations imposed on the states and controls of the system. In the subsequent sections, we discuss the essential elements of a an OCP.

4.3.1.1 System dynamics model

The system dynamics are described with a mathematical model given by a set of differential algebraic equations having the form:

$$\dot{X}(t) = F(X(t), U(t)), \tag{4.21}$$

where $X(t) \in \mathbb{R}^n$, $U(t) \in \mathbb{R}^m$, and $F(X, U)$ is continuously differentiable with respect to the X and U arguments for $t \in [t_0, t_f]$. The system model is considered an equality constraint that needs to be satisfied at each instant.

4.3.1.2 Objective functional

In an OCP, a cost functional is used to quantify the performance of the control strategy. It represents the objective to be optimized in the control problem. The cost functional typically incorporates control objectives such as minimizing (or maximizing) certain system variables, tracking trajectories, and minimizing energy consumption. In this work, we use the minimum tracking error and minimum control effort criteria. The Bloza cost function is given by:

$$J(X(t), U(t), t) = E(X(t_f), U(t_f), t_f) + \int_{t_0}^{t_f} L(X(t), U(t), t)dt, \tag{4.22}$$

E is called Mayer term and L is called Lagrange term. In our application L is chosen as:

$$L(X(t), U(t), t) = \left\|X(t) - X^{ref}(t)\right\|_Q^2 + \|U(t))\|_R^2, \tag{4.23}$$

$$L(X(t), U(t), t) = [X(t) - X^{ref}(t)]^T Q[X(t) - X^{ref}(t)] + U^T(t)RU(t), \tag{4.24}$$

where X^{ref} is the desired trajectory, and Q and R are positive definite weighting matrices for the states and the control inputs respectively.

4.3.1.3 Constraints and boundary conditions

The optimal solution must adhere to a set of constraints and boundary conditions both on the control inputs and the system states. We can consider both equality and inequality constraints defined as:

$$g_1(X(t), U(t)) = 0, \quad \forall t \in [t_0, t_f], \tag{4.25}$$

$$g_2(X(t), U(t)) \leq 0, \quad \forall t \in [t_0, t_f], \tag{4.26}$$

and boundary conditions:

$$\begin{aligned} U(t) &\in \mathcal{U}, \quad \forall t \in [t_0, t_f], \\ X(t) &\in \mathcal{X}, \quad \forall t \in [t_0, t_f], \end{aligned} \tag{4.27}$$

$\mathcal{U}$ and $\mathcal{X}$ are the sets defined by the bounds on the inputs and states respectively:

$$\begin{aligned} \mathcal{U} &= \left\{U \in \mathbb{R}^m \mid U^{min} \leq U \leq U^{max}\right\}, \\ \mathcal{X} &= \left\{X \in \mathbb{R}^n \mid X^{min} \leq X \leq X^{max}\right\}, \end{aligned} \tag{4.28}$$

the initial state is also considered a boundary condition:

$$X(t_0) = X_0. \tag{4.29}$$

4.3.1.4 OCP formulation

Based on the definitions provided earlier, an OCP is generally formulated by seeking to minimize an objective functional, the controls and states that minimize it must adhere to the model equations, path constraints, and boundary conditions. Hence, the constrained OCP can be formulated as follows:

$$\begin{aligned} \underset{\mathbf{U(t)},\mathbf{X(t)}}{minimize} \quad & J = E(X(t_f), U(t_f), t_f) + \int_{t_0}^{t_f} L(X(t), U(t), t)dt, \\ s.t. \quad & \dot{X}(t) = F(X(t), U(t)), && \forall t \in [t_0, t_f], \\ & g_1(X(t), U(t)) = 0, && \forall t \in [t_0, t_f], \\ & g_2(X(t), U(t)) \leq 0, && \forall t \in [t_0, t_f], \\ & X(t_0) = X_0, && \forall t \in [t_0, t_f], \\ & U(t) \in \mathcal{U}, && \forall t \in [t_0, t_f], \\ & X(t) \in \mathcal{X}, && \forall t \in [t_0, t_f]. \end{aligned} \tag{4.30}$$

The optimal trajectory $X^*(t)$ corresponds to the optimal input $U^*(t)$ and we obtain the minimal objective functional J^*. To solve the formulated OCP, there are mainly three approaches [90]:

- **Dynamic programming:** such as Hamilton-Jacobi-Bellman equation [91].

- **Indirect methods:** based on the first order optimality conditions of variations of the OCP. They employ the philosophy of optimize then discretize, such as Pontryagin Minimum Principle, these methods are mainly used for relatively simple optimization problems.

- **Direct methods:** employ the strategy of discretize then optimize. In this work we focus on direct methods because they are well adapted for complex optimization problems such as NMPC problems.

4.3.1.5 Direct methods for OCP solution

In many systems, the OCP is solved off-line because solving it directly can be computationally intensive [91]. Therefore, in this work we will explore reformulating an OCP into a Nonlinear Programming Probelm (NLP) using direct methods [92] which allows to significantly improve the computational efficiency, tackle the resulting NLP using algorithms like sequential quadratic programming, and treat highly nonlinear OCPs with both inequality and equality constraints.

The finite dimensional Nonlinear Programming Problem (NLP) [92] is a standard problem formulation in numerical optimization having the general form:

$$\begin{aligned} \underset{\mathbf{W}}{minimize} \quad & \Phi\left(W\right), \\ s.t. \quad & G_1(W) \leq 0, \\ & G_2(W) = 0, \end{aligned} \tag{4.31}$$

where W is the optimization variable, $\Phi(W)$ is a nonlinear objective function, and $G_1(W)$, $G_2(W)$ are inequality and equality constraints respectively. To solve nonlinear OCPs using a direct method, the control trajectory is typically discretized and parameterized, while the state trajectory can be handled using either a sequential or simultaneous approach.

- **Sequential approach:** the state vector is implicitly handled alongside the control vector and initial value vector. The ordinary differential equations (ODEs) are addressed as an initial value problem using an ODE solver such as the Euler algorithm or Runge-Kutta. Consequently, simulation and optimization are sequentially conducted in each iteration of the NLP solver. The degrees of freedom in the NLP problem consist solely of the discretized control parameters. An example of the sequential method is the direct single shooting method (control parametrization).

- **Simultaneous approach:** both the state trajectory and the control trajectory are parameterized. All the parameterized variables, including both states and controls, are considered as optimization variables in the NLP problem. The discretization methods commonly used in this approach are collocation on finite elements or multiple shooting (control and state parametrization).

The different approaches for solving an OCP can be summerized in Fig. 4.4.

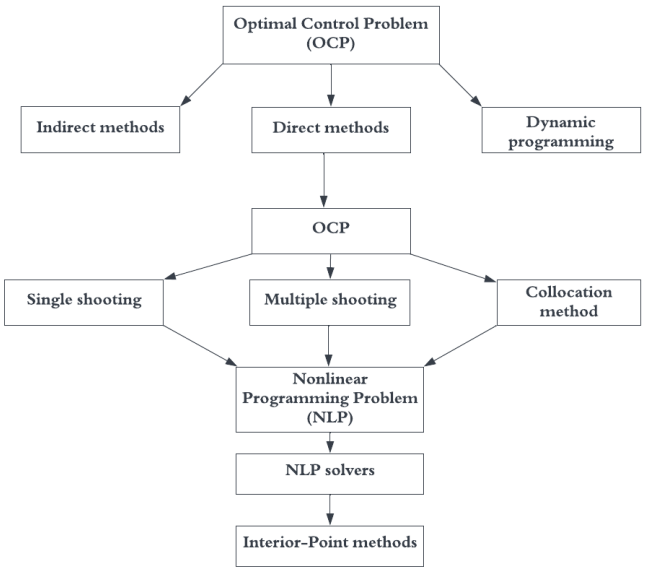

Figure 4.4: Map of Optimal Control

4.3.2 Background on Model Predictive Control

MPC is an optimal control technique in which an optimal control action is calculated to minimize a cost function for a constrained dynamical system over a finite, receding, horizon [10]. The controller can be divided into two main components: the internal dynamic plant model and the optimizer. The internal model is capable of predicting future outputs based on the current state feedback. On the other hand, the optimizer is responsible for solving an optimization problem to determine the optimal input. This controller is particularly useful for supervising MIMO control systems. At each time step, an MPC controller receives the current state of the plant. It then calculates the sequence of optimal control actions based on predictions of the system states to track a reference trajectory so that it minimizes the cost over the horizon by solving a constrained optimization problem [11]. The cost usually consists of the error between the predicted state, calculated using the internal model, and the desired trajectory. The controller then applies only the first control action to the plant, disregarding the following ones. The process is repeated in the next sampling time. The constraints can be on both the system states and the inputs. This process is illustrated in Fig. 4.5.

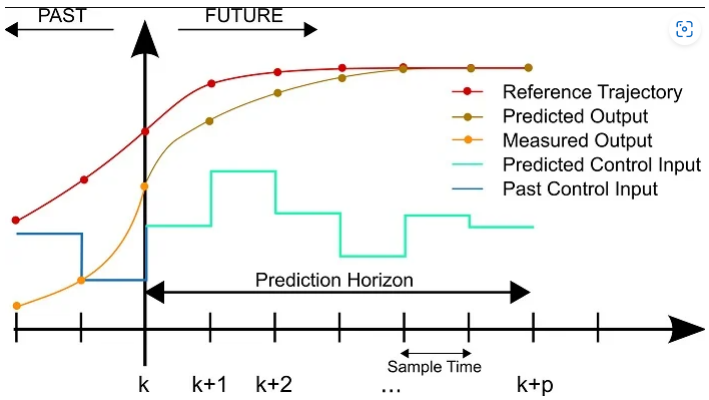

Figure 4.5: A basic working principle of MPC, from [93]

In NMPC, the OCP to be solved at each sampling time over the horizon $[t_0, t_N]$ has the general form [90] :

$$\begin{aligned} \underset{\mathbf{U(t)},\mathbf{X(t)}}{minimize} \quad & J = \int_{t_0}^{t_N} L(X(t), U(t), t)dt, \\ s.t. \quad & \dot{X}(t) = F(X(t), U(t)), && \forall t \in [t_0, t_N], \\ & X(t_0) = X_0, && \forall t \in [t_0, t_N], \\ & U(t) \in \mathcal{U}, && \forall t \in [t_0, t_N], \\ & X(t) \in \mathcal{X}, && \forall t \in [t_0, t_N], \end{aligned} \tag{4.32}$$

one major inconvenience of this approach is the online solving of the OCP at each sample time, which has enormous computational requirements.

In summary, the main goal of MPC is to find an optimal control sequence that minimizes a performance index subject to a plant model as an equality constraint and boundary conditions on the states and control inputs as inequality constraints [93]. The key principles of MPC are *prediction*, *online optimization* and *receding horizon*. In this work, we focus on NMPC, which is specifically designed for systems with nonlinear dynamics, it allows us to use the nonlinear plant model as well as nonlinear constraints.

4.3.3 NMPC problem reformulation using multiple shooting

Multiple shooting technique converts the continuous OCP into a finite dimensional NLP [90]. The main idea behind multiple shooting is to discretize the time horizon of the OCP into multiple smaller intervals, called shooting intervals, and introduce the states and the control inputs at the boundaries of each interval, where both variables are considered optimization variables. The problem is then formulated as an NLP, where the goal is to find the values of these variables that satisfy the dynamic equations and constraints over each interval [92]. Continuity conditions in each shooting are considered equalities while constraints on the state at the end of each shooting are considered as inequalities. This transformation reduces the size of the optimization problem by adding the states as optimization variables and allows a more efficient solution using numerical optimization techniques. The key steps involved in the multiple shooting approach are the following [91]:

4.3.3.1 Discretization

In multiple shooting, the prediction horizon is partitioned into N equidistant shooting intervals. We use k as the discrete time variable, the prediction horizon time grid is:

$$k < k + 1 < ... < k + N, \tag{4.33}$$

this approach offers flexibility in the choice of interval length. Different lengths can be used to capture the system dynamics more accurately which enables tailoring the discretization to the specific requirements of the problem.

The state trajectory is discretized into $N+1$ nodes and the control input is discretized into N nodes at each horizon. The control input is supposed to be piece-wise constant at each shooting interval. At each instant k, the NMPC algorithm is executed over the

horizon $[k, k+N]$ and a decision is made based on the measured or estimated state $X(k)$, a sequence of optimal controls is computed over the prediction horizon. We set the prediction horizon equal to the control horizon. The matrix $P_U(k)$ is filled with optimal control actions $U(k+j)$ over the horizon, $j \in [0, N-1]$:

$$P_U(k) = [U(k), U(k+1), ..., U(k+N-1)], \tag{4.34}$$

$$P_U(k) = \begin{bmatrix} u_1(k), u_1(k+1), ..., u_1(k+N-1) \\ u_2(k), u_2(k+1), ..., u_2(k+N-1) \\ \vdots \\ u_m(k), u_m(k+1), ..., u_m(k+N-1) \end{bmatrix}, \tag{4.35}$$

only the first control action $U(k)$ is applied during the sampling period $[k, k+1]$, then we move to the next horizon and repeat the process. The state prediction matrix corresponding to $P_U(k)$ over the horizon is $P_X(k)$:

$$P_X(k) = \left[X_U(k), X_U(k+1), ..., X_U(k+N)\right], \tag{4.36}$$

$$P_X(k) = \begin{bmatrix} x_{1,U}(k), x_{1,U}(k+1), ..., x_{1,U}(k+N) \\ x_{2,U}(k), x_{2,U}(k+1), ..., x_{2,U}(k+N) \\ \vdots \\ x_{n,U}(k), x_{n,U}(k+1), ..., x_{n,U}(k+N) \end{bmatrix}, \tag{4.37}$$

$X_U(k+j)$ is the predicted state corresponding to the optimal input $U(k+j-1)$, $j \in [1, N]$.

The NMPC problem also requires having the reference trajectories over the whole horizon. Let $X^{ref}(k)$ be the reference of the states. Therefore, the reference matrix $R_X(k)$ is:

$$R_X(k) = [X^{ref}(k), X^{ref}(k+1), ..., X^{ref}(k+N)], \tag{4.38}$$

$$R_X(k) = \begin{bmatrix} x_1^{ref}(k), x_1^{ref}(k+1), ..., x_1^{ref}(k+N) \\ x_2^{ref}(k), x_2^{ref}(k+1), ..., x_2^{ref}(k+N) \\ \vdots \\ x_n^{ref}(k), x_n^{ref}(k+1), ..., x_n^{ref}(k+N) \end{bmatrix}, \tag{4.39}$$

4.3.3.2 Objective function

The objective function captures the system's performance and is typically a sum of stage costs over all intervals. Let the running costs at each instant $k+j$, $j \in [0, N]$ be:

$$L(X_U(k+j), U(k+j)) = \left\|X_U(k+j) - X^{ref}(k+j)\right\|_Q^2 + \|U(k+j))\|_R^2, \tag{4.40}$$

$$\begin{aligned} L(X_U(k+j), U(k+j)) = {} & [X_U(k+j) - X^{ref}(k+j)]^T Q [X_U(k+j) - X^{ref}(k+j)] \\ & + U^T(k+j) R U(k+j), \end{aligned} \tag{4.41}$$

the running cost characterizes the control objectives, it penalizes the tracking error between the predicted states X_U and the desired trajectory X^{ref}, and it includes the minimal control effort objective.

The cost function is the evaluation of the running costs along the whole horizon:

$$J_N = \sum_{j=0}^{N-1} L\left(X_U(k+j), U(k+j)\right), \tag{4.42}$$

the minimal value of the cost function is:

$$V_N(X) = \min_{\mathbf{U}} J_N\left(X_U, U\right). \tag{4.43}$$

4.3.3.3 Optimization variable

The decision variable W in the context of multiple shooting is given by:

$$W = [U^T(k), U^T(k+1), ..., U^T(k+N-1), X_U^T(k), X_U^T(k+1), ..., X_U^T(k+N)]^T, \tag{4.44}$$

for simplicity, we write:

$$W = [U_0^T, ..., U_{N-1}^T, X_{U,0}^T, X_{U,1}^T, ..., X_{U,N}^T]^T, \tag{4.45}$$

it contains both the controls and the states as optimization variables over the prediction horizon, its dimension is $(m * N + n * (N+1))$. Using W as the optimization variable in the cost function is a major advantage of multiple shooting, it allows the 'lifting' which means reformulating a function with more variables which makes it less nonlinear. Also multiple shooting allows the initialization with a known guess for the state trajectory.

4.3.3.4 Dynamics constraints

The differential equations describing the system dynamics are discretized and imposed as constraints between adjacent intervals. These constraints naturally ensure the continuity of states and control inputs between intervals, ensuring that the system's dynamics are satisfied over the entire time horizon. Consider the nonlinear plant model developed in section 3.2:

$$\dot{X}(t) = F(X(t), U(t)), \tag{4.46}$$

using Euler method, we write the system model in the discrete time domain. Other more complex algorithms such as Runge-Kutta can be employed. However, solving the NMPC problem can become more computationally expensive.

$$X(k+1) = X(k) + T * F(X(k), U(k)) = f(X(k), U(k)), \tag{4.47}$$

T is the sampling time. In multiple shooting, this model is added as a constraint between the actual state and the predicted state (based on the plant model). This equality constraint is considered at each shooting step as follows:

$$f(X_U(k), U(k)) - X_U(k+1) = 0, \tag{4.48}$$

this equality constraint is written in term of the optimization variables contained in W as follows:

$$G_2(W) = \begin{bmatrix} X(k) - X_{U,0} \\ f(X_{U,0}, U_0) - X_{U,1} \\ \vdots \\ f(X_{U,N-1}, U_{N-1}) - X_{U,N} \end{bmatrix} = 0, \tag{4.49}$$

where $X(k) = f(X(k-1), U(k-1))$ is the real state coming from the system at each iteration after applying the optimal control $U(k-1)$.

This approach simplifies the constraint formulation and eliminates the need for complex differential equation solvers. Moreover, the introduction of state variables at the boundaries reduces the size of the optimization problem which leads to significant computational efficiency gains, allowing for faster solution times compared to solving the original OCP directly.

4.3.3.5 Inequality constraints and bounds

Any additional constraints, such as state constraints or control constraints, are also included in the NLP formulation.

$$G_1(W) = \begin{bmatrix} g_1(X_{U,0}, U_0) \\ \vdots \\ g_1(X_{U,N-1}, U_{N-1}) \\ g_1(X_{U,N}) \end{bmatrix} \leq 0, \tag{4.50}$$

bounds on control inputs and states are also considered:

$$\begin{aligned} &U(k+j) \in \mathcal{U}, \quad \forall j \in [0, N-1], \\ &X_U(k+j) \in \mathcal{X}, \quad \forall j \in [0, N], \end{aligned} \tag{4.51}$$

where:

$$\begin{aligned} \mathcal{U} &= \{U \in \mathbb{R}^m \mid U^{min} \leq U \leq U^{max}\}, \\ \mathcal{X} &= \{X_U \in \mathbb{R}^n \mid X^{min} \leq X_U \leq X^{max}\}, \end{aligned} \tag{4.52}$$

4.3.3.6 The resulting Nonlinear Programming Problem

Multiple shooting allowed to reformulate the NMPC OCP to consider the variables at interval boundaries:

$$\begin{aligned} \underset{\mathbf{W}}{minimize} \quad & J_N = \sum_{j=0}^{N-1} L\left(X_U(k+j), U(k+j)\right), \\ s.t. \quad & X_U(k+1) = f\left(X_U(k), U(k)\right), \\ & X_U(0) = X_0, \\ & U(k+j) \in \mathcal{U}, \quad \forall j \in [0, N-1], \\ & X_U(k+j) \in \mathcal{X}, \quad \forall j \in [0, N], \end{aligned} \tag{4.53}$$

the resulting Nonlinear Programming Problem can be written in terms of the optimization variable $W = [U_0^T, ..., U_{N-1}^T, X_{U,0}^T, X_{U,1}^T, ..., X_{U,N}^T]^T$:

$$\begin{aligned} \underset{U_0,...,U_{N-1},X_{U,0},X_{U,1},...,X_{U,N}}{minimize} \quad & J_N = \sum_{j=0}^{N-1} L\left(X_{U,j}, U_j\right), & \\ s.t. \quad & g_1(X_{U,j}, U_j) \leq 0, & j \in [0, N], \\ & f(X_{U,j-1}, U_{j-1}) - X_{U,j} = 0, & j \in [0, N], \end{aligned} \tag{4.54}$$

which has the general form:

$$\begin{aligned} \underset{\mathbf{W}}{minimize} \quad & \Phi(W), \\ s.t. \quad & G_1(W) \leq 0, \\ & G_2(W) = 0, \end{aligned} \tag{4.55}$$

overall, the main difference between an OCP and an NLP lies in their formulation and structure. This transform through multiple shooting reduces the problem size, improves computational efficiency and simplifies constraint handling, making multiple shooting a popular technique for solving nonlinear MPC problems, particularly when dealing with complex systems and long time horizons. By transforming the OCP into an NLP through multiple shooting, the resulting problem can be solved using a wide range of existing nonlinear programming solvers. These solvers are well-developed and optimized for NLP problems, offering various optimization algorithms and solution techniques.

In this work, the NLP is solved in MATLAB using CasADi, an open-source software containing symbolic tools specifically designed for solving nonlinear optimization problems. CasADi offers a high-level optimization framework that makes the process of formulating and solving the problem very efficient. It is particularly useful for complex nonlinear dynamic systems which makes it adequate for the quadrotor trajectory tracking problem. CasADi is proven to run much faster than other optimization packages [94], its computational performance makes it very suitable for our application. By utilizing CasADi, the user can express the objective function, constraints, and dynamics equations of the NLP in a concise and intuitive manner and then invoke the appropriate solver to find the optimal solution. The NLP solver used is the IPOPT (Interior Point OPTimizer) [91] solver which is a popular NLP solver that is integrated into CasADi. IPOPT uses an interior-point method, which is a class of optimization algorithms that iteratively approach the optimal solution by moving along the interior of the feasible region. It is particularly effective for handling convex and non-convex OCPs.

4.3.4 Application of NMPC on a single quadrotor

We use multiple shooting NMPC for trajectory tracking of a single quadrotor. The quadrotor has 12 states and 4 inputs. Therefore, the decision variable W is of dimension $(4N + 12(N + 1))$, i.e., $W \in \mathbb{R}^{(4N+12(N+1))}$, and is given as follows:

$$W = [U_0^T, ..., U_{N-1}^T, X_{U,0}^T, X_{U,1}^T, ..., X_{U,N}^T]^T. \tag{4.56}$$

To initialize the decision variables, it is advantageous to use the previous predictions at instant k as an initialization for the next horizon that starts at instant $k+1$ for the states

and inputs. Therefore, we trim the first prediction at instant k that we already used, and we double the last prediction at instant $k+N$. Using the previous prediction matrix:

$$P_X(k) = \left[X_U(k), X_U(k+1), ..., X_U(k+N)\right], \tag{4.57}$$

we trim the first column and double the last column:

$$P_X^0(k+1) = \left[X_U(k+1), ..., X_U(k+N), X_U(k+N)\right], \tag{4.58}$$

we do the same for initializing the controls:

$$P_U^0(k+1) = [U(k+1), ..., U(k+N-1), U(k+N-1)]. \tag{4.59}$$

The nonlinear quadrotor model $f(X,U)$ is provided to NMPC to predict the system states. It is used as an equality constraint at each shooting step.

$$G_2(W) = \begin{bmatrix} X(k) - X_{U,0} \\ f(X_{U,0}, U_0) - X_{U,1} \\ \vdots \\ f(X_{U,N-1}, U_{N-1}) - X_{U,N} \end{bmatrix} = 0. \tag{4.60}$$

The optimal solution must also satisfy boundary conditions. For the quadrotor, we consider lower and upper bounds on states and control inputs:

$$U^{min} \leq U \leq U^{max}, \tag{4.61}$$

$$X^{min} \leq X_U \leq X^{max}. \tag{4.62}$$

4.4 Design of estimation-based control techniques

In this section, we discuss the design of estimation-based control techniques for quadrotor systems. We first discuss the motivations and emphasize the necessity and benefits of using estimation techniques for precise control. Then, we consider measurements and sensor choice for our application. Finally, we examine the Extended Kalman Filter, explaining its basic principles and how the algorithm estimates and filters the system states based on noisy sensor measurements.

4.4.1 Motivation and estimation approach

The accuracy of the feedback signal plays a crucial role in the performance of a feedback-based controller. To accurately represent the system's state, various sensors and filters are employed. While commercial quadrotors are equipped with onboard sensors like Global Positioning System (GPS), Inertial Measurement Unit (IMU), radio detection and ranging (radar), sound navigation ranging (Sonar), light detection and ranging (Lidar), cameras, Attitude and Heading Reference System (AHRS), Inertial Navigation System (INS), etc., these sensors alone often fail to provide a reliable feedback signal. As a result, this limitation is addressed by developing an estimation approach alongside the onboard sensors. This approach aims to overcome the drawbacks associated with the onboard sensors and enhance the overall accuracy of the feedback signal.

In previous research, estimation methods have been employed to improve the performance and control of quadrotor systems. For instance, a robustified nonlinear dynamic inversion control scheme based on an uncertainty and disturbance estimator (UDE) was proposed to address nonlinearities, input coupling, uncertainties, and external disturbances [95]. Sensor fusion algorithms, such as EKF, have also been widely adopted for integrating multiple measurements in complex scenarios for quadrotors. A methodology was presented to estimate quadrotor orientation using a single low-cost IMU sensor through the use of two EKFs and a Direction Cosine Matrix algorithm [96]. Another study derived an EKF for various drone models in different dimensions, aiming to infer the state of the quadrotor from sensor values and control inputs [97]. These examples emphasize the importance of estimation approaches, including the EKF, in inferring the state of quadrotor systems.

4.4.2 Choice of measurements and sensors

For our specific application, we measured both the linear and angular positions of the quadrotor. This choice was influenced by a comprehensive evaluation of the quadrotor system's observability, as demonstrated in the study conducted in [98]. Additionally, through our own extensive trials, we have concluded that EKF estimation using these measurements yields accurate results. Although it's important to consider the limitations in terms of accuracy and update rate associated with the measurements.

To mesure the quadrotor's global position, a GPS receiver is used. For the quadrotor's angular position, we opted for an AHRS which is widely used for attitude estimation due to its simplicity, accuracy, and real-time performance. By combining data from gyroscopes, accelerometers, and magnetometers, the AHRS estimates the quadrotor's attitude angles. However, it's worth noting that AHRS alone may encounter challenges in providing highly accurate orientation estimates over extended periods due to sensor drift and noise. In order to fuse these measurements and address potential inaccuracies of individual sensors, we implement an EKF which is known for its sensor fusion capabilities, enabling us to estimate the remaining six states of the quadrotor (linear and angular velocities). Additionally, the EKF can provide an improved estimation of the measurements, which helps mitigate the impact of sensor inaccuracies. This approach of combining a GPS, AHRS, and an EKF aims to obtain accurate and reliable estimates of the quadrotor's states, while addressing the limitations and uncertainties associated with individual sensors.

4.4.3 Background on the Extended Kalman Filter

EKF is a suboptimal nonlinear version of the Kalman Filter (KF). KF is a recursive algorithm that estimates the state of a linear dynamical system with measurements and process noise. However, when dealing with nonlinear systems, such as those with nonlinear state transition and observation models, the standard KF cannot be directly applied [83].

On the other hand, EKF utilizes the process of linearization by approximating the nonlinear state transition and measurement models based on the current state estimate. This linearization is achieved by computing the Jacobians, which capture the local linear behavior of the models around the estimated state. By linearizing the models, EKF is able to update the covariance estimate using the same equations as the standard KF. EKF

operates in two main stages: the prediction stage, where the system's state is estimated based on the previous state estimate and the transition model, and the update stage, where the estimated state is corrected based on the measurements.

4.4.3.1 Extended Kalman Filter equations

The nonlinear system in continuous time is represented as follows:

$$\dot{X}(t) = F(X, U) + \xi(t), \tag{4.63}$$

$$Z(t) = h^{mes}(X) + \upsilon(t), \tag{4.64}$$

where $F(X, U)$ is the nonlinear state model, $h^{mes}(X)$ is the measurements model, $\xi(t)$, $\upsilon(t)$ are white Gaussian process and measurement noises respectively. In discrete time domain, the system equations become:

$$\begin{aligned} X(k) &= X(k-1) + F(X(k-1), U(k-1)) * T + \xi(k-1), \\ &= f(X(k-1), U(k-1)) + \xi(k-1), \end{aligned} \tag{4.65}$$

$$Z(k) = h^{mes}(X(k), U(k)) + \upsilon(k), \tag{4.66}$$

T is the sampling time. We linearize the states and measurement equations by taking the partial derivatives, the Jacobians are defined as:

$$A(k) = \left.\frac{\partial F}{\partial X}\right|_{\hat{X}(k),U(k)}, \tag{4.67}$$

$$H(k) = \left.\frac{\partial h^{mes}}{\partial X}\right|_{\hat{X}(k),U(k)}, \tag{4.68}$$

the linearization point $\hat{X}(k)$ is the latest state estimation at each time sample. The calculations are performed using the following two steps, the notations $(^-)$ and $(^+)$ represent the estimated value before and after the measurement update:

Step 1: Prediction

In this step, given the estimation at the $k-1$ step, the state of the system at time step k is predicted based on the system's model.

$$\hat{X}^-(k) = f(\hat{X}^+(k-1), U(k-1)), \tag{4.69}$$

we predict also the covariance of the estimation error at time k:

$$P^-(k) = A(k-1)P^+(k-1)A^T(k-1) + Q_\xi, \tag{4.70}$$

Q_ξ is the covariance of the process noise $\xi(k)$.

Step 2: Update

We compute the Kalman gain $K(k)$ at time k:

$$K(k) = P^-(k)H^T(k)\Big[H(k)P^-(k)H^T(k) + R_\upsilon\Big]^{-1}, \tag{4.71}$$

this gain is designed to minimize the covariance of the estimation error $\tilde{X} = X - \hat{X}$. Then, using the measurements, we update the prediction:

$$\hat{X}^{+}(k) = \hat{X}^{-}(k) + K(k)\Big[Z(k) - h^{mes}(\hat{X}^{-}(k))\Big], \tag{4.72}$$

we finally update the covariance estimate:

$$P^{+}(k) = \Big[I - K(k)H(k)\Big]P^{-}(k), \tag{4.73}$$

R_υ is the covariance of the measurement noise $\upsilon(k)$. In summary, the EKF algorithm consists of four main steps, discretization, linearization, prediction and update.

4.4.3.2 Covariance matrices tuning

The choice of the Q_ξ and R_υ matrices is very important, they are used to model the process noise and measurement noise, respectively. In general, a high value for the Q_ξ matrix indicates high process noise, which means that the system dynamics are highly uncertain or variable. On the other hand, a low value for the Q_ξ matrix indicates low process noise, which means that the system dynamics are less uncertain. Similarly, a high value for the R_υ matrix indicates high measurement noise, which means that the measurements are highly uncertain. A low value for the R_υ matrix indicates low measurement noise, which means that the measurements are less uncertain.

The filter determines the relative weights given to the measurements and the model predictions by calculating the Kalman gain, which is based on the covariance matrices Q_ξ and R_υ. The Kalman gain determines the amount of weight given to the measurement and the model prediction in the estimate. If the measurement noise is high and the model is accurate, the Kalman gain will give more weight to the model prediction. If the measurement noise is small or the model is less accurate, the Kalman gain will give more weight to the measurement. It is important to choose appropriate values for the Q_ξ and R_υ matrices to ensure that the EKF is able to accurately estimate the state of the system. The values of Q_ξ and R_υ are usually determined through experimentation or by using statistical analysis of the system and sensor noise characteristics.

4.4.4 Application of EKF on a single quadrotor

The nonlinear quadrotor model with process and measurement noise is given as follows:

$$\dot{X}(t) = F(X, U) + \xi(t), \tag{4.74}$$

$$Z(t) = h^{mes}(X) + \upsilon(t), \tag{4.75}$$

$$h^{mes}(X) = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T, \tag{4.76}$$

the measurements vector contains the linear and angular positions of the quadrotor. EKF needs both the state and measurement models to be linarized. For the state-transition model, the Jacobian $A(k)$ is a 12-by-12 sparse matrix. Its nonzero elements are:

$$
\begin{aligned}
&A_{1,4} = 1,\\
&A_{2,5} = 1,\\
&A_{3,6} = 1,\\
&A_{7,8} = -\frac{I_z - I_y}{I_x} r,\\
&A_{7,9} = -\frac{I_z - I_y}{I_x} q,\\
&A_{8,7} = -\frac{I_x - I_z}{I_y} r,\\
&A_{8,9} = -\frac{I_x - I_z}{I_y} p,\\
&A_{9,7} = -\frac{I_y - I_x}{I_z} q,\\
&A_{9,8} = -\frac{I_y - I_x}{I_z} p,\\
&A_{10,7} = 1,\\
&A_{10,8} = sin(\phi)tan(\theta),\\
&A_{10,9} = cos(\phi)tan(\theta),\\
&A_{10,10} = cos(\phi)tan(\theta)q,\\
&A_{10,11} = -sin(\phi)tan(\theta)r,\\
&A_{11,8} = cos(\phi),\\
&A_{11,9} = -sin(\phi),\\
&A_{11,10} = -sin(\phi)q - cos(\phi)r,\\
&A_{12,8} = \frac{sin(\phi)}{cos(\theta)},\\
&A_{12,9} = \frac{cos(\phi)}{cos(\theta)},\\
&A_{12,10} = \frac{cos(\phi)}{cos(\theta)} q - \frac{sin(\phi)}{cos(\theta)} r,\\
&A_{12,11} = \frac{sin(\phi)sin(\theta)q + cos(\phi)sin(\theta)}{cos^2(\theta)},
\end{aligned}
\tag{4.77}
$$

the Jacobian of the measurements model for the quadrotor is a 6-by-12 matrix, it is given by:

$$
H(k) = \begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times6} \\ 0_{3\times6} & I_{3\times3} & 0_{3\times3} \end{bmatrix}, \tag{4.78}
$$

it is worth noting that the computational complexity of EKF is high because it requires evaluating the Jacobians at the current estimate in real time.

4.5 Simulation results

In this section, the different control and estimation strategies developed previously will be validated through simulations of the trajectory tracking problem of the single quadrotor system. The reference trajectory is a 3D continuous helical trajectory given by $(x^{ref}(t), y^{ref}(t), z^{ref}(t)) = (2\sin 0.1t, 2-2\cos 0.1t, 0.1t)$ and the desired yaw angle $\psi^{ref}(t)$ is a square signal. The wind forces are neglected. The simulation parameters of the quadrotor model are listed in table 4.2, they are chosen to represent a real quadrotor [99].

Parameters	Value	Unit
g	9.81	$m.s^{-2}$
m	0.2	kg
I_x	0.1	$kg.m^2$
I_y	0.1	$kg.m^2$
I_z	0.08	$kg.m^2$

Table 4.2: Quadrotor's model parameters

To ensure an accurate simulation of the quadcopter's closed-loop response, it is important to take into account actuator saturation. This means that the produced thrust and torques should be limited by the maximum achievable motor speeds.

4.5.1 PD controller results

After executing the genetic algorithm with 15 generations having 15 individuals each, we get the optimal PD parameters in table 4.3 with the minimal cost function value $min\, J_G(\Theta) = 1.2821$.

Axis	k_P	k_D	Saturation
x	0.4062	0.1617	$\lvert\theta^{ref}\rvert \leq 1.05\ [rad]$
θ	8.8283	1.7920	$\lvert u_3\rvert \leq 2\ [N.m]$
y	0.4062	0.1617	$\lvert\phi^{ref}\rvert \leq 1.05\ [rad]$
ϕ	8.8283	1.7920	$\lvert u_2\rvert \leq 2\ [N.m]$
z	11.8091	2.5949	$\lvert u_1\rvert \leq 2.1\ [N]$
ψ	6.9740	1.7235	$\lvert u_4\rvert \leq 2\ [N.m]$

Table 4.3: Optimal PD simulation parameters

Trajectory tracking results for the initial condition $X_0 = [0.3\ 0.3\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

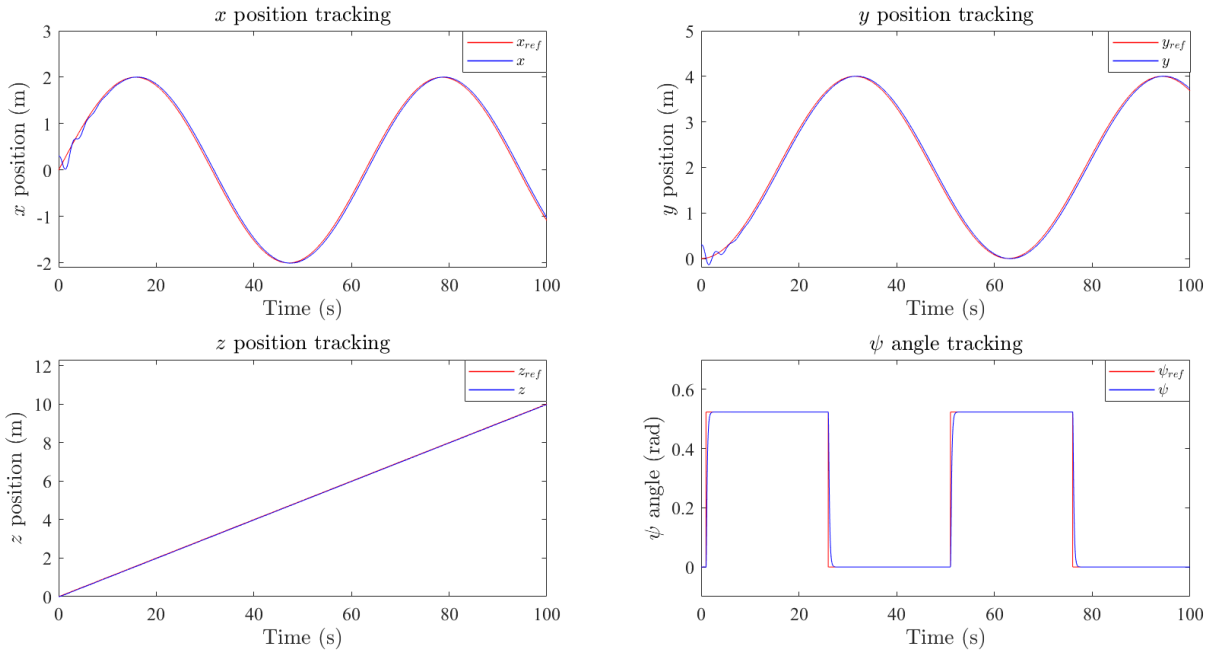

Figure 4.6: PD trajectory tracking

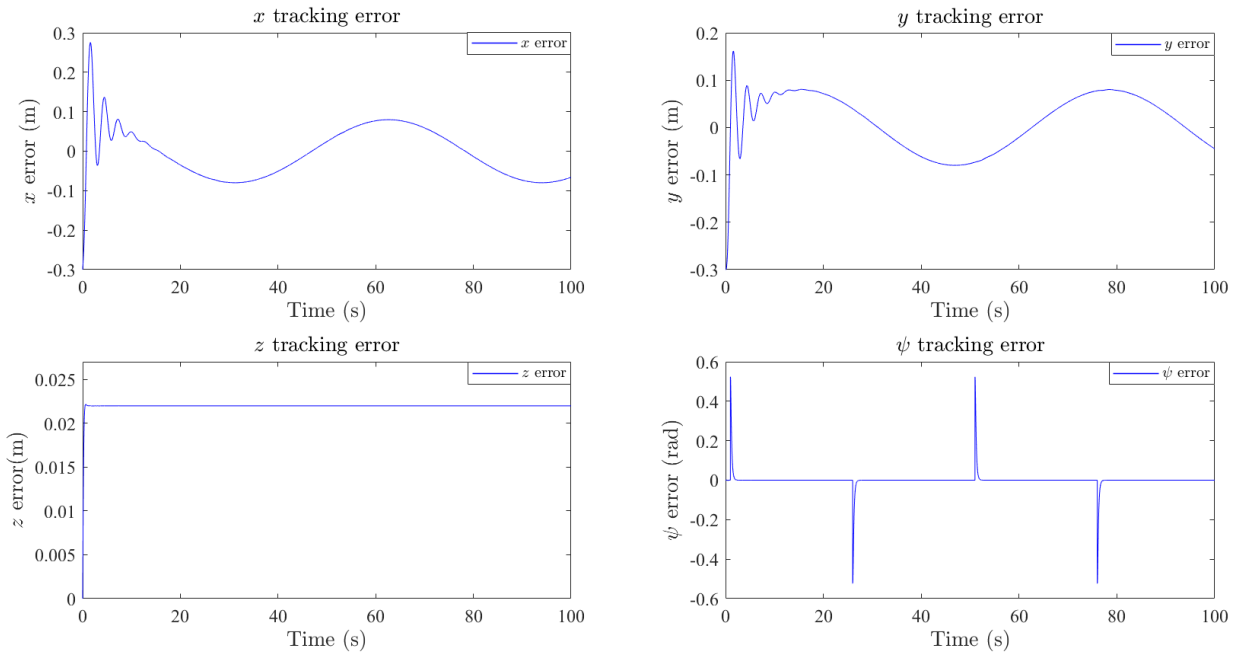

Figure 4.7: PD tracking errors

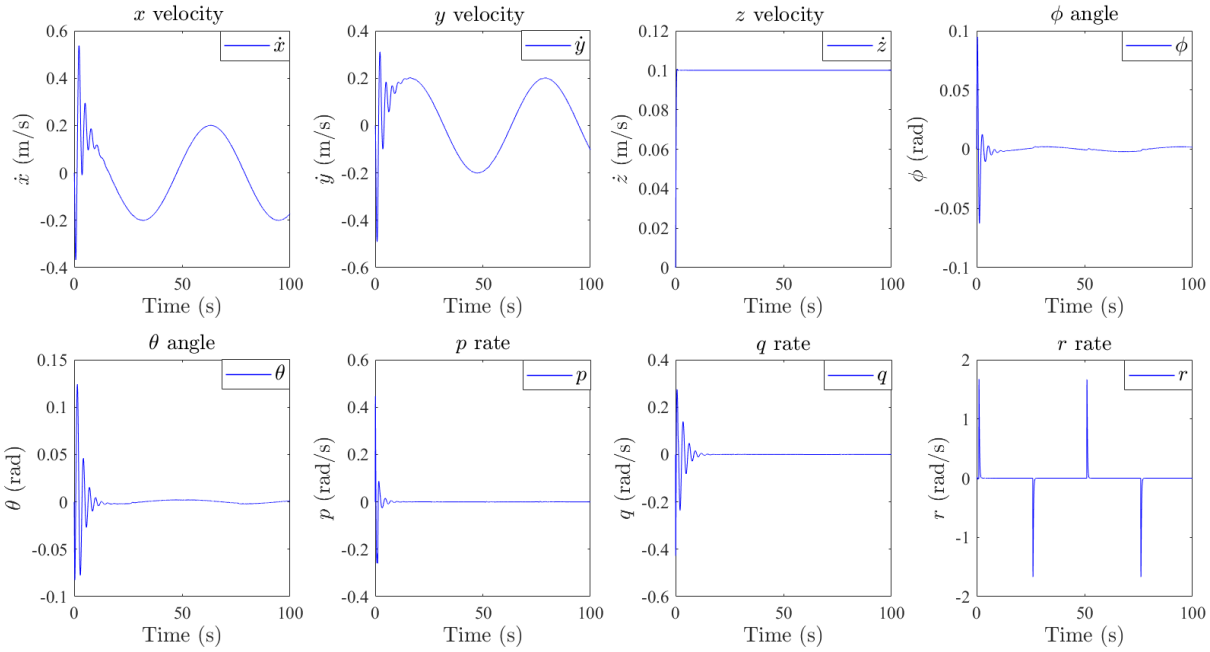

Figure 4.8: Quadrotor's states with PD controller

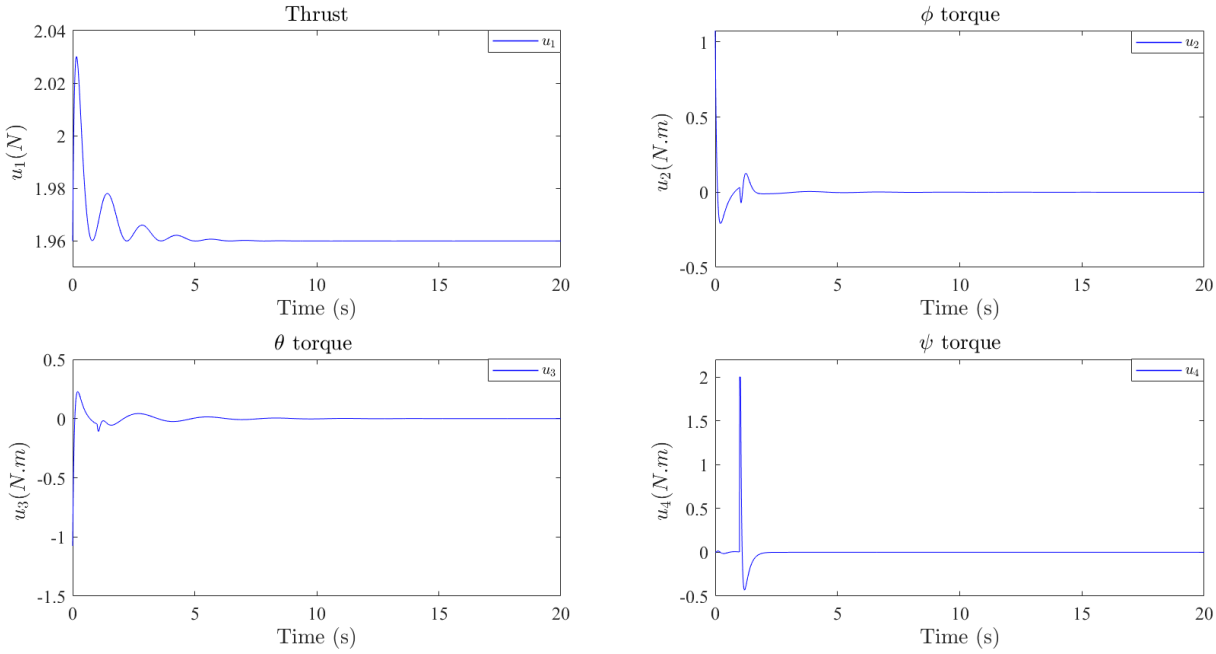

Figure 4.9: PD input efforts

4.5.2 EKF-based PD controller results

In this subsection, we use EKF in closed-loop with the PD controller, meaning the controller uses the estimated state $\hat{X}$ for the feedback instead of X. We used additive white Gaussian noise for process and measurement noise. The following weighting matrices to characterize the noise and the initial estimation error covariance are selected through an

iterative process of trial and error:

$$
\begin{aligned}
&Q_\xi = diag(0.05, 0.02, 0.02, 0.02, 5, 5, 0.02, 0.02, 0.0002, 0.02, 0.02, 0.0002),\\
&R_\nu = diag(3.5, 2, 2, 0.000001, 0.000001, 0.00005),\\
&P_0 = 0.001 * diag(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1).
\end{aligned}
\tag{4.79}
$$

Trajectory tracking results for the initial condition $\hat{X}_0 = [0.3\ 0.3\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

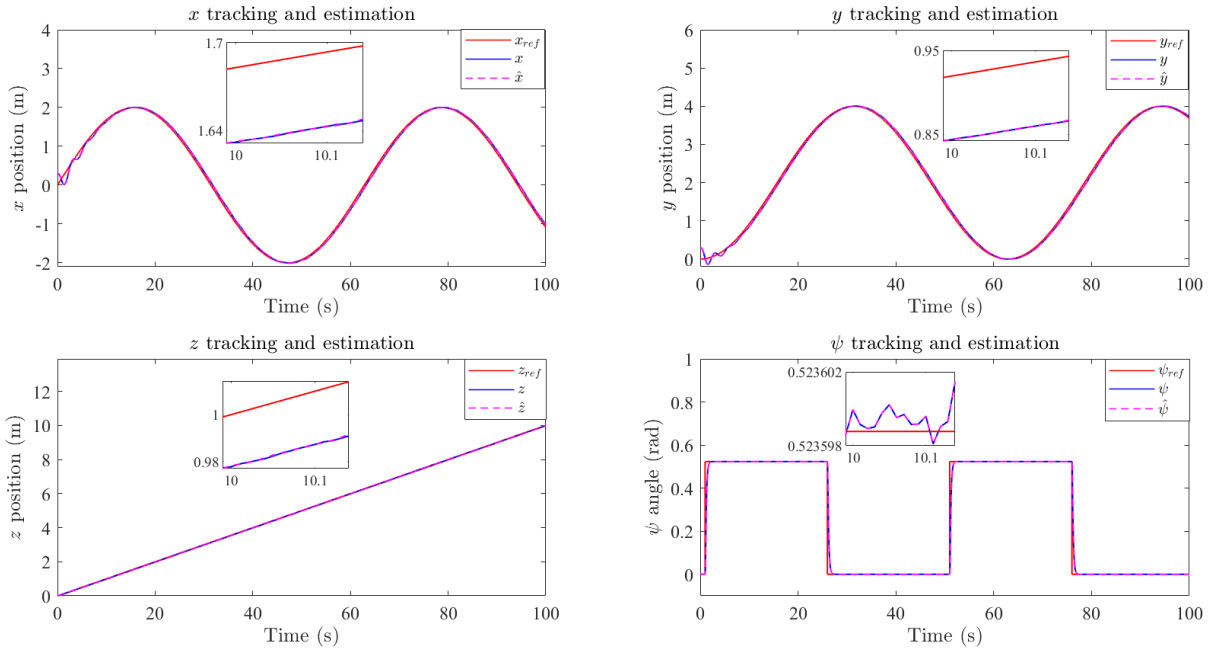

Figure 4.10: EKF-based PD trajectory tracking

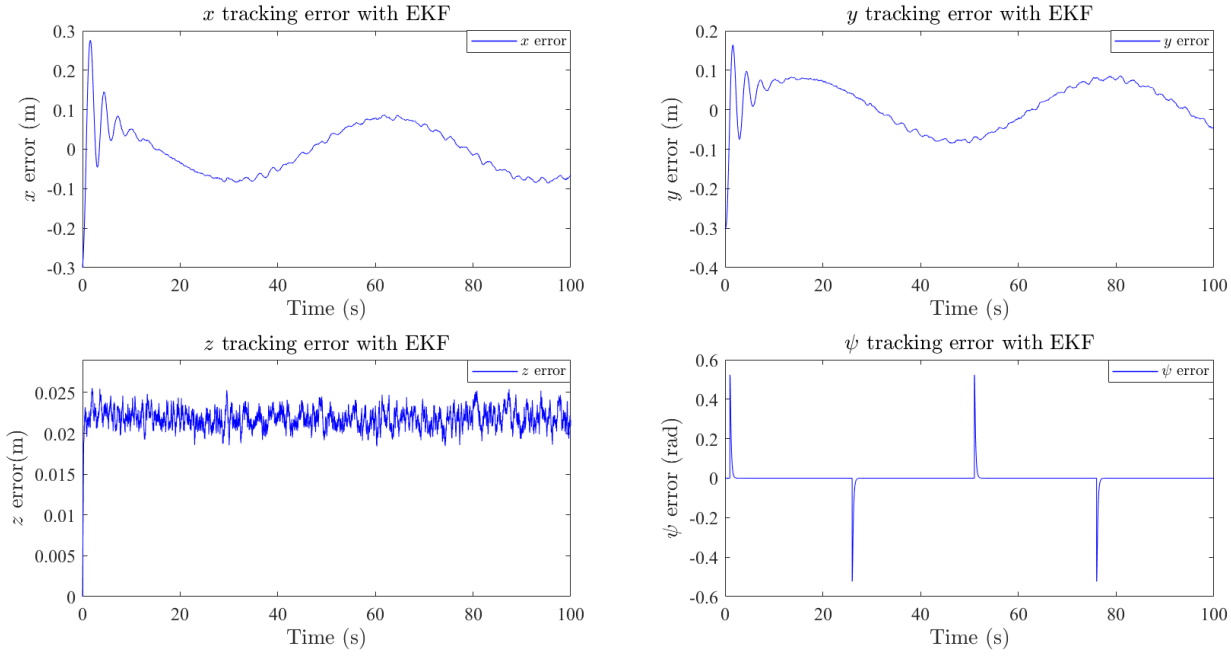

Figure 4.11: EKF-based PD tracking errors

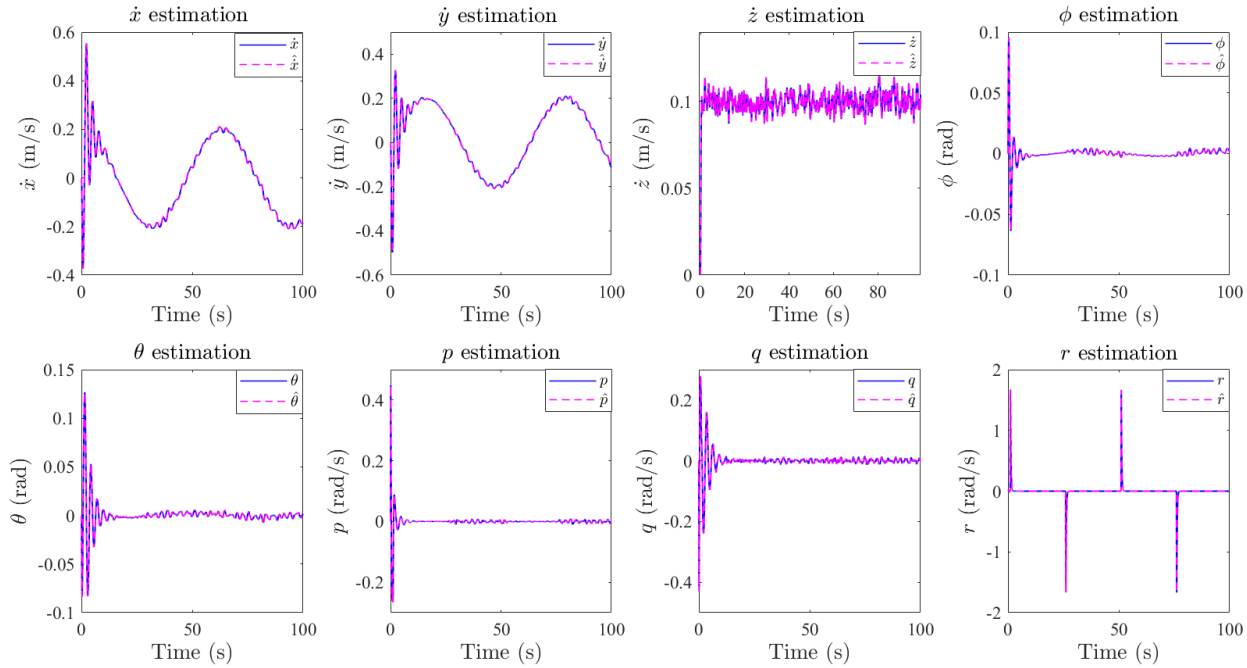

Figure 4.12: Quadrotor's states with EKF-based PD controller

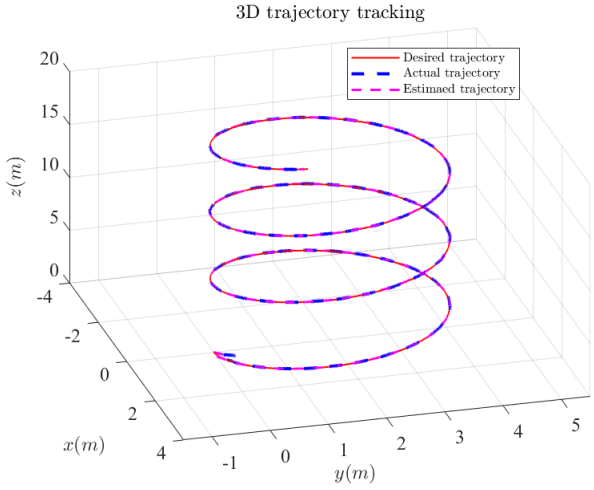

Figure 4.13: EKF-based PD trajectory tracking in 3D space

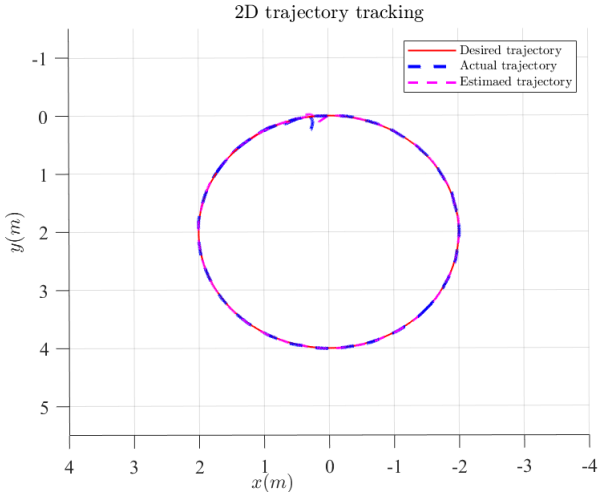

Figure 4.14: EKF-based PD trajectory tracking in the $x-y$ plane

4.5.3 NMPC controller results

The NMPC algorithm is run over a prediction horizon of $5s$, the weighting matrices in the cost function are shown in table 4.4. The input and state boundaries are the same as for the PD controller, shown in table 4.3.

NMPC parameters	Values
Sampling time T	0.05 [s]
Number of intervals N	100
Prediction horizon length	5 [s]
Weighting matrix Q	$diag(1,1,1,1,1,1,1,1,1,1,1,0)$
Weighting matrix R	$diag(1,1,1,1)$

Table 4.4: NMPC simulation parameters

Trajectory tracking results for the initial condition $X_0 = [0.3\ 0.3\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

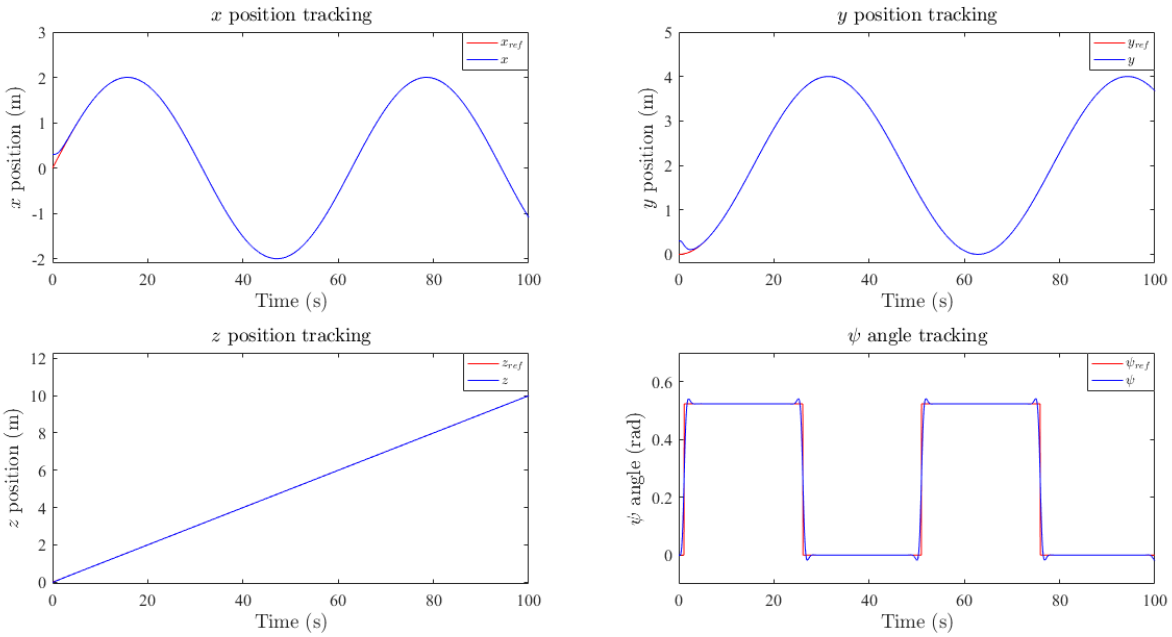

Figure 4.15: NMPC trajectory tracking

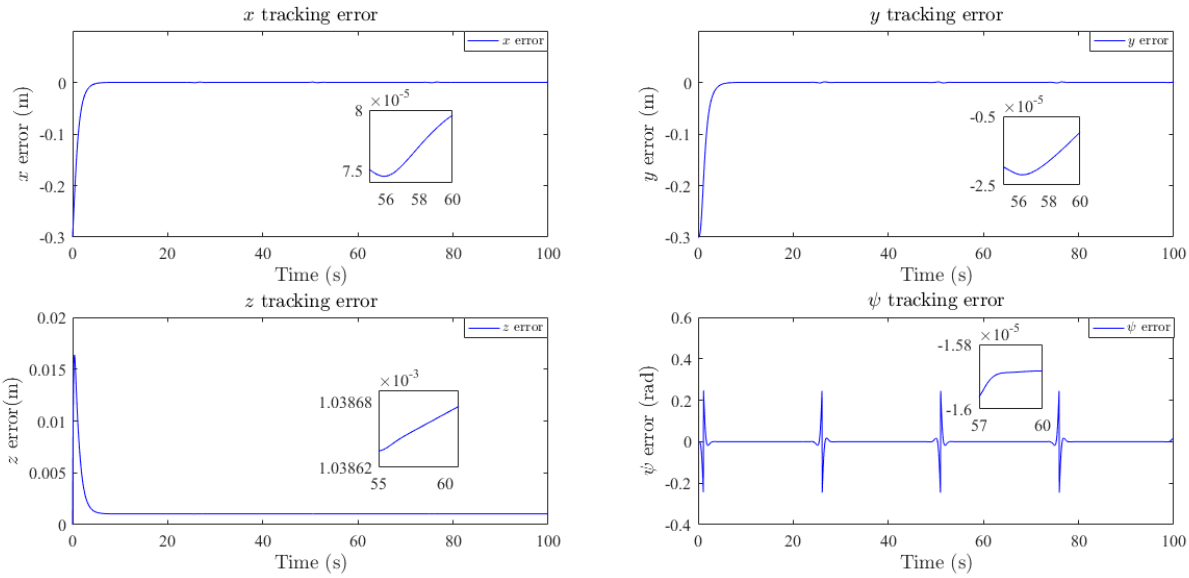

Figure 4.16: NMPC tracking errors

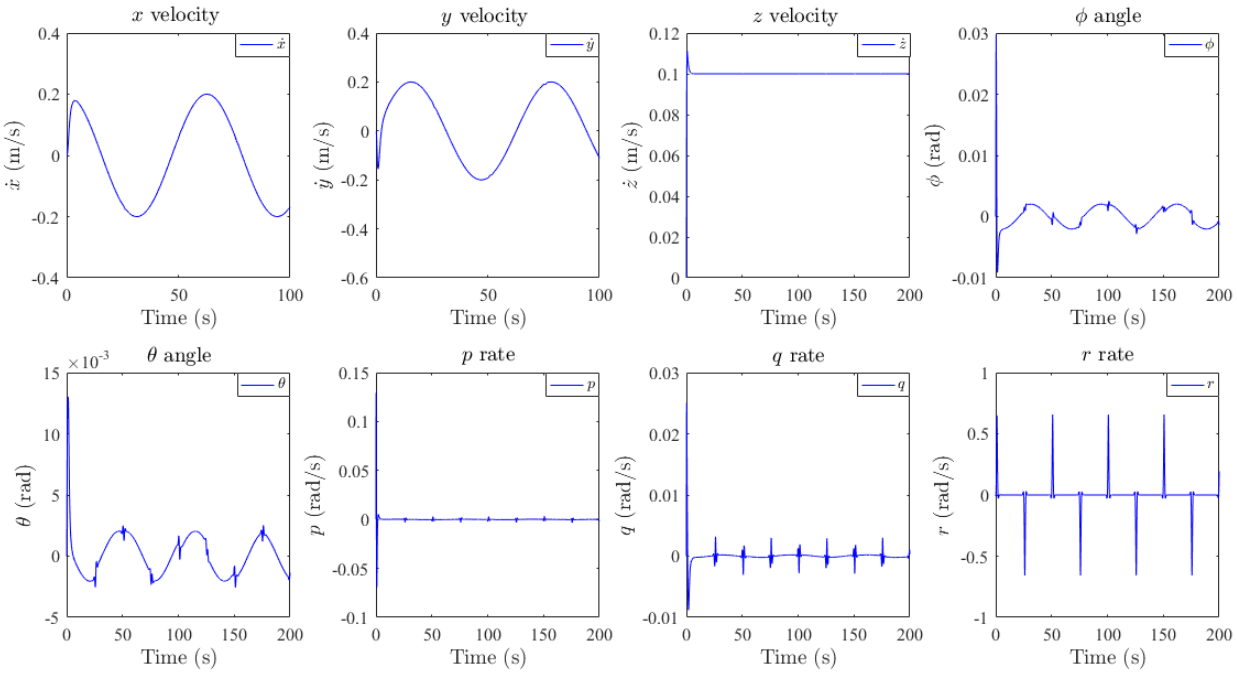

Figure 4.17: Quadrotor's states with NMPC controller

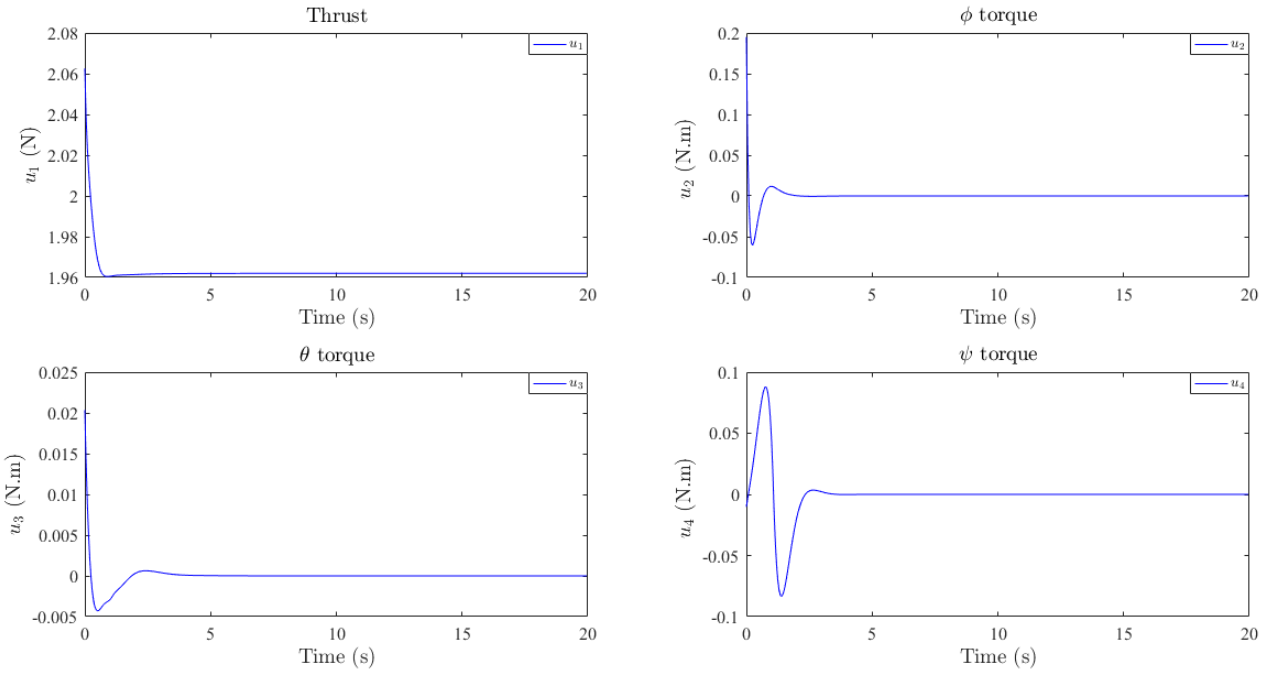

Figure 4.18: NMPC input efforts

4.5.4 EKF-based NMPC controller results

In this subsection, we use EKF in closed-loop with the NMPC controller. The matrices Q_ξ, R_ν and P_0 are the same as for the EKF-based PD.

Trajectory tracking results for the initial condition $\hat{X}_0 = [0.3\ 0.3\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$

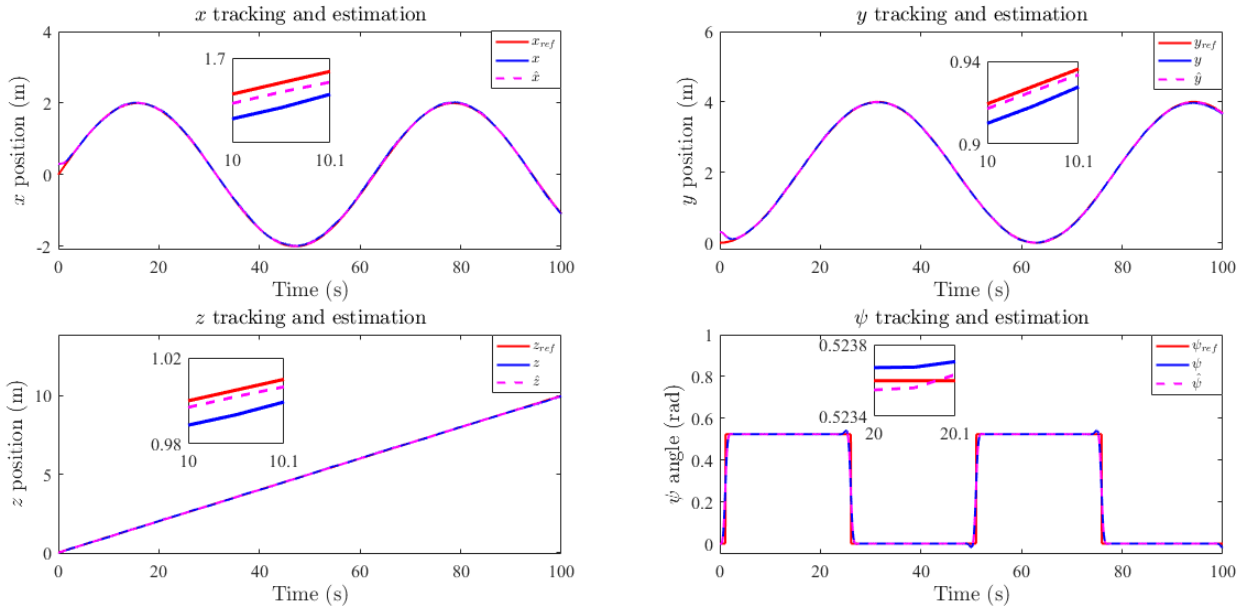

Figure 4.19: EKF-based NMPC trajectory tracking

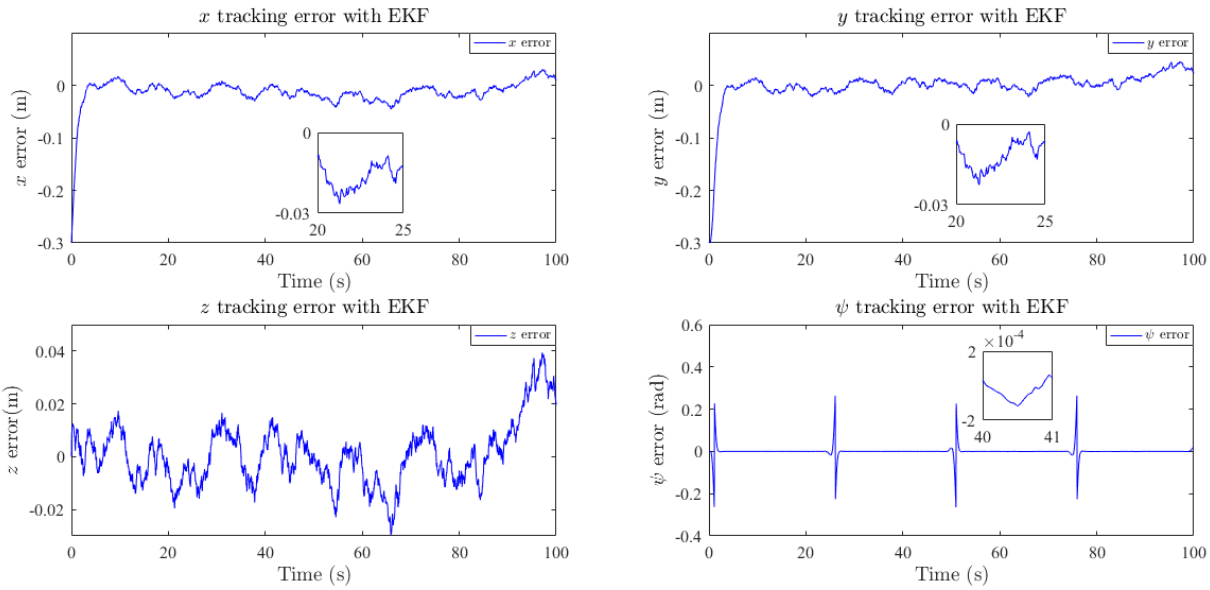

Figure 4.20: EKF-based NMPC tracking errors

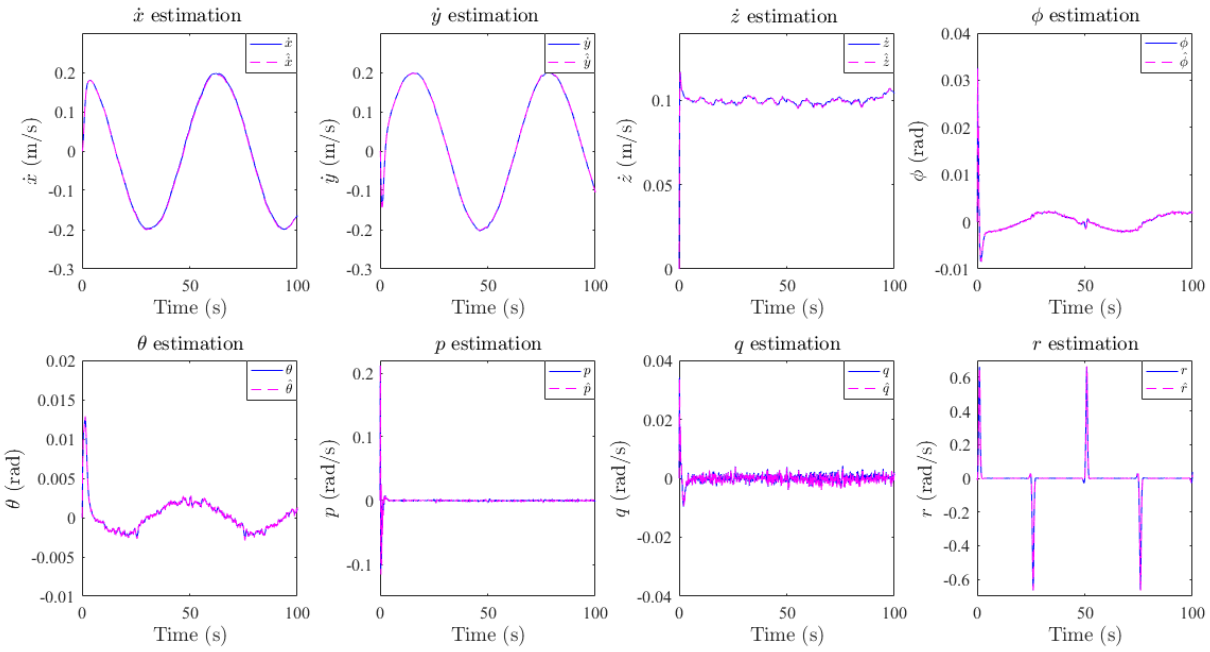

Figure 4.21: Quadrotor's states with EKF-based NMPC controller

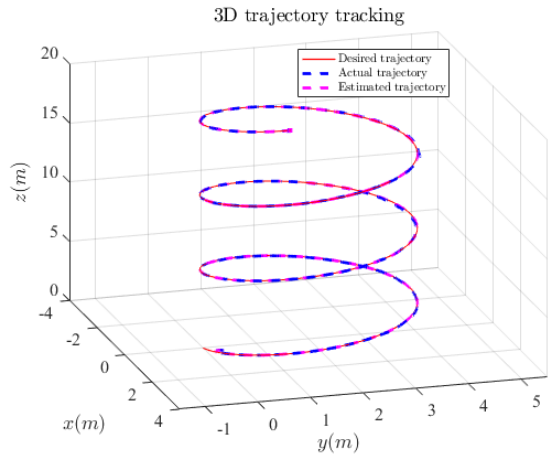

Figure 4.22: EKF-based NMPC trajectory tracking in 3D space

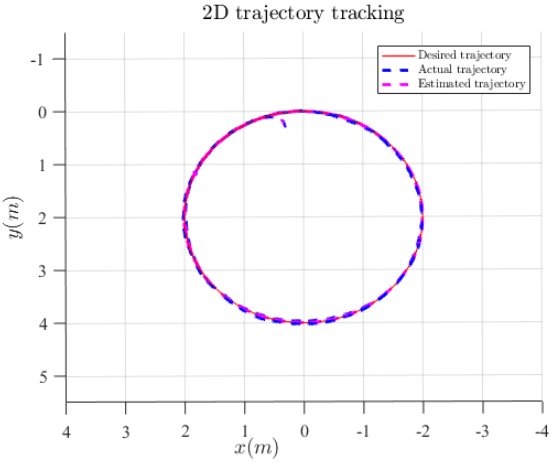

Figure 4.23: EKF-based NMPC trajectory tracking in the $x-y$ plane

4.6 Comparison and results discussion

In the previous sections, we designed two control techniques for quadrotor control, PD and NMPC controllers combined with EKF for state estimation. While PD and NMPC share the common goal of achieving desired trajectories, they differ in their underlying principles, control strategies, and performance characteristics. Understanding the advantages and limitations of each approach is very important. Therefore, in this section we compare and assess the performance of the two controllers and determine their suitability for specific control objectives and applications.

4.6.1 Full state controllers

In this section, we delve into a comparison between NMPC and PD controllers for quadrotors. We will analyze their strengths and weaknesses, and evaluate their performance in terms of tracking accuracy, response time, input effort, computational requirements, and implementation complexity. To provide a comprehensive evaluation, we summarize the key performance metrics of PD and NMPC controllers in table 4.5.

Controller	PD			NMPC		
Evaluation	Error(%)	$T_{r(5\%)}(s)$	Overshoot (%)	Error (%)	$T_{r(5\%)}(s)$	Overshoot (%)
x	2.9846	7.63	/	0.0026	2.70	/
y	1.8582	11.45	/	0.0016	4.35	/
z	1.0992	4.45	/	0.0521	1.45	/
ψ	0.0160	1.63	0	0.0037	1.65	3.2659

Table 4.5: Performance metrics of full-state controllers

Results discussion

- The PD offered promising results for the ψ and z axis. However, it is a little bit less accurate for the x and y axis, for instance, the x axis exhibited a relative error that is close to 3%. This is due to complex x and y dynamics and their coupling, requiring a cascade control structure.

- Despite its simplicity and easy implementation, the PD controller achieved satisfying results for the quadrotor's trajectory tracking problem. Its remarkable performance is also due to the fine tuning of its parameters achieved using the Genetic Algorithm.

- The Genetic Algorithm is a highly efficient method for fine-tuning PD gains, particularly for complex nonlinear systems with numerous parameters to optimize, such as quadrotors. This approach enables us to save valuable time by tuning all 12 parameters simultaneously.

- NMPC outperforms the PD controller with significantly smaller tracking errors. Additionnaly, the PD exhibited oscillations in the beginning of the simulation for the x and y axis, while the NMPC eliminated these oscillations. This can be attributed to its advanced model-based approach and online optimization, resulting in increased precision. NMPC effectively handles complex dynamics and constraints, leading to remarkable tracking accuracy, highlighting its superiority over traditional PD control.

- The response time $T_{r(5\%)}$ is significantly smaller for NMPC compared to PD control.

- For the PD, the ψ angle didn't have an overshoot, while with the NMPC controller it had a small overshoot of 3.2%, this is due to computational limitations that hinder NMPC's ability to respond promptly to abrupt changes.

- For the PD, the control actions in the beginning of the simulation have a higher peak compared to the NMPC control actions.

- NMPC heavily relies on accurate system models to make predictions and optimize control inputs. Therefore, it is sensitive to model inaccuracies, if the model does not capture the quadrotor's dynamics accurately, the control performance may degrade.

- Setting up an NMPC controller for quadrotor control requires tuning several parameters, including the prediction horizon, control weighting matrices, and constraint handling. The selection of an appropriate horizon length is crucial for the quadrotor's trajectory tracking problem. Finding an appropriate parameter set can be challenging and often involves iterative trial-and-error.

- Another important consideration in comparing the NMPC and PD is the computational time. It is observed that the computational time for the PD is significantly shorter than that of the NMPC. The PD operates based on simple feedback control, requiring fewer and less complex calculations compared to NMPC. This computational advantage of the PD allows for faster execution and real-time responsiveness, making it suitable for real-time applications. On the other hand, NMPC involves solving an optimization problem over a horizon at each iteration, which can be very computationally demanding especially for complex nonlinear systems like quadrotors and with longer prediction horizons, limiting its real-time applicability especially on resource-limited hardware. However, advances in computational hardware and algorithmic optimizations continue to improve the computational efficiency of NMPC, narrowing the gap between the two control approaches.

- The use of the CasADi optimization framework for NMPC implementation has significantly improved computational efficiency. CasADi provides efficient numerical optimization solvers and symbolic computation capabilities, leading to faster computation. Studies have shown that CasADi can be up to 20 times faster than other numerical optimization approaches such as the MATLAB MPC toolbox and achieves smaller tracking error [94].

- As CasADi offers faster computational speed compared to conventional MPC toolboxes, it provides a significant advantage for real-time implementation, which is crucial for our application. Additionally, the integration of MATLAB with CasADi serves as a solid basis for developing advanced control algorithms for quadrotor systems. Our control technique leverages this integration effectively, enabling seamless modeling, simulation, and performance evaluation of the entire system across various operating scenarios.

In summary, PD control is sufficient enough for certain applications that don't require very high tracking accuracy, making it well-suited for resource-limited hardware. On the other hand, NMPC controller is more suitable for sophisticated applications that demand high tracking accuracy, although it requires significant computational efficiency.

4.6.2 EKF-based controllers

In this section, we will assess the performance of PD and NMPC controllers combined with EKF estimation for quadrotor systems. By incorporating EKF estimation, we can account for the unavailability of certain states and sensor noise, making the control process more

realistic. This comparison will provide insights to help select the most suitable estimation-based control strategy for the quadrotor system. Table 4.6 allows for an assessment of the performance of EKF-based control.

Controller	EKF-based PD			EKF-based NMPC		
Evaluation	Error(%)	$T_{r(5\%)}(s)$	Overshoot	Error (%)	$T_{r(5\%)}(s)$	Overshoot
x	3.0329	7.63	/	1.6385	3.00	/
y	1.9867	11.82	/	0.7873	3.80	/
z	1.1993	4.43	/	1.0077	3.75	/
ψ	0.0120	1.63	0	0.0700	1.65	3.4377

Table 4.6: Performance metrics of EKF-based controllers

Results discussion

- EKF estimation is highly effective for estimating unavailable system states and filtering measurements based on noisy sensor data, the obtained estimation results are very accurate. Therefore, EKF is very suited for nonlinear systems such as quadrotors.
- Accurate feedback signal is crucial for precise control in PD and NMPC controllers. EKF combined with both these two control techniques exhibited great effectiveness for providing accurate estimates to use for the feedback control.
- This integration of the EKF into the control system enhances realism by generating more reliable state estimates that closely resemble the true system behavior. The accurate and realistic state estimation enables the controllers to make more informed decisions and generate control actions that align closely with the desired behavior of the quadrotor.
- The choice of Q_ξ and R_υ matrices, representing the uncertainties in the system dynamics and measurements respectively, is very important for tuning the EKF and achieving accurate state estimation. However, tuning Q_ξ and R_υ by trial and error can be time consuming.
- The comparison reveals that tracking errors are slightly larger in the estimation-based approaches compared to full-state controllers. This difference can be attributed to the fact that EKF takes into account the noise present in the real-world system, resulting in more realistic and accurate state estimates. On the other hand, in the simulation without estimation, perfect knowledge of the system is assumed, neglecting the effects of noise. Therefore, while the errors may be slightly larger in the estimation-based approaches, it is important to note that these approaches provide a more realistic representation of the quadrotor system and can handle real-world conditions more effectively.
- The integration of EKF estimation in control strategies introduces an additional computational overhead, particularly for NMPC. The estimation process involves calculations for state propagation and covariance matrix updates, contributing to increased computational time. This can be a significant consideration, especially for NMPC, which already has a relatively high computational burden. However,

it is important to weigh this increased computational cost against the benefits of a more realistic approach. Advances in computational hardware and algorithmic optimizations, along with the utilization of efficient numerical tools, can help alleviate some of the computational challenges associated with estimation-based NMPC. Ultimately, the trade-off between computational time and enhanced control performance needs to be carefully evaluated to determine the most suitable approach for quadrotor control applications.

4.7 Conclusion

In this chapter, the control and estimation problem for a single quadrotor was investigated, we employed two control techniques to address the trajectory tracking problem. First, we designed an optimized PD controller using GA to fine-tune its parameters. Subsequently, we developed an NMPC controller, this was done by reformulating the OCP into an NLP using the multiple shooting technique. To implement this controller efficiently, we took advantage of the computational advantages of the CasADi optimization framework. The developed control strategies were coupled with EKF estimation to make the control strategies more realistic, both techniques were validated through simulations. Furthermore, a comparison between the two techniques was conducted to evaluate their performance.

This chapter serves as a basis for the next chapter about the formation of multiple quadrotors. Indeed, the single quadrotor controlled in this chapter is going to be considered the leader quadrotor in the context of leader-follower formation control developed in the next chapter.

Chapter 5

Formation control of multi-quadrotor systems

5.1 Introduction

In this chapter, the formation problem of multi-quadrotor systems is investigated, particularly, the rigid formation task where the desired inter-distance between the quadrotors is constant, resulting in a 'V' shape fixed pattern . To achieve this, we employ the leader-follower formation control strategy. The single quadrotor controlled in the previous chapter is considered the leader, meanwhile, for the followers we use two controllers. The first one is a Lyapunov based formation controller to keep the inter-distance in the $x-y$ plane and orientation between the leader and the followers, this is done by developing a nonlinear dynamics model of the formation error, allowing the formation pattern to be maintained. Additionnaly, a PD and NMPC controllers are employed for trajectory tracking of each quadrotor in the formation. To do this, the previously designed PD and NMPC controllers for a single quadrotor are adapted for the leader-follower problem. For the NMPC controller, we propose an original prediction-based leader-follower formation control scheme. For each agent in the group, state estimation using EKF is done based on noisy sensor measurement. Numerical simulations demonstrate the effectiveness of the proposed techniques in maintaining the formation. Furthermore, a comparison between the two control techniques is conducted to evaluate their performance.

5.2 Leader-follower formation control

We consider a group of n quadrotors described by the model developed in section 3.3, with one leader $i = L$ and $n-1$ followers, $i \in \{F_1, ..F_{n-1}\}$, the i^{th} quadrotor is described by:

$$\begin{cases} \dot{X}_i = F(X_i, U_i), & i \in \{L, F_1, F_2, ...F_{n-1}\}, \\ Y_i = h(X_i), \\ Z_i = h^{mes}(X_i), \end{cases} \tag{5.1}$$

in this work, we limit the formation task to three quadrotors, one leader $i = L$ and two followers $i = F_1$ and $i = F_2$. In this section, we design controllers for each ith quadrotor in the formation to maintain a desired 'V' shape in the $x-y$ plane shown in Fig. 5.1.

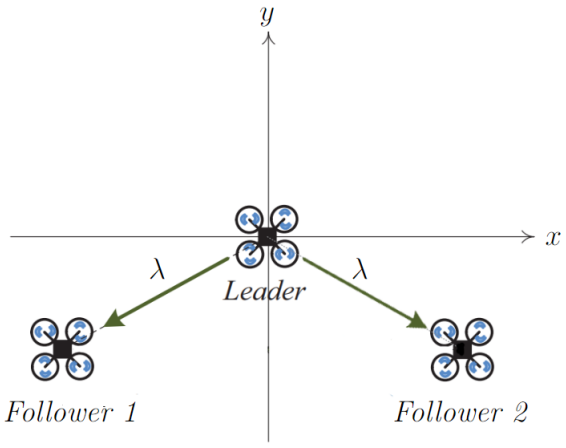

Figure 5.1: Desired formation pattern, from [100]

The leader tracks a designated trajectory, while the followers are positioned at a specific distance from the leader and have a particular orientation to create the desired formation shape. The leader control has been developed in the previous chapter, in this chapter, we focus on the control of the followers. It is important to note that the followers aren't given a predefined trajectory to track, their reference trajectories solely depend on the state of the leader.

5.2.1 Formation controller

In the process of formation flight, the leader follows a predetermined trajectory, while for the followers, a formation controller is implemented to maintain a specific shape in the $x-y$ plane, which is determined by the relative kinematics between the leader and the followers [101]. The formation controller generates the reference velocities for the followers, which then track these references to maintain a relative distance and orientation. The leader-follower formation control scheme is shown in Fig. 5.2.

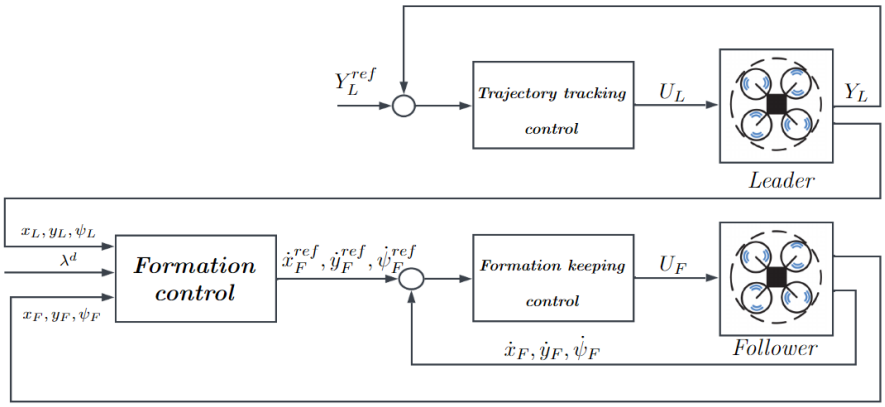

Figure 5.2: Leader-follower control scheme

For each follower, the formation controller takes the leader's x and y positions, and ψ orientation and generates the follower's reference velocities $\dot{x}$, $\dot{y}$, and $\dot{\psi}$ in order to keep a fixed distance and a fixed deviation in the $x-y$ plane between the leader and the follower [69]. This is shown in Fig. 5.3.

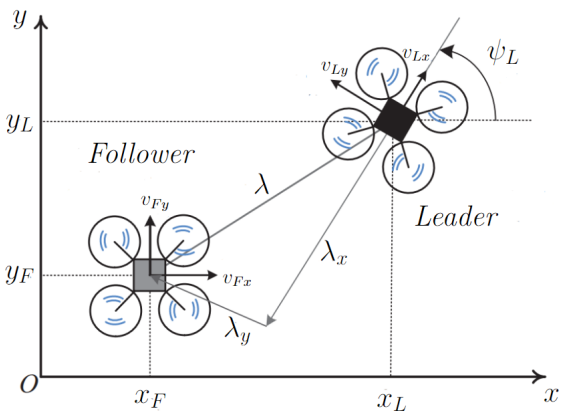

Figure 5.3: Quadrotors formation in the $x-y$ plane, from [100]

We consider the vertical subsystem and the horizontal subsystem as completely decoupled, Therefore the control design is for each subsystem is done separately. Also, the trajectory dynamics of each quadrotor are slower than attitude dynamics. Therefore, we can decompose the control scheme into an inter-loop and outer-loop, and the treat the formation controller separately from the quadrotor's model [100].

We start with the relative translational kinematics in the $x-y$ plane at a fixed z altitude. $\dot{x}_i$, $\dot{y}_i$ are the velocity components in the earth-fixed frame and v_{ix}, v_{iy} are the velocity components in the body-fixed frame. For each quadrotor $i \in \{L, F_1, F_2\}$, the translational dynamics in the $x-y$ plane are given by:

$$\dot{x}_i = v_{ix}\cos\psi_i - v_{iy}\sin\psi_i, \tag{5.2}$$

$$\dot{y}_i = v_{ix}\sin\psi_i + v_{iy}\cos\psi_i, \tag{5.3}$$

similarly we can write:

$$v_{ix} = \dot{x}_i\cos\psi_i + \dot{y}_i\sin\psi_i, \tag{5.4}$$

$$v_{iy} = -\dot{x}_i\sin\psi_i + \dot{y}_i\cos\psi_i, \tag{5.5}$$

we consider λ_i the distance between the center of mass of the leader and the followers $i \in \{F_1, F_2\}$, λ_{ix} and λ_{iy} are the x and y components in the leader's body frame. For each follower $i \in \{F_1, F_2\}$, we can write:

$$\lambda_{ix} = -(x_L - x_i)\cos\psi_L - (y_L - y_i)\sin\psi_L, \tag{5.6}$$

$$\lambda_{iy} = (x_L - x_i)\sin\psi_L - (y_L - y_i)\cos\psi_L, \tag{5.7}$$

by differentiating:

$$\dot{\lambda}_{ix} = -(\dot{x}_L - \dot{x}_i)\cos\psi_L + (x_L - x_i)\dot{\psi}_L\sin\psi_L - (\dot{y}_L - \dot{y}_i)\sin\psi_L - (y_L - y_i)\dot{\psi}_L\cos\psi_L, \tag{5.8}$$

using the equations (5.4) and (5.7), we can write:

$$\dot{\lambda}_{ix} = \lambda_{iy}\dot{\psi}_L + \dot{x}_i\cos\psi_L + \dot{y}_i\sin\psi_L - v_{Lx}, \tag{5.9}$$

using the equations (5.2) and (5.3):

$$\dot{\lambda}_{ix} = \lambda_{iy}\dot{\psi}_L + (v_{ix}\cos\psi_i - v_{iy}\sin\psi_i)\cos\psi_L + (v_{ix}\sin\psi_i + v_{iy}\cos\psi_i)\sin\psi_L - v_{Lx}, \tag{5.10}$$

$$\dot{\lambda}_{ix} = \lambda_{iy}\dot{\psi}_L + v_{ix}(\cos\psi_i\cos\psi_L + \sin\psi_i\sin\psi_L) + v_{iy}(\cos\psi_i\sin\psi_L - \sin\psi_i\cos\psi_L) - v_{Lx}, \tag{5.11}$$

we use the following trigonometric identities:

$$\cos(a-b) = \cos a\cos b + \sin a\sin b, \tag{5.12}$$

$$\sin(a-b) = \sin a\cos b - \cos a\sin b, \tag{5.13}$$

let λ_i^d be the desired constant distance. We define the formation errors for $i \in \{F_1, F_2\}$:

$$e_{ix} = \lambda_{ix}^d - \lambda_{ix}, \tag{5.14}$$

$$e_{iy} = \lambda_{iy}^d - \lambda_{iy}, \tag{5.15}$$

$$e_{i\psi} = \psi_i - \psi_L, \tag{5.16}$$

we get:

$$\dot{\lambda}_{ix} = \lambda_{iy}\dot{\psi}_L + v_{ix}\cos e_{i\psi} - v_{iy}\sin e_{i\psi} - v_{Lx}, \tag{5.17}$$

similarly for the y component:

$$\dot{\lambda}_{iy} = -\lambda_{ix}\dot{\psi}_L + v_{ix}\sin e_{i\psi} + v_{iy}\cos e_{i\psi} - v_{Ly}, \tag{5.18}$$

therefore, the formation errors dynamics are given by:

$$\dot{e}_{ix} = -(\lambda^d_{iy} - e_{iy})\dot{\psi}_L - v_{ix}\cos e_{i\psi} + v_{iy}\sin e_{i\psi} + v_{Lx}, \tag{5.19}$$

$$\dot{e}_{iy} = (\lambda^d_{ix} - e_{ix})\dot{\psi}_L - v_{ix}\sin e_{i\psi} - v_{iy}\cos e_{i\psi} + v_{Ly}, \tag{5.20}$$

$$\dot{e}_{i\psi} = \dot{\psi}_i - \dot{\psi}_L, \tag{5.21}$$

the formation controller is designed to stabilize the formation errors using the inputs v_{ix}, v_{iy} and $\dot{\psi}_i$, $i \in \{F_1, F_2\}$ which are used later as reference velocities for each follower.

$$\dot{e}_i = A_i(e_i) + B_i(e_i)v_i, \tag{5.22}$$

$$A_i(e_i) = \begin{bmatrix} -(\lambda^d_{iy} - e_{iy})\dot{\psi}_L + v_{Lx} \\ (\lambda^d_{ix} - e_{ix})\dot{\psi}_L + v_{Ly} \\ -\dot{\psi}_L \end{bmatrix}, \tag{5.23}$$

$$B_i(e_i) = \begin{bmatrix} -\cos e_{i\psi} & \sin e_{i\psi} & 0 \\ -\sin e_{i\psi} & -\cos e_{i\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.24}$$

where $e_i = [e_{ix}\ e_{iy}\ e_{i\psi}]^T$ and $v_i = [v_{ix}\ v_{iy}\ \dot{\psi}_i]^T$.

Proposition 5.1 (Lyapunov based formation controller)
Given the leader-follower dynamical model in (5.1), then the reference velocities given by the following equation

$$v_i^{ref} = B_i^{-1}(e_i)(-A_i(e_i) - K_i e_i) \tag{5.25}$$

guarantee the stability of the formation errors $e_i = [e_{ix}\ e_{iy}\ e_{i\psi}]^T$ between the leader and the ith follower [101].

Proof. Consider the positive definite Lyapunov candidate function be defined as follows:

$$V(e_i) = \frac{1}{2}e_i^T e_i, \tag{5.26}$$

by differentiating:

$$\dot{V}(e_i) = e_i^T \dot{e}_i = e_i^T[A_i(e_i) + B_i(e_i)v_i], \tag{5.27}$$

to satisfy the Lyapunov stability condition, $\dot{V}$ has to be a negative definite function. Therefore, the reference velocities v_i^{ref} are given as follows:

$$v_i^{ref} = [v_{ix}^{ref}\ v_{iy}^{ref}\ \dot{\psi}_i^{ref}]^T = B_i^{-1}(e_i)(-A_i(e_i) - K_i e_i), \tag{5.28}$$

$K_i = diag(K_{ix}, K_{iy}, K_{i\psi})$ is a diagonal positive definite matrix. We tune this gain so that the formation controller errors converge to zero. Then:

$$\dot{V} = -e_i^T K_i e_i < 0. \tag{5.29}$$

□

therefore, this Lyapunov-based formation controller allows the formation errors to converge to zero, allowing the followers to maintain the desired formation pattern.

The generated v_{ix}^{ref} and v_{iy}^{ref} velocities are defined in the body frame, the necessary transform to the earth-frame is done to get $\dot{x}_i^{ref}$ and $\dot{y}_i^{ref}$. In the next sections, the followers are controlled to track these reference velocities, for this, we adapt the previously designed PD and NMPC controllers of the leader to the followers case.

5.2.2 Optimal PD control of the followers

In this section, the optimized PD controller using GA is adapted for the followers to maintain the formation pattern. Since the formation controller generated desired velocities for the followers, we control the follower's velocities to track their references. For each follower, six PD controllers are employed, the same cascade structure as for the leader is used. The gains are tuned using GA for each follower. The input vector of the *ith* follower is denoted $U_i = [u_{1i} \quad u_{2i} \quad u_{3i} \quad u_{4i}]^T$, for $i \in \{F_1, F_2\}$.

5.2.2.1 Outer loop for translational subsystem

For each follower $i \in \{F_1, F_2\}$, the outer loop determines the follower's desired pitch and roll angles $(\theta_i^{ref}, \phi_i^{ref})$ based on the desired v_{ix} and v_{iy} velocities in the body frame generated by the formation controller.

$$e_{vx}^i(t) = v_{ix}^{ref}(t) - v_{ix}(t), \tag{5.30}$$

$$e_{vy}^i(t) = v_{iy}^{ref}(t) - v_{iy}(t), \tag{5.31}$$

where e_{vx}^i, e_{vy}^i are the velocity tracking errors in the body frame. This outer loop position controller consists of two Proportional (P) controllers that take the e_{vx}^i, e_{vy}^i errors and generates the ϕ_i and θ_i references.

$$\phi_i^{ref}(t) = k_{Py}^i e_{vy}^i(t), \tag{5.32}$$

$$\theta_i^{ref}(t) = k_{Px}^i e_{vx}^i(t), \tag{5.33}$$

where k_{Px}^i , k_{Py}^i are the P gains for the x and y axis respectively for the follower.

5.2.2.2 Inner loop for rotational subsystem

The inner loop controls the followers' rotational subsystem (ϕ_i, θ_i), it has two PD controllers that generate the roll and pitch moments u_{2i} and u_{3i}.

$$u_{2i} = k_{P\phi}^i e_\phi^i(t) + k_{D\phi}^i \dot{e}_\phi^i(t), \tag{5.34}$$

$$e_\phi^i(t) = \phi_i^{ref}(t) - \phi_i(t), \tag{5.35}$$

$$u_{3i} = k_{P\theta}^i e_\theta^i(t) + k_{D\theta}^i \dot{e}_\theta^i(t), \tag{5.36}$$

$$e_\theta^i(t) = \theta_i^{ref}(t) - \theta_i(t), \tag{5.37}$$

where $k_{P\phi}^i$ and $k_{D\phi}^i$ are the PD gains of the roll controller while $k_{P\theta}^i$ and $k_{D\theta}^i$ are the PD gains of the pitch controller.

5.2.2.3 Altitude and yaw control

The z axis is controlled in position just like for the leader, we use a PD controller.

$$u_{1i} = \frac{mg + k^i_{Pz} e^i_z(t) + k^i_{Dz} \dot{e}^i_z(t)}{\cos\phi_i \cos\theta_i}, \tag{5.38}$$

$$e^i_z(t) = z^{ref}_i(t) - z_i(t). \tag{5.39}$$

For the yaw controller, we use a Proportional action to control the yaw velocity to the reference generated by the formation controller.

$$u_{4i} = k^i_{P\psi} e^i_\psi(t), \tag{5.40}$$

$$e^i_\psi(t) = \dot{\psi}^{ref}_i(t) - \dot{\psi}_i(t), \tag{5.41}$$

where k^i_{Pz} and k^i_{Dz} are the PD gains of the z axis while $k^i_{P\psi}$ is the P gain of the yaw controller.

5.2.2.4 Genetic Algorithm for PD tuning

As for the leader, we use GA for the followers $i \in \{F_1, F_2\}$ to optimize the PD gains. The objective function for the ith follower is given by the sum of the cost functions of each axis.

$$\underset{\Theta_i}{minimize}\ J^i_G(\Theta_i) = J^i_x + J^i_y + J^i_z + J^i_\phi + J^i_\theta + J^i_\psi, \tag{5.42}$$

where $\Theta_i = [k^i_{Px}\ k^i_{Py}\ k^i_{Pz}\ k^i_{Dz}\ k^i_{P\phi}\ k^i_{D\phi}\ k^i_{P\theta}\ k^i_{D\theta}\ k^i_{P\psi}]^T$ is the followers' vector of parameters to be optimized. The cost functions for each axis are a weighted sum of ISE and ISC indices.

$$J^i_x = \int_0^{t_f} w_1 (e^i_{vx}(t))^2 dt, \tag{5.43}$$

$$J^i_y = \int_0^{t_f} w_1 (e^i_{vy}(t))^2 dt, \tag{5.44}$$

$$J^i_z = \int_0^{t_f} [w_1 (e^i_z(t))^2 + w_2 u^2_{1i}(t)] dt, \tag{5.45}$$

$$J^i_\phi = \int_0^{t_f} [w_1 (e^i_\phi(t))^2 + w_2 u^2_{2i}(t)] dt, \tag{5.46}$$

$$J^i_\theta = \int_0^{t_f} [w_1 (e^i_\theta(t))^2 + w_2 u^2_{3i}(t)] dt, \tag{5.47}$$

$$J^i_\psi = \int_0^{t_f} [w_1 (e^i_\psi(t))^2 + w_2 u^2_{4i}(t)] dt. \tag{5.48}$$

t_f is the final simulation time. w_1, w_2 are the ISE and ISC optimization weights respectively, they are the same as for the leader, they must satisfy $w_1 + w_2 = 1$.

5.2.3 Application of NMPC to the followers

In this section, the previously designed NMPC controller for the leader is adapted to the followers to keep the desired formation. As for the leader, an Optimization Control Problem (OCP) is formulated. The OCP is then discretized using the multiple shooting technique and reformulated into a Nonlinear Programming Porblem (NLP).

5.2.3.1 Discretization

For each follower, the prediction horizon is discretized into N shooting intervals. At each instant k, the NMPC algorithm is executed over the horizon $[k, k+N]$ and a sequence of optimal controls is computed for the *ith* follower, $i \in \{F_1, F_2\}$. The matrix $P_{i,U}(k)$ is filled with optimal control actions $U_i(k+j)$ over the horizon, $j \in [0, N-1]$:

$$P_{i,U}(k) = [U_i(k), U_i(k+1), ..., U_i(k+N-1)], \tag{5.49}$$

the state prediction matrix corresponding to $P_{i,U}(k)$ over the horizon is $P_{i,X}(k)$:

$$P_{i,X}(k) = \left[X_{i,U}(k), X_{i,U}(k+1), ..., X_{i,U}(k+N)\right], \tag{5.50}$$

$X_{i,U}(k+j)$ is the predicted state corresponding to the optimal input $U_i(k+j-1)$, $j \in [1, N]$. The NMPC problem also requires having the references over the whole horizon. Let $X_i^{ref}(k)$ be the reference of the states. Therefore, the reference matrix is:

$$R_{i,X}(k) = [X_i^{ref}(k), X_i^{ref}(k+1), ..., X_i^{ref}(k+N)]. \tag{5.51}$$

5.2.3.2 Objective function

The running cost for the *ith* follower penalizes the difference between the state vector and its reference, as well as the control energy, it is given by:

$$\begin{aligned} L_i(X_{i,U}(k+j), U_i(k+j)) = [X_{i,U}(k+j) - X_i^{ref}(k+j)]^T Q_i [X_{i,U}(k+j) - X_i^{ref}(k+j)] \\ + U_i^T(k+j) R_i U_i(k+j), \end{aligned} \tag{5.52}$$

Q_i and R_i are the state and input weighting matrices respectively for the followers. The cost function is the evaluation of the running costs along the whole horizon:

$$J_N^i = \sum_{j=0}^{N-1} L_i \left(X_{i,U}(k+j), U_i(k+j)\right), \tag{5.53}$$

for $j \in [0, N-1]$ and $i \in \{F_1, F_2\}$.

5.2.3.3 Optimization variable

The decision variable W_i for each follower in the context of multiple shooting contains both the inputs and states, it is given by:

$$W_i = [U_i^T(k), U_i^T(k+1), ..., U_i^T(k+N-1), X_{i,U}^T(k), X_{i,U}^T(k+1), ..., X_{i,U}^T(k+N)]^T. \tag{5.54}$$

5.2.3.4 Dynamics constraints and bounds

For each follower, the dynamics model is discretized and imposed as a constraint between adjacent intervals. We write the system model in the discrete time domain:

$$X_i(k+1) = X_i(k) + T * F(X_i(k), U_i(k)) = f(X_i(k), U_i(k)), \tag{5.55}$$

this equality constraint is considered at each shooting step as follows:

$$f(X_{i,U}(k), U_i(k)) - X_{i,U}(k+1) = 0, \tag{5.56}$$

we also consider bounds on control inputs and states:

$$\begin{aligned} U_i(k+j) &\in \mathcal{U}, \quad \forall j \in [0, N-1], \\ X_{i,U}(k+j) &\in \mathcal{X}, \quad \forall j \in [0, N], \end{aligned} \tag{5.57}$$

where:

$$\begin{aligned} \mathcal{U} &= \{U_i \in \mathbb{R}^m \mid U_i^{min} \leq U_i \leq U_i^{max}\}, \\ \mathcal{X} &= \{X_{i,U} \in \mathbb{R}^n \mid X_i^{min} \leq X_{i,U} \leq X_i^{max}\}. \end{aligned} \tag{5.58}$$

5.2.3.5 The resulting Nonlinear Programming Problem

The resulting NMPC Nonlinear Programming Problem for the *ith* follower can be written as follows:

$$\begin{aligned} \underset{\mathbf{W_i}}{minimize} \quad & J_N^i = \sum_{j=0}^{N-1} L_i \left(X_{i,U}(k+j), U_i(k+j)\right), \\ s.t. \quad & X_{i,U}(k+1) = f\left(X_{i,U}(k), U_i(k)\right), \\ & X_{i,U}(0) = X_0^i, \\ & U_i(k+j) \in \mathcal{U}, \qquad \forall j \in [0, N-1], \\ & X_{i,U}(k+j) \in \mathcal{X}, \quad \forall j \in [0, N], \\ for\ all\ & i \in \{F_1, F_2\}, \end{aligned} \tag{5.59}$$

this NLP is solved using the IPOPT solver in the optimization framework CasADi just like for the leader.

5.2.3.6 Prediction based leader-follower NMPC

In order to compute the optimal control inputs for the followers, the NMPC algorithm requires the followers' references throughout the entire horizon $[k, k+N]$ at each time step. These references are determined solely based on the leader's states. Thus, to generate the followers' references using the formation controller, we need the leader's states over the entire horizon at each instant. However, since this information is not available yet at time step k, we address this issue by utilizing the leader's predicted states in the formation controller. This approach enables us to generate the followers' references without relying on the leader's real states. For this, we use the following leader's state prediction matrix:

$$P_{L,X}(k) = \left[X_{L,U}(k), X_{L,U}(k+1), ..., X_{L,U}(k+N)\right], \tag{5.60}$$

at each instant k, we extract the leader's predictions of the states $\dot{x}_{L,U}(k+j)$, $\dot{y}_{L,U}(k+j)$, $\dot{\psi}_{L,U}(k+j)$ and $\psi_{L,U}(k+j)$ over the horizon $j \in [0, N]$. We transform these velocities to the body frame as follows:

$$\begin{aligned} v_{Lx}(k+j) &= \dot{x}_{L,U}(k+j)cos(\psi_{L,U}(k+j)) + \dot{y}_{L,U}(k+j)sin(\psi_{L,U}(k+j)), \\ v_{Ly}(k+j) &= -\dot{x}_{L,U}(k+j)sin(\psi_{L,U}(k+j)) + \dot{y}_{L,U}(k+j)cos(\psi_{L,U}(k+j)), \end{aligned} \tag{5.61}$$

and we use them in the formation controller over the horizon $j \in [0, N]$. The discretized formation error dynamics can be written as:

$$e_i(k+j+1) = e_i(k+j) + T * [A_i(e_i(k+j)) + B_i(e_i(k+j))v_i(k+j)], \tag{5.62}$$

where $e_i(k+j) = [e_{ix}(k+j), e_{iy}(k+j), e_{i\psi}(k+j)]^T$ are the formation controller errors and $v_i(k+j) = [v_{ix}(k+j), v_{iy}(k+j), \dot{\psi}_i(k+j)]^T$ are the followers' velocities. We compute their references over the horizon $j \in [0, N]$ for each follower $i \in \{F_1, F_2\}$:

$$v_i^{ref}(k+j) = B_i^{-1}(e_i(k+j))[-A_i(e_i(k+j)) - K_i e_i(k+j)], \tag{5.63}$$

now that we have the followers' references over the horizon, we transform them back in the earth frame and we use them in the optimization problem.

Additionally, since the leader's position predictions are available too, we can compute the followers' desired position based on leader's information and add it to the followers' cost function to enhance their tracking performance. This is done as follows:

$$\begin{bmatrix} x_i^{ref}(k+j) \\ y_i^{ref}(k+j) \end{bmatrix} = \begin{bmatrix} x_{L,U}(k+j) \\ y_{L,U}(k+j) \end{bmatrix} + \begin{bmatrix} \cos\psi_{L,U}(k+j) & -\sin\psi_{L,U}(k+j) \\ \sin\psi_{L,U}(k+j) & \cos\psi_{L,U}(k+j) \end{bmatrix} \begin{bmatrix} \lambda_{ix}^d \\ \lambda_{iy}^d \end{bmatrix}, \tag{5.64}$$

therefore, the followers' position reference consists of the leader's predicted position plus the desired inter-distance between the leader and the followers'. This distance is defined in the body-frame, the necessary transform is done to bring it to the earth frame. We fill the generated velocity and position references in the followers reference vector $X_i^{ref}(k+j)$ over the horizon $j \in [0, N]$.

5.2.4 Application of EKF to the followers

Each follower in the formation is equipped with the same sensors as the leader, detailed in section 4.4.3. We use EKF state estimation for each follower. Process and measurement noise are added to each follower's system model and measurement model for $i \in \{F_1, F_2\}$.

$$\dot{X}_i(t) = F(X_i, U_i) + \xi_i(t), \tag{5.65}$$

$$Z_i(t) = h^{mes}(X_i) + \upsilon_i(t), \tag{5.66}$$

$$h^{mes}(X_i) = [x_i \quad y_i \quad z_i \quad \phi_i \quad \theta_i \quad \psi_i]^T, \tag{5.67}$$

$\xi_i(t)$, $\upsilon_i(t)$ are white Gaussian process and measurement noises respectively. Q_ξ^i and R_υ^i are the covariances of the process noise and the measurement noise respectively. Each follower's model is discretized and the EKF algorithm detailed in section 4.4.3 is applied.

5.3 Simulation results

In this section, we present the simulation results of the formation of three quadrotors (one leader and two followers) in a 'V' shape. There are two control objectives:

- The tracking of the desired trajectory by the leader $(x_L^{ref}(t), y_L^{ref}(t), z_L^{ref}(t), \psi_L^{ref}(t)) = (2\sin 0.1t, 2 - 2\cos 0.1t, 0.1t, 0)$, these results have been presented in the previous chapter. It is worth recalling that this trajectory is not given to the followers. The initial linear position of the leader is $r_{xyz,0}^L = [0\ 0\ 0]^T$.

- The formation keeping by the two followers described by the desired inter-distance and deviation from the leader:
  - First follower: the initial position is $r^{F1}_{xyz,0} = [-1\ -1\ 0]^T$. The desired distance is $\lambda^d_{F1} = 2\sqrt{2}$ ($\lambda^d_{F1x} = -2$, $\lambda^d_{F1y} = -2$). The desired yaw deviation from the leader is $\psi_{F1} - \psi_L = 0$.
  - Second follower: the initial position is $r^{F2}_{xyz,0} = [-1\ 1\ 0]^T$. The desired distance is $\lambda^d_{F2} = 2\sqrt{2}$ ($\lambda^d_{F2x} = -2$, $\lambda^d_{F2y} = 2$). The desired yaw deviation from the leader is $\psi_{F2} - \psi_L = 0$.

The followers' model parameters (m, I_x, I_y, I_z) are identical to the leader. We present the simulation results only for the first follower, as it exhibits symmetric behavior to the second follower. Then, we conclude with a simulation of the formation of three quadrotors.

5.3.1 PD controller results

After executing the genetic algorithm, we get the optimal PD parameters for the followers in table 5.1. We used the same parameters for the two followers, $i \in \{F_1, F_2\}$. The formation controller gains are $K_{F1} = K_{F2} = diag(1, 1, 1)$.

Axis	k^i_P	k^i_D	Saturation
$\dot{x}_i$	1.3265	0	$\lvert\theta^{ref}_i\rvert \leq 1.05\ [rad]$
θ_i	6.4446	2.3152	$\lvert u_{3i}\rvert \leq 2\ [N.m]$
$\dot{y}_i$	1.3265	0	$\lvert\phi^{ref}_i\rvert \leq 1.05\ [rad]$
ϕ_i	6.4446	2.3152	$\lvert u_{2i}\rvert \leq 2\ [N.m]$
z_i	7.5220	2.8205	$\lvert u_{1i}\rvert \leq 2.1\ [N]$
$\dot{\psi}_i$	1.1153	0	$\lvert u_{4i}\rvert \leq 2\ [N.m]$

Table 5.1: Followers' optimal PD simulation parameters

Trajectory tracking results for the first follower:

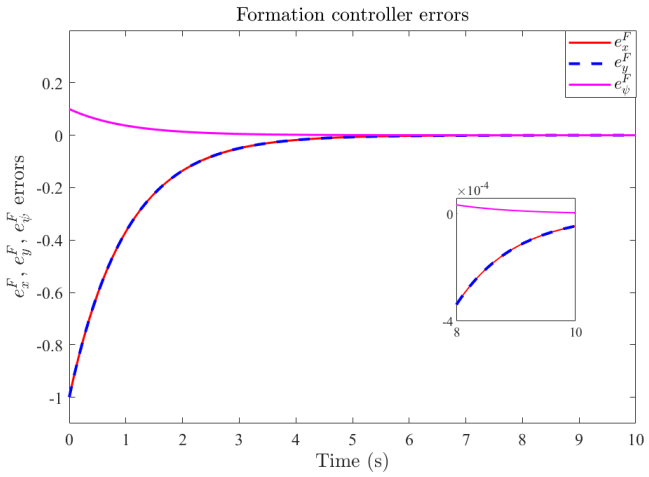

Figure 5.4: PD formation controller errors

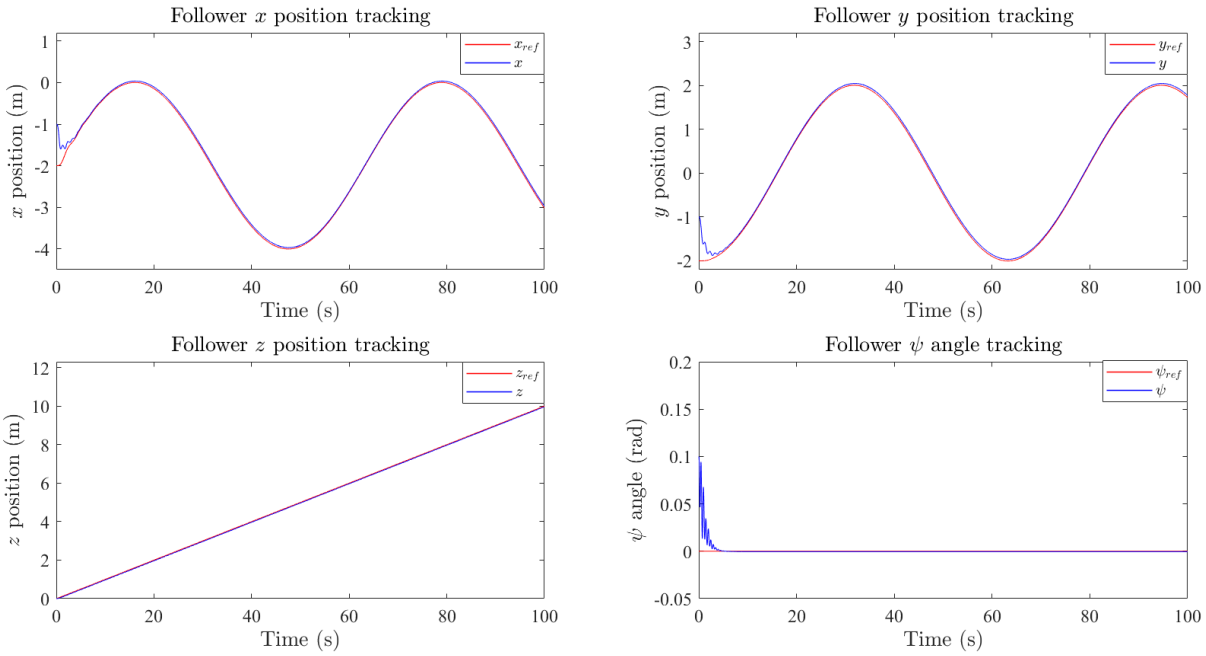

Figure 5.5: Follower PD trajectory tracking

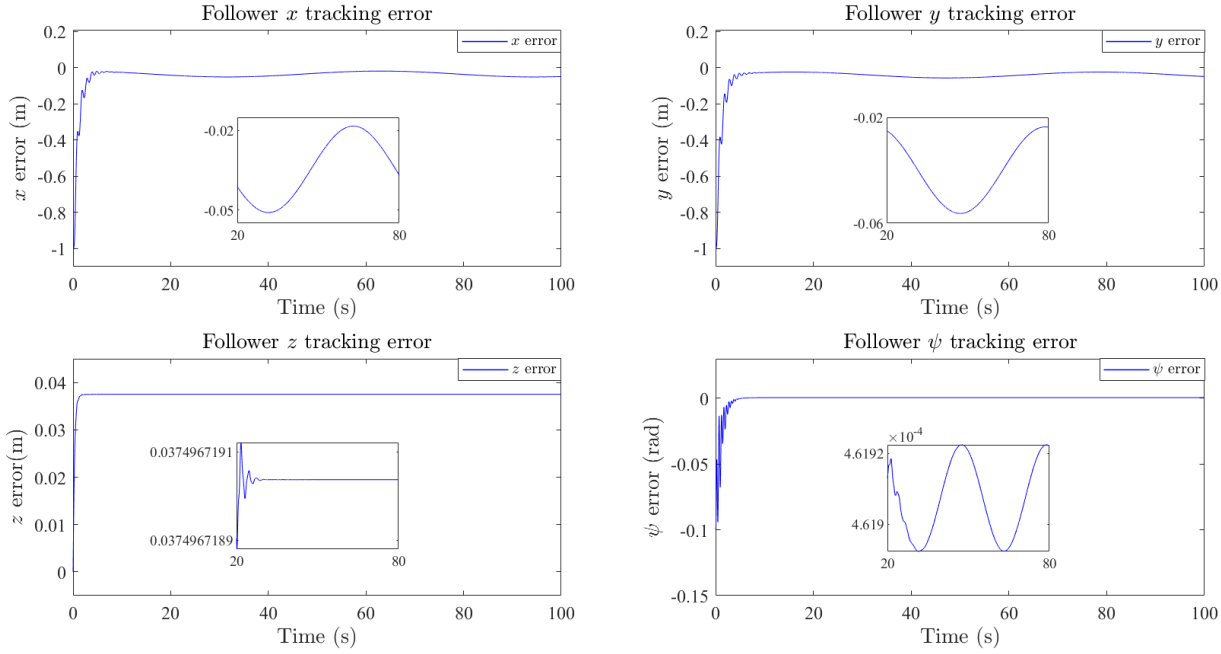

Figure 5.6: Follower PD tracking errors

5.3.2 EKF-based PD controller results

We use EKF in closed-loop with the PD controller for the followers. We used additive white Gaussian noise for process and measurement noise. The weighting matrices to characterize the noise and the initial estimation error covariance are selected through an iterative process of trial and error, they are identical for the two followers $i \in \{F_1, F_2\}$.

$$
\begin{aligned}
Q_{\xi}^{i} &= diag(0.05, 0.02, 0.02, 0.02, 5, 5, 0.02, 0.02, 0.0002, 0.02, 0.02, 0.0002), \\
R_{\upsilon}^{i} &= diag(3.5, 2, 2, 0.000001, 0.000001, 0.00005), \\
P_{0}^{i} &= 0.001 * diag(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1).
\end{aligned} \tag{5.68}
$$

Trajectory tracking results for the first follower:

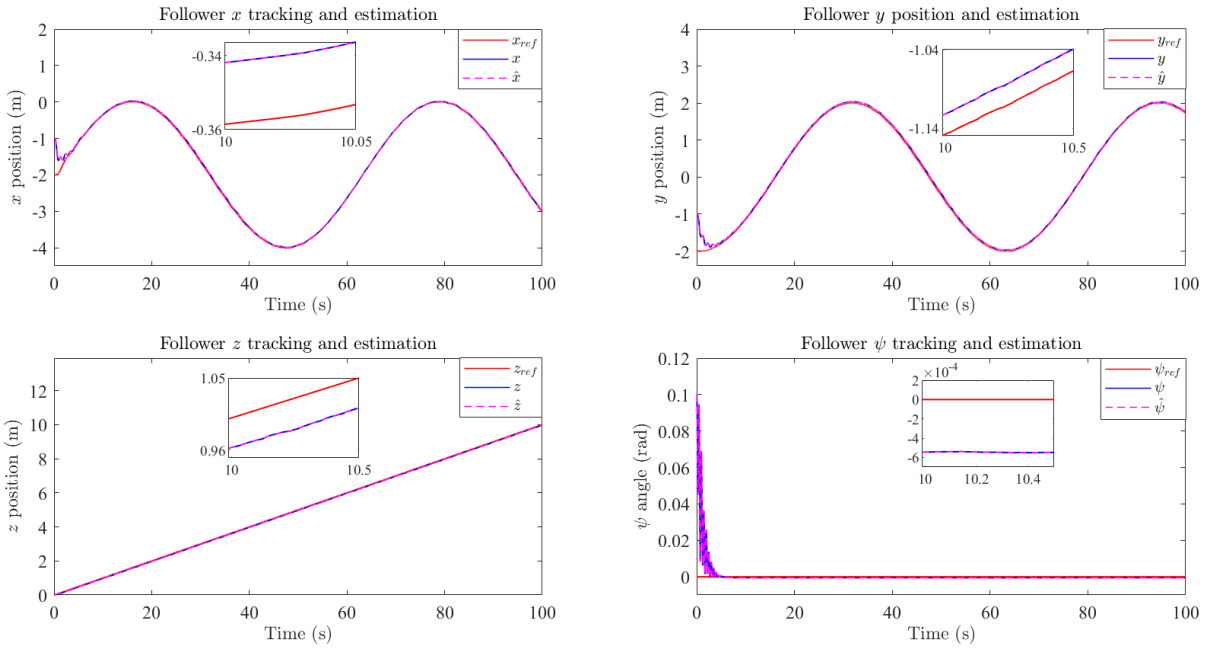

Figure 5.7: Follower EKF-based PD trajectory tracking

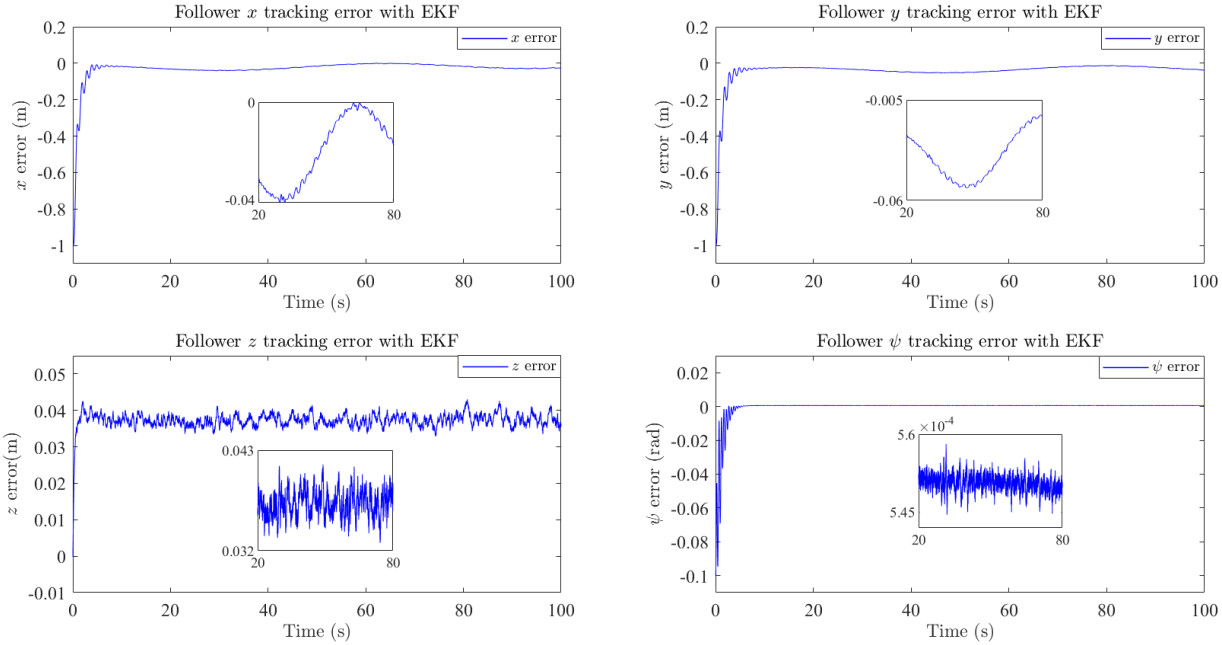

Figure 5.8: Follower EKF-based PD tracking errors

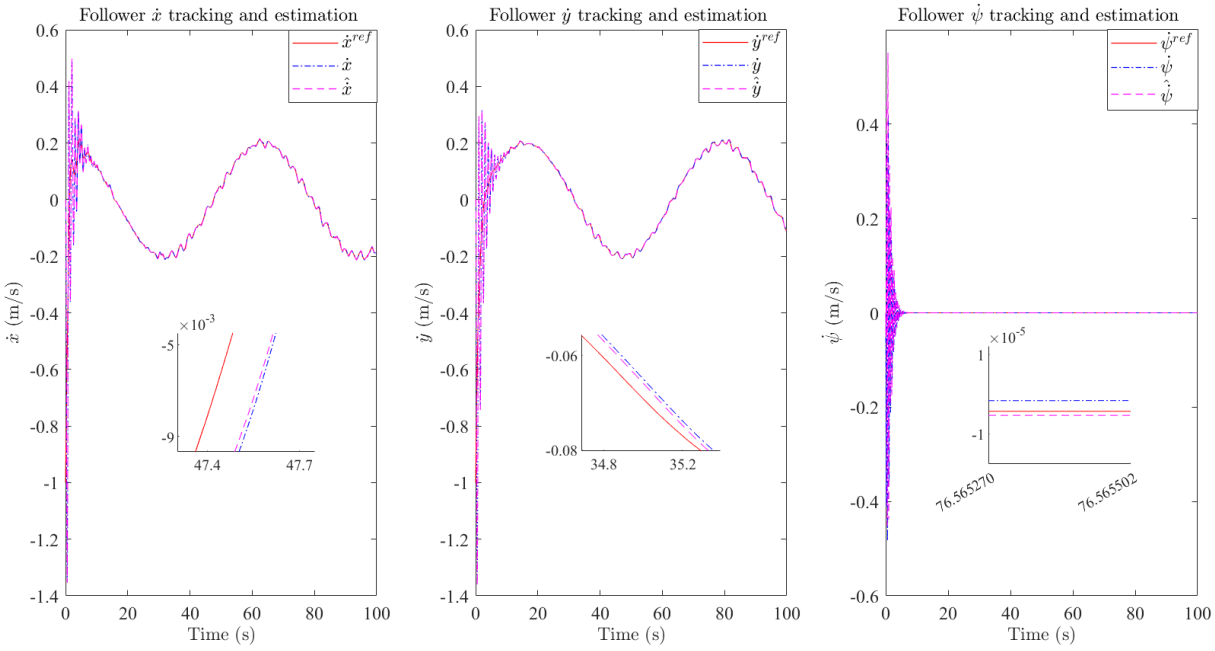

Figure 5.9: Follower EKF-based PD velocity tracking

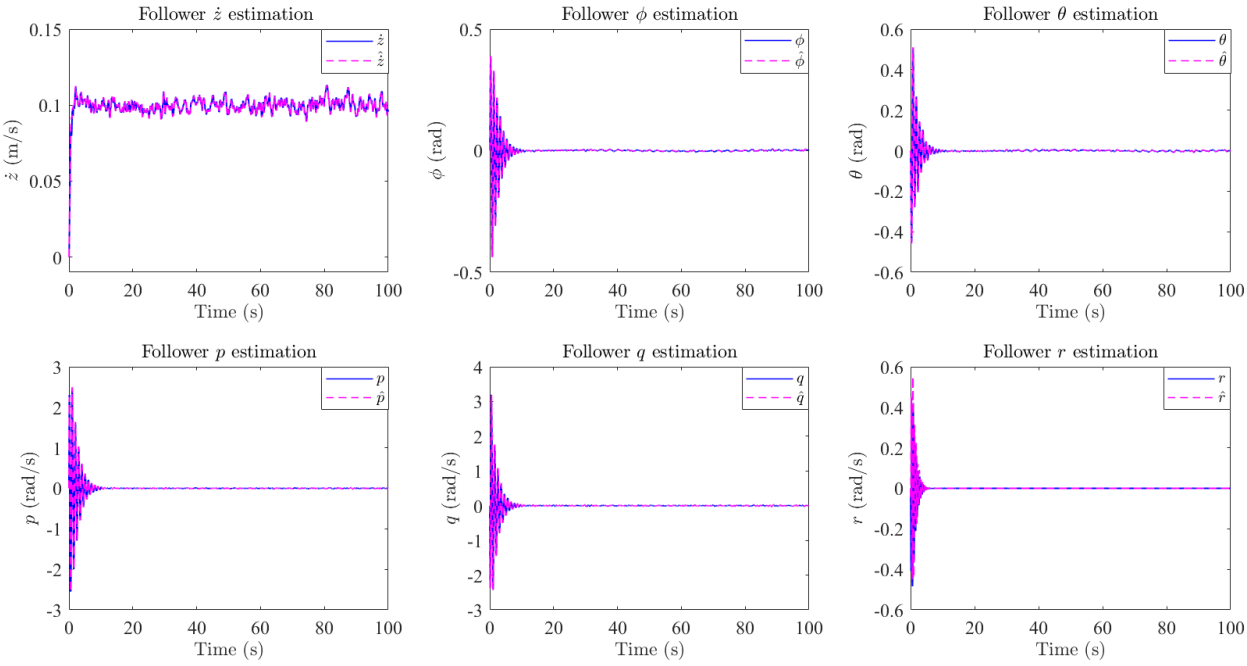

Figure 5.10: Follower's states with EKF-based PD controller

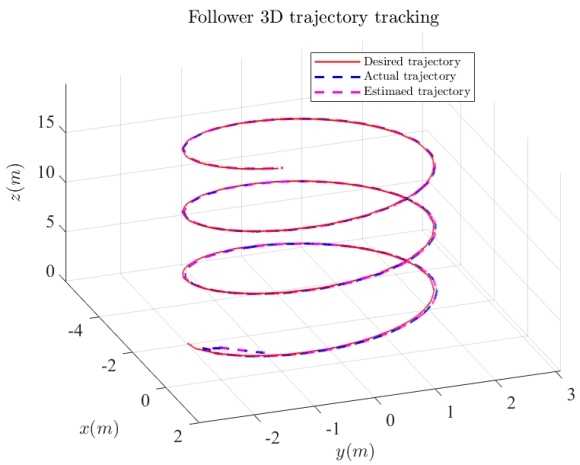

Figure 5.11: Follower EKF-based PD trajectory tracking in 3D space

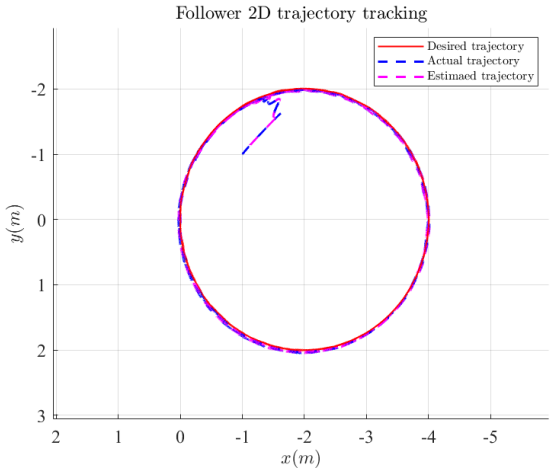

Figure 5.12: Follower EKF-based PD trajectory tracking in the $x-y$ plane

Formation of three quadrotors using the EKF-based PD control:

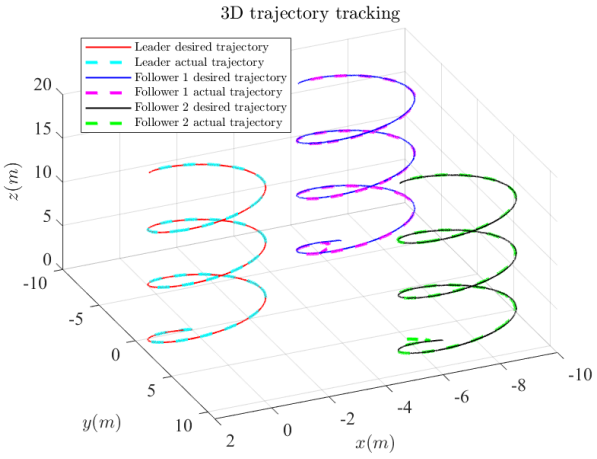

Figure 5.13: Formation of 3 quadrotors in 3D space

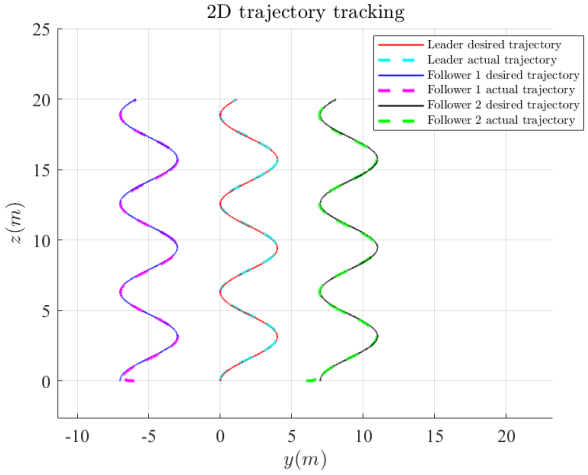

Figure 5.14: Formation of 3 quadrotors in the $y-z$ plane

5.3.3 NMPC controller results

Now we present the follower's results using NMPC. The NMPC parameters are identical for the two followers, they are shown in the table 5.2.

NMPC parameters	Values
Sampling time T	0.05 [s]
Number of intervals N	100
Prediction horizon length	5 [s]
Weighting matrices Q_{F_1}, Q_{F_2}	$diag(1,1,1,1,1,1,1,1,1,1,1,1)$
Weighting matrices R_{F_1}, R_{F_2}	$diag(1,1,1,1)$

Table 5.2: Followers' NMPC simulation parameters

Trajectory tracking results for the first follower:

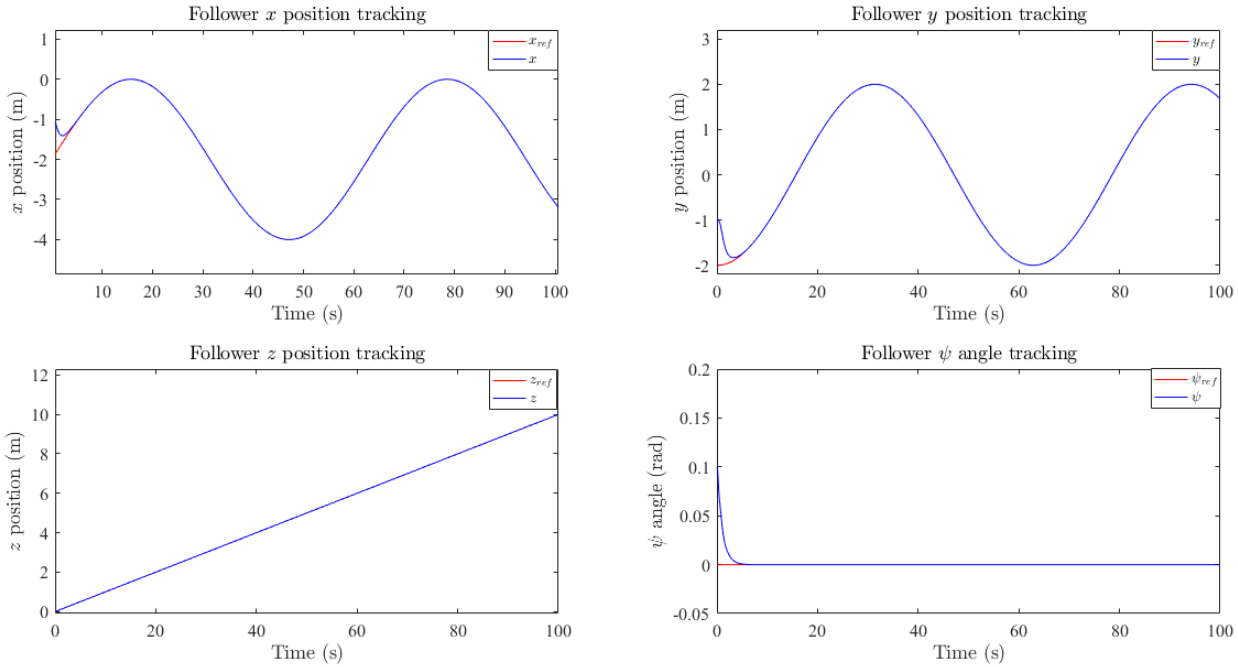

Figure 5.15: Follower NMPC trajectory tracking

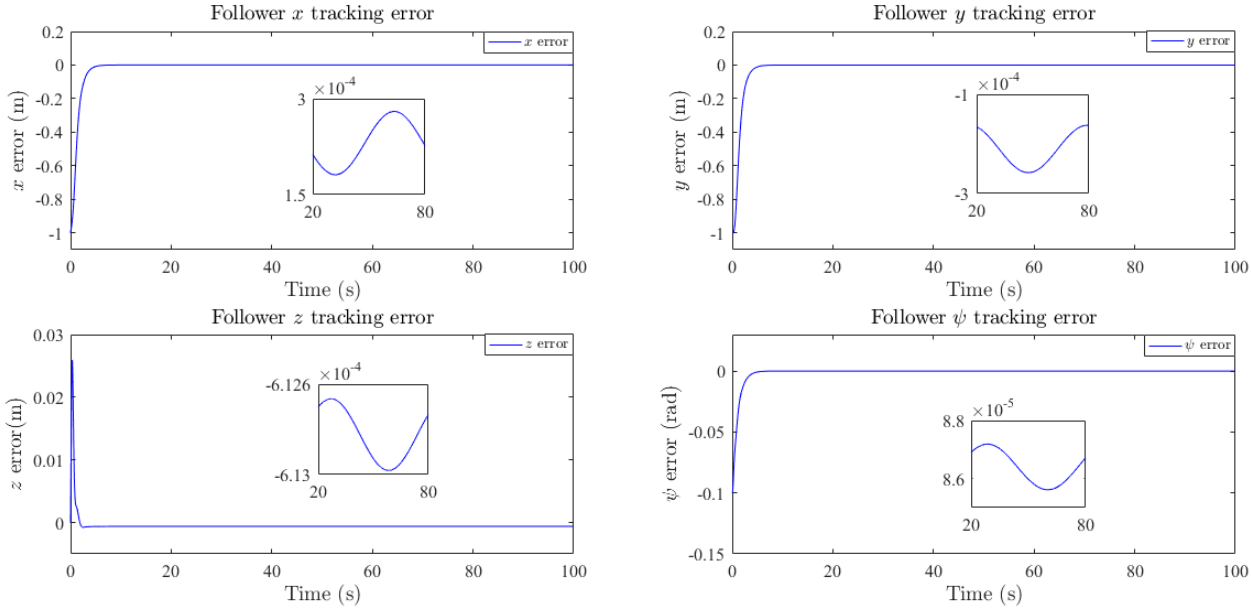

Figure 5.16: Follower NMPC tracking errors

5.3.4 EKF-based NMPC controller results

In this subsection, we use EKF in closed-loop with the NMPC controller. The EKF covariance matrices Q_ξ^i, R_ν^i and P_0^i are the same as for the EKF-based PD.

Trajectory tracking results for the first follower:

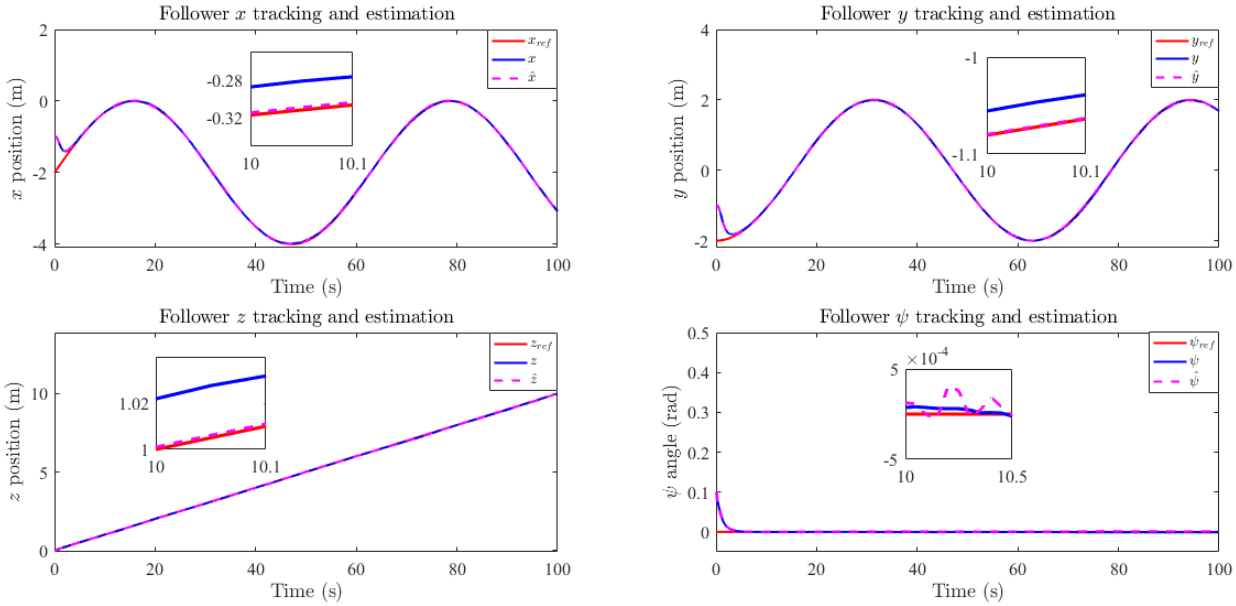

Figure 5.17: Follower EKF-based NMPC trajectory tracking

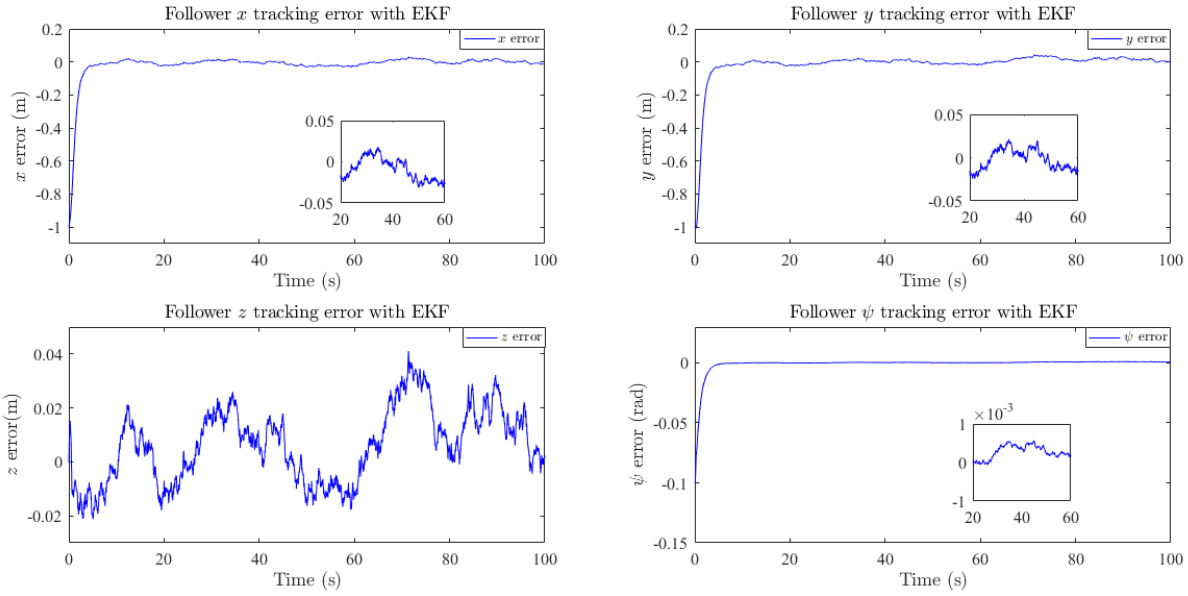

Figure 5.18: Follower EKF-based NMPC tracking errors

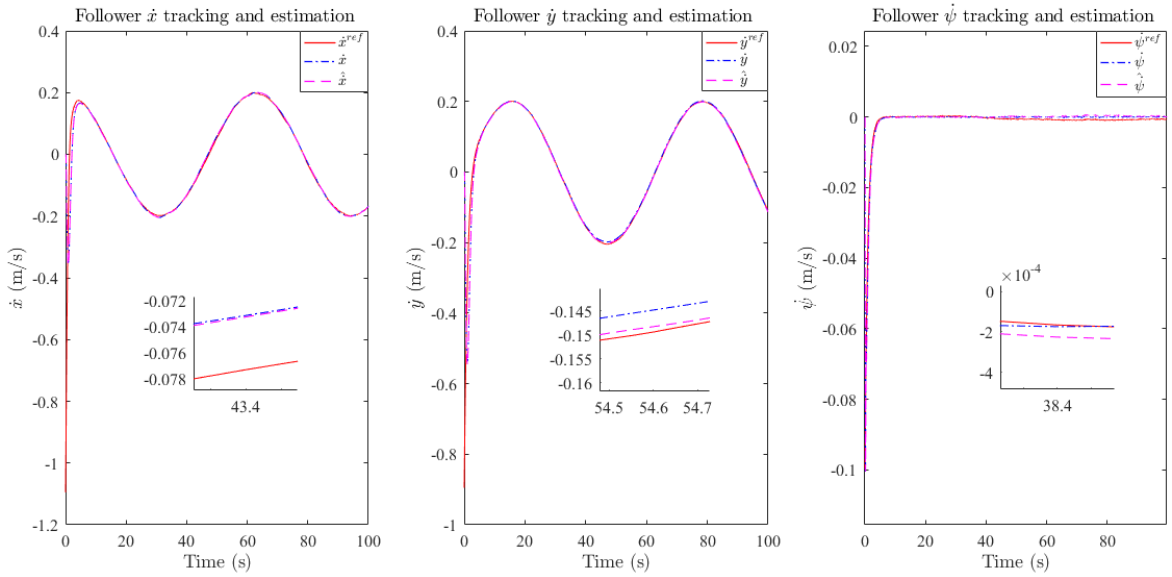

Figure 5.19: Follower EKF-based NMPC velocity tracking

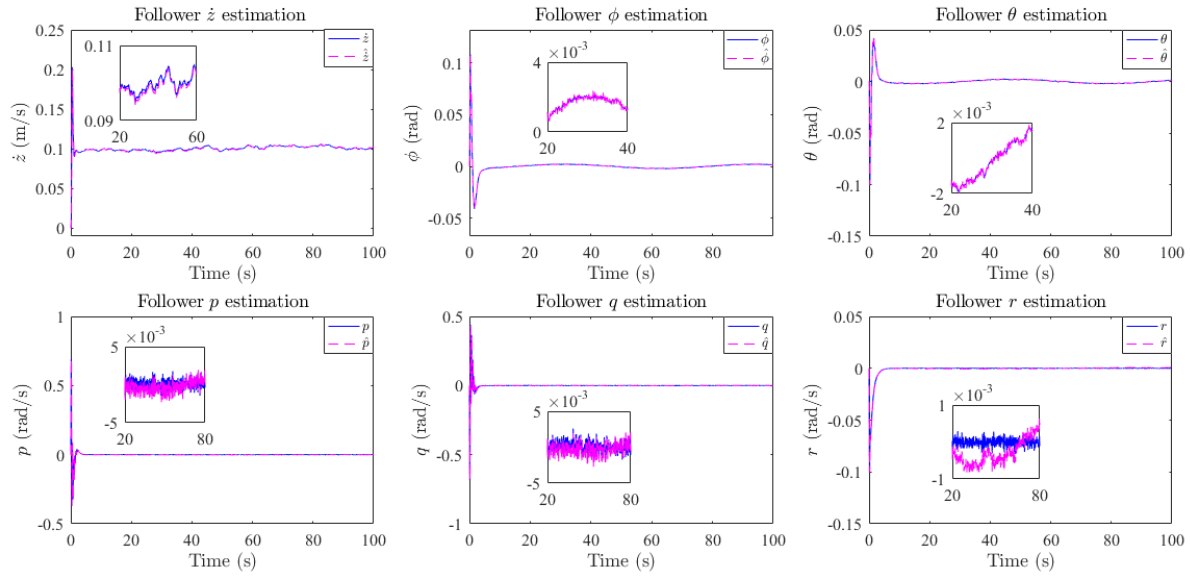

Figure 5.20: Follower's states with EKF-based NMPC controller

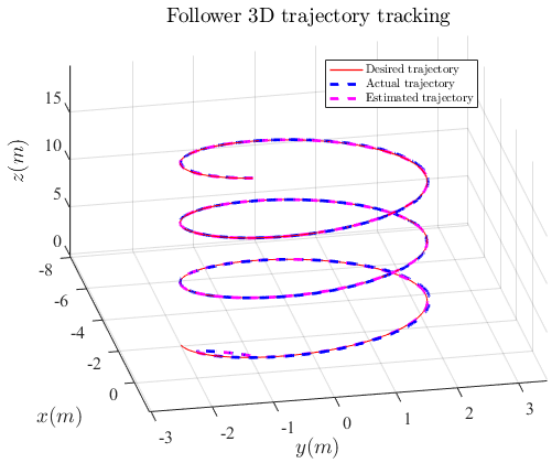

Figure 5.21: Follower EKF-based NMPC trajectory tracking in 3D space

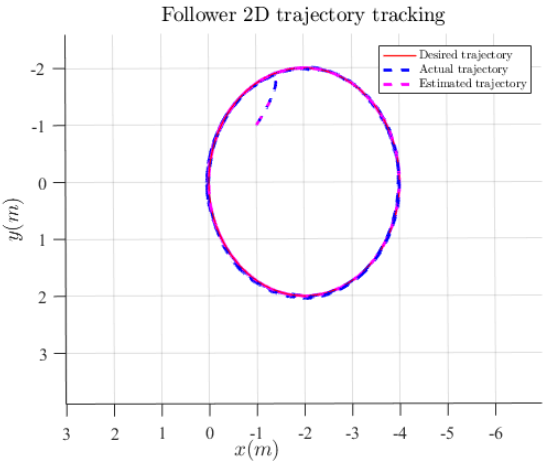

Figure 5.22: Follower EKF-based NMPC trajectory tracking in the $x-y$ plane

Now we present the leader's $\dot{x}_L$ and $\dot{y}_L$ velocities predictions, $\dot{x}_{L,U}$ and $\dot{y}_{L,U}$ respectively, that are used in the formation controller to generate the followers' references, for $N = 100$ and $T = 0.05s$ we get the following predictions over the horizon $[k, k+N]$:

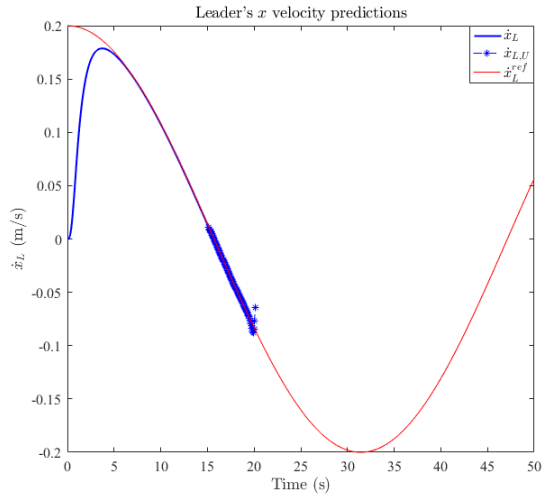

Figure 5.23: Leader's x velocity predictions

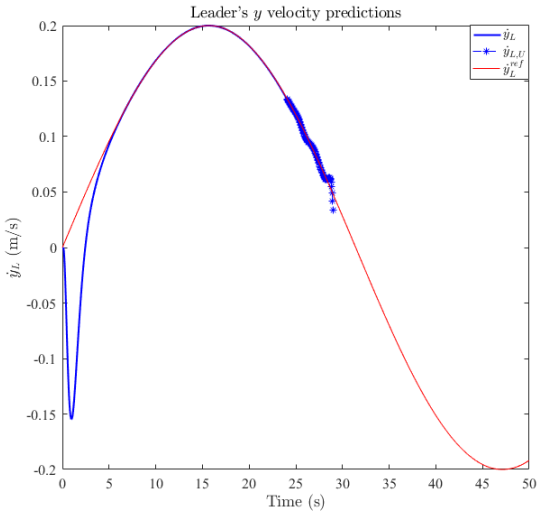

Figure 5.24: Leader's y velocity predictions

Formation of three quadrotors using the EKF-based NMPC control:

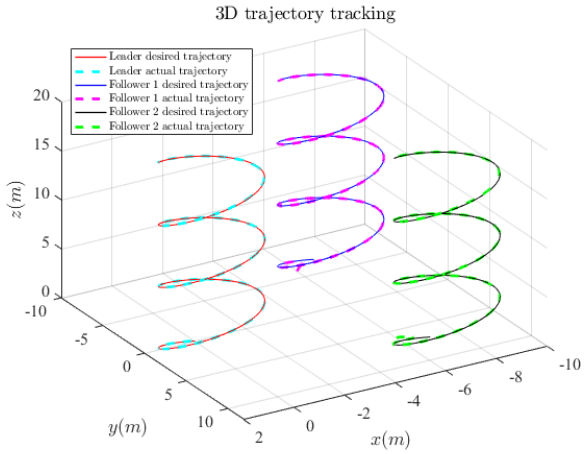

Figure 5.25: Formation of 3 quadrotors in 3D space

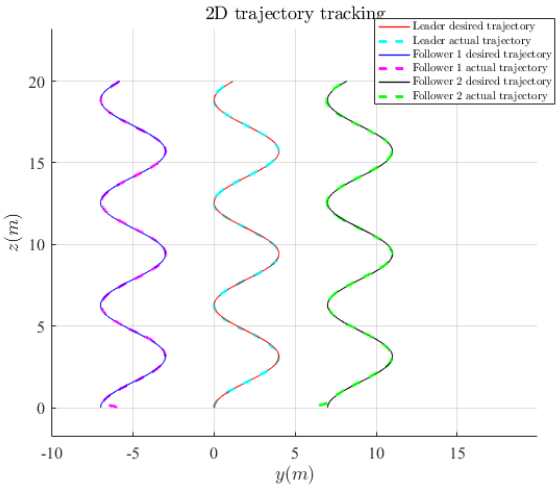

Figure 5.26: Formation of 3 quadrotors in the $y-z$ plane

5.4 Comparison and results discussion

This section provides a comparative analysis of the simulation results for quadrotor formation control, particularly the followers' results in the context of the leader-follower approach. First, we compare the performance of PD and NMPC controllers, then we compare EKF-based PD and NMPC controllers for the followers. The findings aim to guide the selection of the most suitable control approach for quadrotor formation control, enhancing precision and coordination in real-world scenarios.

5.4.1 Full state controllers

The focus of this section is a comparison between PD and NMPC full state controllers. Through an analysis of their performance metrics, we aim to evaluate the effectiveness of each control strategy in achieving precise and coordinated quadrotor formation flight. Table 5.3 provides some information about the two control strategies.

Controller	PD		NMPC	
Evaluation	Error(%)	$T_{r(5\%)}(s)$	Error (%)	$T_{r(5\%)}(s)$
x	1.3126	3.47	0.0108	3.05
y	3.7966	3.36	0.0460	2.85
z	1.8758	4.25	0.0307	0.90
ψ	0.0462	1.06	0.0076	0.80
$\dot{x}$	0.5892	7.27	0.0001	4.30
$\dot{y}$	0.3342	7.82	0.0004	4.90
$\dot{\psi}$	0.0792	2.35	0.0299	1.95

Table 5.3: Performance metrics of full-state controllers for the follower

Result discussion

- The leader-follower formation control approach proved to be highly effective, showcasing its simplicity and intuitive nature. This approach enabled the agents in the system to successfully achieve the desired shape, with the followers relying solely on the leader's information.
- Both the NMPC and PD controllers demonstrated satisfying performance in the leader-follower formation control, ensuring accurate trajectory tracking for the leader, and precise formation keeping for the two followers, maintaining the formation pattern in space.
- The PD controller, despite its simplicity, yielded very satisfying results in the leader-follower formation control. The straightforward nature of the PD control technique allowed for efficient implementation on the three quadrotors in the formation. The Genetic Algorithm also contributed to the followers' gain tuning, enhancing the PD performance in maintaining the desired formation pattern.
- However, when comparing the performance of NMPC and PD controllers, it is evident that NMPC outperforms PD in terms of overall control performance, given that the relative errors and time response for NMPC are much smaller than for the PD. For instance, the PD relative error in the y axis for the followers is close to 3.8% compared to 0.04% for the NMPC.
- The formation controller generates reference velocities for the followers' based on the leader's velocities to maintain the desired distance and deviation from the leader. This approach of synchronizing the followers' velocity with the leader's velocity ensures consistent motion patterns, coordinated movements, and collective behavior to maintain the formation shape and adapt to changes in the leader's velocity.
- The formation controller, based on Lyapunov's method for stability, demonstrated effectiveness by accurately generating the necessary velocities for the followers to maintain the desired formation shape. It played a crucial role in stabilizing the errors between the leader and the followers, ensuring that each follower's velocity aligned with the desired formation. Simulation results show that all quadrotors converge to the desired formation shape.
- Adapting the NMPC controller to the followers' case requires tuning several parameters, such as prediction horizon length and weighting matrices. Additionally, unlike the PD, NMPC also requires having the followers' references at each instant over the whole horizon which poses a supplementary challenge.
- The adoption of a prediction-based approach in the followers' NMPC controller showed impressive results. By incorporating the predicted leader's velocities into the formation controller, the followers received the necessary velocity references over the entire prediction horizon. This allowed enhanced coordination and synchronization between the leader and followers, resulting in improved formation control performance. This prediction-based approach proved to be a valuable solution for providing future state information of the leader, which would otherwise be inaccessible to the followers.

- The computational complexity of the NMPC and PD controllers differs, with NMPC being more complex but yielding better results, while PD offers simplicity with acceptable performance. The NMPC controller's higher computational requirements stem from its iterative optimization and consideration of future time steps. Despite the increased complexity, NMPC demonstrates superior accuracy and precision in leader-follower formation control. In contrast, the PD controller requires fewer computational resources and is easier to implement in real-time scenarios.

- The implementation of NMPC for the leader-follower case in the optimization framework CasADi offered significant advantages. These advantages include the seamless extension of the leader's problem formulation to the followers, the online optimization capabilities, effective handling of complex dynamics and constraints for each quadrotor in the formation, and enhanced computational efficiency.

5.4.2 EKF-based controllers

In this section, we compare the two PD and NMPC controllers with EKF estimation for formation control. Building upon the simulations conducted in previous sections, we analyze their performance metrics to assess their effectiveness in achieving precise and coordinated quadrotor formation flight. Table 5.4 presents results for each control strategy, enabling a comprehensive comparison of their respective performance.

Controller	EKF-based PD		EKF-based NMPC	
Evaluation	Error(%)	$T_{r(5\%)}(s)$	Error (%)	$T_{r(5\%)}(s)$
x	2.0266	4.44	0.8127	2.90
y	3.3955	2.43	1.5965	2.90
z	1.9496	10.01	1.1534	2.95
ψ	0.0558	0.69	0.0431	0.80
$\dot{x}$	3.1830	7.50	0.9380	4.50
$\dot{y}$	1.3883	5.69	1.6883	4.90
$\dot{\psi}$	0.1820	0.74	0.0381	2.05

Table 5.4: Performance metrics of EKF-based controllers for the follower

Result discussion

- The adoption of EKF state estimation exhibited exceptional accuracy in estimating the states of both the leader and followers. This accuracy played a crucial role in the formation maintenance of the three quadrotors.

- For the full state controllers, the performance of the NMPC controller in both position and velocity tracking surpassed that of the PD controller. The absence of noise and EKF estimation allowed the NMPC controller to achieve much smaller errors. When noise and EKF estimation were introduced, both PD and NMPC controllers' relative errors increased. However, the NMPC controller's experienced a more notable increase in errors compared to the PD controller.

- Despite the increased errors, NMPC still outperforms PD control, demonstrating smaller relative errors overall. These results highlight the superiority of the NMPC controller in achieving precise and coordinated quadrotor formation flight.
- However, it is important to acknowledge that NMPC exhibited some sensitivity to the feedback signal, particularly in the presence of noise. This sensitivity suggests that further tuning of the prediction horizon and other parameters could potentially enhance the NMPC controller's performance and mitigate its sensitivity.
- The notable advantage of the NMPC controller lies in its quicker response time $T_{r(5\%)}$ compared to the PD. This characteristic proves valuable for prompt reactions and improved tracking performance in dynamic environments.

In conclusion, the choice between the NMPC and PD controllers for leader-follower formation control depends on several factors. The NMPC controller, despite its higher computational complexity, offers superior accuracy, precision, and response time, making it suitable for applications that prioritize optimal control performance. On the other hand, the PD controller's simplicity and real-time implementation feasibility make it a practical choice when acceptable performance is sufficient. The decision should consider the desired control objectives, available computational resources, and the trade-offs between complexity and real-time requirements. Ultimately, selecting the appropriate controller requires a careful evaluation of the specific application's needs and constraints.

5.5 Conclusion

This chapter presented a leader-follower formation control strategy for multiple quadrotors, an effective formation controller based on Lyapunov method is presented for the formation keeping, and the trajectory tracking problem is adapted to the followers. Furthermore, two control strategies were employed, namely PD and NMPC, combined with EKF for state estimation for the trajectory tracking of the leader and the formation keeping by the followers. A prediction-based control approach has been proposed in the NMPC controller, which solved the significant challenge of generating the followers' references based on the leader's state. The developed control and estimation strategies enabled the quadrotors to achieve coordinated motion and maintain the desired formation geometry. Simulation experiments were conducted to evaluate the effectiveness of each controller.

Chapter 6

Conclusion and future work

6.1 General conclusion

In this work, our primary focus was on multi-UAV systems, particularly emphasizing the problem of formation control in multi-quadrotor systems. Our investigation delves into the leader-follower approach, where one quadrotor assumes the role of the leader while the remaining ones act as followers. The main objective of this study was to gain valuable insights into various aspects of quadrotor systems, including mathematical modeling, control techniques, estimation techniques, and formation control of multi-quadrotor systems.

The trajectory tracking problem of the leader quadrotor has been firstly investigated. This problem was tackled using two control techniques, namely PD and NMPC controllers, combined with an EKF state estimation. First, we implemented a cascade PD control scheme for the single quadrotor case, the fine-tuning of its parameters was achieved by employing the Genetic Algorithm using a multi-objective cost function to minimize the tracking errors and the control effort. Then, we used an NMPC controller by formulating an Optimal Control Problem (OCP), which was discretized using the multiple shooting technique and transformed into a Nonlinear Programming Problem (NLP). The resulting NLP was then solved using the interior-point optimization method using the frame work CasADi which significantly enhanced the computational efficiency. This NMPC controller allowed the solving of an online optimization problem over a finite receding horizon at each time step, as well as the handling of complex nonlinear constraints. The two controllers were implemented in the full-state case and the estimation-based case.

Then, the rigid formation task of a multi-quadrotor system was addressed, where the followers have to keep a fixed inter-distance and orientation from the leader, resulting in a 'V' shape. A Lyapunov based formation controller was used to ensure the formation keeping by the followers. Moreover, the previously developed PD and NMPC controllers were adapted for the formation keeping problem of the followers. For the NMPC controller, our contribution lies in the introduction of a prediction-based control approach, where the leader's predictions were used to generate references for the followers. Numerical results were presented demonstrating the effectiveness of the proposed control and estimation approaches to keep the formation pattern and ensure coordinated motion of the three quadrotors.

A comparison between the two PD and NMPC controllers has been conducted in the context of a single quadrotor and the leader-follower approach. It aimed to asses their performance and reveal insights into the strengths and limitations of both control methods, shedding light on their respective merits and trade-offs in terms of tracking accuracy, response time, and computational complexity. The PD control method demonstrated efficiency and simplicity in its implementation. In contrast, the NMPC approach exhibited superior performance in trajectory tracking compared to PD. However, it should be noted that the NMPC approach is computationally demanding. These findings serve as a valuable resource in selecting the appropriate control strategy based on specific application requirements and desired performance metrics.

6.2 Future work

As the field of formation control of multi-quadrotor systems continues to evolve rapidly, there are several avenues for future research and development that can further enhance the capabilities and applications of these systems. Building upon the findings and contributions of this work, the following areas warrant further exploration:

- Real-time implementation of the developed control and estimation strategies for a multi-UAV system, necessitating further exploration into hardware and communication protocols to ensure efficient control in dynamic environments.
- Implementing EKF estimation with measurement of Euler rates: an alternative approach can be explored by measuring Euler rates using a gyroscope, which may be a cost-effective alternative to a full-fledged AHRS system.
- Modeling the quadrotor motors: in order to accurately simulate the behavior of the quadrotor, it is important to model the motors that drive the rotors. This will enhance the overall accuracy of the quadrotor model and enable more realistic simulations and control system design.
- Robustness testing: conducting tests to assess the performance and stability of the developed formation control strategies under various disturbances and uncertainties in real-world scenarios.
- Variable distance between the leader and the followers: experimenting with different distances between the leader and followers drones to understand the system's dynamics and communication requirements in practical environments.
- Variable orientation in the formation: investigating the effects of a variable orientation between the leader and the followers in the formation, enabling the drones to adapt to different orientations and facilitate more diverse formation geometries.
- Formation in 3D space: extending the developed formation controller in the $x - y$ plane, where the different quadrotors have to be at an equal altitude z at each instant, to a 3D formation controller were the agents can navigate at different heights.
- Scaling up the formation: extending the formation control framework to accommodate a larger number of drones and hierarchical stages of followers, enabling the formation to handle complex missions in real-world applications.
- Obstacle avoidance: developing and integrating robust obstacle avoidance algorithms to enhance the autonomy and safety of the drones within the formation when operating in dynamic environments.
- Interchangeable leaders: implementing the concept of interchangeable leaders, allowing different drones to take turns as the leader during the mission, providing flexibility and adaptability in real-world scenarios.
- Improving computational efficiency of the NMPC controller: optimizing the computational efficiency of the control strategies and algorithms to ensure real-time implementation on resource-constrained platforms, making the formation control feasible for real-world applications.

Appendixes

CasADi

CasADi is an open-source software tool for numerical optimization in general and optimal control (i.e. optimization involving differential equations) in particular. CasADi is designed to be used in scientific and engineering applications that involve optimization, simulation, and control. It supports a wide range of problem types, including nonlinear programming (NLP), optimal control, and dynamic optimization.

CasADi started out as a tool for algorithmic differentiation (AD) using a syntax borrowed from computer algebra systems (CAS), which explains its name. While AD still forms one of the core functionalities of the tool, the scope of the tool has since been considerably broadened, with the addition of support for ODE/DAE integration, nonlinear programming and interfaces to other numerical tools. In its current form, it is a general-purpose tool for gradient-based numerical optimization – with a strong focus on optimal control – and CasADi is just a name without any particular meaning.

The framework offers a user-friendly interface for defining optimization problems and provides a suite of efficient algorithms for solving them. It supports various solvers, both local and global, to handle different types of optimization problems. CasADi also incorporates advanced features like sensitivity analysis, multi-threading, and parallel computing to enhance performance. One of the key features of CasADi is its automatic differentiation capabilities. It allows users to obtain derivatives of functions defined in CasADi with respect to their inputs or parameters. This feature is particularly useful in optimization and control problems, where gradient information is required for efficient solution methods such as gradient-based optimization algorithms.

CasADi is implemented in C++, but it provides interfaces for several programming languages, including MATLAB, Python, and Julia. This makes it accessible to a broad user community working in diverse scientific and engineering domains. It is important to point out that CasADi is not a conventional AD tool, that can be used to calculate derivative information from existing user code with little to no modification. If you have an existing model written in C++, Python or MATLAB/Octave, you need to be prepared to re-implement the model using CasADi syntax. Secondly, CasADi is not a computer algebra system. While the symbolic core does include an increasing set of tools for manipulating symbolic expressions, these capabilities are very limited compared to a proper CAS tool. Finally, CasADi is not an "optimal control problem solver", that allows the user to enter an OCP and then gives the solution back. Instead, it tries to provide the user with a set of "building blocks" that can be used to implement general-purpose or specific-purpose OCP solvers efficiently with a modest programming effort.

Bibliography

[1] R. Shakeri, M. Al-Garadi, A. Badawy, *et al.*, "Design challenges of multi-uav systems in cyber-physical applications: A comprehensive survey, and future directions," *IEEE Communications Surveys Tutorials*, vol. 21, pp. 3340–3385, Jun. 2019.

[2] L. Newcome, *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles* (EngineeringPro collection). American Institute of Aeronautics and Astronautics, 2004, ISBN: 9781563476440.

[3] P. Howlett, "An optimal strategy for the control of a train," *The ANZIAM Journal*, vol. 31, no. 4, pp. 454–471, 1990.

[4] P. Peter and H. Phil, "Optimal driving strategies for a train journey with speed limits," *The ANZIAM Journal*, vol. 36, no. 1, pp. 38–49, 1994.

[5] T. Balch and R. Arkin, "Behavior-based formation control for multi-robot teams," *Robotics and Automation, IEEE Transactions on*, vol. 14, pp. 926–939, Jan. 1999.

[6] J. Wang, X. Nian, and H.-b. Wang, "Consensus and formation control of discrete-time multi-agent systems," *Journal of Central South University of Technology (English Edition)*, vol. 18, pp. 1161–1168, Aug. 2011.

[7] Z. Hou, "Modeling and formation controller design for multi-quadrotor systems with leader-follower configuration," Theses, Université de Technologie de Compiègne, Feb. 2016.

[8] J. García, J. M. Molina, and J. Trincado, "Real evaluation for designing sensor fusion in uav platforms," *Information Fusion*, vol. 63, pp. 136–152, 2020, ISSN: 1566-2535.

[9] Y. Liu, R. Yu, S. Cai, and H. Mu, "Cooperative localization of imu-based uav using relative observation by ekf," in *Proceedings of 2021 International Conference on Autonomous Unmanned Systems (ICAUS 2021)*, M. Wu, Y. Niu, M. Gu, and J. Cheng, Eds., Singapore: Springer Singapore, 2022, pp. 311–321, ISBN: 978-981-16-9492-9.

[10] C. Camacho E. F. Bordons, *Model Predictive Control*. Springer London, Jun. 2004.

[11] J. P. Lars Grüne, *Nonlinear Model Predictive Control*. Springer London, Apr. 2011.

[12] N. Michael, S. Shen, K. Mohta, *et al.*, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, Sep. 2012, ISSN: 1556-4959.

[13] G. Loianno, J. Thomas, and V. Kumar, "Cooperative localization and mapping of mavs using rgb-d sensors," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4021–4028.

[14] J. Zhang, J. Yan, and P. Zhang, "Fixed-wing uav formation control design with collision avoidance based on an improved artificial potential field," *IEEE Access*, vol. 6, no. 1, pp. 78 342–78 351, Dec. 2018.

[15] W. Y. et al., "Obstacle avoidance research of the automated guided vehicle based on improved artificial potential field method with chaotic optimization," *Sci. Technol. Innov. Herald*, vol. 14, no. 17, pp. 150–153, 2017.

[16] Z. Ali, A. Israr, E. Alkhammash, and M. Hadjouni, "A leader-follower formation control of multi-uavs via an adaptive hybrid controller," *Complexity*, vol. 2021, p. 16, Nov. 2021.

[17] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

[18] G. Tuta Navajas and S. Roa Prada, "Building your own quadrotor: A mechatronics system design case study," in *2014 III International Congress of Engineering Mechatronics and Automation (CIIMA)*, 2014, pp. 1–5.

[19] M. Tanveer, S. F. Ahmed, H. Desa, F. Warsi, and M. Joyo, "Stabilized controller design for attitude and altitude controlling of quad-rotor under disturbance and noisy conditions," *American Journal of Applied Sciences*, vol. 10, pp. 819–831, Aug. 2013.

[20] M. Fatan, B. L. Sefidgari, and A. V. Barenji, "An adaptive neuro pid for controlling the altitude of quadcopter robot," in *2013 18th International Conference on Methods Models in Automation Robotics (MMAR)*, 2013, pp. 662–665.

[21] N. Cao and A. F. Lynch, "Inner–outer loop control for quadrotor uavs with input and state constraints," *IEEE Transactions on Control Systems Technology*, vol. 24, pp. 1797–1804, 2016.

[22] A. Tayebi and S. McGilvray, "Attitude stabilization of a vtol quadrotor aircraft," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 3, pp. 562–571, 2006.

[23] J. Han, L. Di, C. Coopmans, and Y. Chen, "Fractional order controller for pitch loop control of a vtol uav," *Journal of Intelligent Robotic Systems*, vol. 73, May 2013.

[24] S. Seyedtabaii, "New flat phase margin fractional order pid design: Perturbed uav roll control study," *Robotics and Autonomous Systems*, vol. 96, pp. 58–64, 2017, ISSN: 0921-8890.

[25] H. Noshahri and H. Kharrati, "Pid controller design for unmanned aerial vehicle using genetic algorithm," in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, 2014, pp. 213–217.

[26] T. Mac, C. Copot, T. Duc, and R. Keyser, "Ar.drone uav control parameters tuning based on particle swarm optimization algorithm," May 2016, pp. 1–6.

[27] H. J. K. D. Lee and S. Sastry, "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter," *Int. J. Control Autom. Syst.*, vol. 7, pp. 419–428, 2009.

[28] P. Adigbli, C. Grand, J.-B. Mouret, and S. Doncieux, "Nonlinear attitude and position control of a micro quadrotor using sliding mode and backstepping techniques," pp. 17–21, Oct. 2007.

[29] R. Xu and Ü. Özgüner, "Sliding mode control of a quadrotor helicopter," *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 4957–4962, 2006.

[30] L. Besnard, Y. Shtessel, and D. Landrum, "Control of a quadrotor vehicle using sliding mode disturbance observer," Aug. 2007, pp. 5230–5235.

[31] Z. G. Xiong JJ, "Global fast dynamic terminal sliding mode control for a quadrotor uav," *ISA Trans*, vol. 66, pp. 233–240, 2017.

[32] D. Cabecinhas, R. Cunha, and C. Silvestre, "A globally stabilizing path following controller for rotorcraft with wind disturbance rejection," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 708–714, 2015.

[33] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 2247–2252.

[34] T. Madani and A. Benallegue, "Backstepping sliding mode control applied to a miniature quadrotor flying robot," in *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, 2006, pp. 700–705.

[35] E. Kayacan and R. Maslim, "Type-2 fuzzy logic trajectory tracking control of quadrotor vtol aircraft with elliptic membership functions," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 339–348, 2017.

[36] J.-J. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, New Jersey., 1991.

[37] H. Voos, "Nonlinear control of a quadrotor micro-uav using feedback-linearization," May 2009, pp. 1–6.

[38] M. A. Lotufo, L. Colangelo, and C. Novara, "Control design for uav quadrotors via embedded model control," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 5, pp. 1741–1756, 2020.

[39] "Robust deep reinforcement learning for quadcopter control," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 90–95, 2021, Modeling, Estimation and Control Conference MECC 2021, ISSN: 2405-8963.

[40] N. Koksal, B. Fidan, and K. Buyukkabasakal, "Real-time implementation of decentralized adaptive formation control on multi-quadrotor systems," Jul. 2015, pp. 3162–3167.

[41] M. Chen and M. Huzmezan, "A combined mbpc/2 dof h infinity controller for a quad rotor uav," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*.

[42] T. Báča, G. Loianno, and M. Saska, "Embedded model predictive control of unmanned micro aerial vehicles," Aug. 2016.

[43] M. Islam and M. Okasha, "A comparative study of pd, lqr and mpc on quadrotor using quaternion approach," in *2019 7th International Conference on Mechatronics Engineering (ICOM)*, 2019, pp. 1–6.

[44] C. Liu, H. Lu, and W.-H. Chen, "An explicit mpc for quadrotor trajectory tracking," in *2015 34th Chinese Control Conference (CCC)*, 2015, pp. 4055–4060.

[45] Tajeddin, Sadegh, "Automatic code generation of real-time nonlinear model predictive control for plug-in hybrid electric vehicle intelligent cruise controllers," M.S. thesis, 2016.

[46] R. C. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems Magazine*, vol. 34, no. 1, pp. 28–41, 2014.

[47] "A cascaded approach for quadrotor's attitude estimation," *Procedia Technology*, vol. 15, pp. 268–277, 2014, 2nd International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering, ISSN: 2212-0173.

[48] A. D. Wu, E. N. Johnson, and A. A. Proctor, "Vision-aided inertial navigation for flight control," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 9, pp. 348–360, 2005.

[49] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor control: Attitude, altitude and position experimental studies," *Iet Control Theory and Applications*, vol. 6, pp. 1812–1827, 2012.

[50] "Nonlinear kalman filters and particle filters for integrated navigation of unmanned aerial vehicles," *Robotics and Autonomous Systems*, vol. 60, no. 7, pp. 978–995, 2012, ISSN: 0921-8890.

[51] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative grasping and transport using multiple quadrotors," in *Distributed Autonomous Robotic Systems: The 10th International Symposium*, A. Martinoli, F. Mondada, N. Correll, *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 545–558.

[52] A.-C. Stan, "A decentralised control method for unknown environment exploration using turtlebot 3 multi-robot system," in *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2022, pp. 1–6.

[53] A. Chriki, H. Touati, H. Snoussi, and F. Kamoun, "Uav-gcs centralized data-oriented communication architecture for crowd surveillance applications," in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2019, pp. 2064–2069.

[54] F. Aljalaud, H. Kurdi, and K. Youcef-Toumi, "Autonomous multi-uav path planning in pipe inspection missions based on booby behavior," *Mathematics*, vol. 11, no. 9, 2023, ISSN: 2227-7390.

[55] H. Shen, Y. Jiang, F. Deng, and Y. Shan, "Task unloading strategy of multi uav for transmission line inspection based on deep reinforcement learning," *Electronics*, vol. 11, no. 14, 2022, ISSN: 2079-9292.

[56] R. H. Kabir and K. Lee, "Wildlife monitoring using a multi-uav system with optimal transport theory," *Applied Sciences*, vol. 11, no. 9, 2021, ISSN: 2076-3417.

[57] L. Hogie, P. Bouvry, and F. Guinand, "An overview of manets simulation," *Electronic Notes in Theoretical Computer Science*, vol. 150, no. 1, pp. 81–101, 2006, Proceedings of the First International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2005), ISSN: 1571-0661.

[58] S. Al-Emadi and A. Al-Mohannadi, "Towards enhancement of network communication architectures and routing protocols for fanets: A survey," in *2020 3rd International Conference on Advanced Communication Technologies and Networking (CommNet)*, 2020, pp. 1–10.

[59] M. R. Ghori, A. S. Sadiq, and A. Ghani, "Vanet routing protocols: Review, implementation and analysis," *Journal of Physics: Conference Series*, vol. 1049, no. 1, p. 012 064, Jul. 2018.

[60] G. Skorobogatov, C. Barrado, and E. Salamí, "Multiple uav systems: A survey," *Unmanned Systems*, vol. 08, Nov. 2019.

[61] I. Maza, A. Ollero, E. Casado, and D. Scarlatti, "Classification of multi-uav architectures," *Handbook of unmanned aerial vehicles*, pp. 953–975, 2015.

[62] W. Ni and D. Cheng, "Leader-following consensus of multi-agent systems under fixed and switching topologies," *Systems Control Letters*, vol. 59, no. 3, pp. 209–217, 2010, ISSN: 0167-6911.

[63] Z. Hou and I. Fantoni, "Leader-follower formation saturated control for multiple quadrotors with switching topology," *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pp. 8–14, 2015.

[64] Z. Hou and I. Fantoni, "Distributed leader-follower formation control for multiple quadrotors with weighted topology," *2015 10th System of Systems Engineering Conference, SoSE 2015*, May 2015.

[65] E. Semsar-Kazerooni and K. Khorasani, "Optimal consensus algorithms for cooperative team of agents subject to partial information," *Automatica*, vol. 44, no. 11, pp. 2766–2777, 2008, ISSN: 0005-1098.

[66] E. Semsar-Kazerooni and K. Khorasani, "Switching control of a modified leader-follower team of agents under the leader and network topological changes," *Control Theory Applications, IET*, vol. 5, pp. 1369–1377, Aug. 2011.

[67] A. Chriki, H. Touati, H. Snoussi, and F. Kamoun, "Uav-gcs centralized data-oriented communication architecture for crowd surveillance applications," in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2019, pp. 2064–2069.

[68] W. Y. H. Adoni, S. Lorenz, J. S. Fareedh, R. Gloaguen, and M. Bussmann, "Investigation of autonomous multi-uav systems for target detection in distributed environment: Current developments and open challenges," *Drones*, vol. 7, no. 4, 2023, ISSN: 2504-446X.

[69] R. Abbas and Q. Wu, "Tracking formation control for multiple quadrotors based on fuzzy logic controller and least square oriented by genetic algorithm," *The Open Automation and Control Systems Journal*, vol. 7, pp. 842–850, Aug. 2015.

[70] N. Linorman and H. Liu, "Formation uav flight control using virtual structure and motion synchronization," Jul. 2008, pp. 1782–1787.

[71] C. Hua, J. Chen, and Y. Li, "Leader-follower finite-time formation control of multiple quadrotors with prescribed performance," *International Journal of Systems Science*, vol. 48, pp. 1–10, May 2017.

[72] W. N. Wang, "Research on formation reconfiguration and formation keeping control algorithm for three dimensional unmanned aerial vehicles," *Shen Yang Aerosp. Univ*, 2018.

[73] Y. H. Q. et al, "March-inspired multi-robot compact formation strategy," *CAAI Trans. Intell. Syst*, no. 5, pp. 673–679, 2018.

[74] D. M. Stipanović, G. İnalhan, R. Teo, and C. J. Tomlin, "Decentralized overlapping control of a formation of unmanned aerial vehicles," *Automatica*, vol. 40, no. 8, pp. 1285–1296, 2004, ISSN: 0005-1098.

[75] A. Bemporad and C. Rocchi, "Decentralized hybrid model predictive control of a formation of unmanned aerial vehicles," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 900–11 906, 2011, 18th IFAC World Congress, ISSN: 1474-6670.

[76] Y. Fei, P. Shi, and C.-C. Lim, "Neural network adaptive dynamic sliding mode formation control of multi-agent systems," *International Journal of Systems Science*, vol. 51, no. 11, pp. 2025–2040, 2020.

[77] Z. Sui, Z. Pu, J. Yi, and S. Wu, "Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–15, Jul. 2020.

[78] 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," *IET Control Theory & Applications*, vol. 14, no. 12, pp. 1628–1638, 2020.

[79] T. Z. Muslimov and R. A. Munasypov, "Adaptive decentralized flocking control of multi-uav circular formations based on vector fields and backstepping," *ISA Transactions*, vol. 107, pp. 143–159, 2020, ISSN: 0019-0578.

[80] L. Wei, M. Chen, and T. Li, "Dynamic event-triggered cooperative formation control for uavs subject to time-varying disturbances," *IET Control Theory Applications*, vol. 14, pp. 2514–2525, Nov. 2020.

[81] A. Sheta, M. Braik, D. R. Maddi, A. Mahdy, S. Aljahdali, and H. Turabieh, "Optimization of pid controller to stabilize quadcopter movements using meta-heuristic search algorithms," *Applied Sciences*, vol. 11, no. 14, 2021, ISSN: 2076-3417.

[82] M. Walid, N. Slaheddine, A. Mohamed, and B. Lamjed, "Modeling and control of a quadrotor uav," in *2014 15th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2014, pp. 343–348.

[83] J. Cayero, J. Cugueró, and B. Morcego, "Impedance control of a planar quadrotor with an extended kalman filter external forces estimator," 2015.

[84] M. Rinaldi, S. Primatesta, and G. Guglieri, "A comparative study for control of quadrotor uavs," *Applied Sciences*, vol. 13, no. 6, 2023, ISSN: 2076-3417.

[85] Mardlijah and D. Prihatini, "Control design of quadcopter using output feedback control pole placement," in *2022 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, 2022, pp. 197–202.

[86] M. Sarim, A. Nemati, M. Kumar, and K. Cohen, "Extended kalman filter based quadrotor state estimation based on asynchronous multisensor data," Oct. 2015.

[87] M. Islam, M. Okasha, and M. M. Idres, "Trajectory tracking in quadrotor platform by using pd controller and lqr control approach," *IOP Conference Series: Materials Science and Engineering*, vol. 260, no. 1, p. 012 026, Nov. 2017.

[88] D. C. Meena and A. Devanshu, "Genetic algorithm tuned pid controller for process control," in *2017 International Conference on Inventive Systems and Control (ICISC)*, 2017, pp. 1–6.

[89] P. M. Meshram and R. G. Kanojiya, "Tuning of pid controller using ziegler-nichols method for speed control of dc motor," in *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, 2012, pp. 117–122.

[90] Y. Chen, N. Scarabottolo, M. Bruschetta, and A. Beghi, "Efficient move blocking strategy for multiple shooting based nonlinear model predictive control," *IET Control Theory and Applications*, vol. 14, pp. 343–351, Jan. 2020.

[91] J. Aburajabaltamimi, "Development of efficient algorithms for model predictive control of fast systems," 2011.

[92] J. Tamimi and P. Li, "Nonlinear model predictive control using multiple shooting combined with collocation on finite elements," *IFAC Proceedings Volumes*, vol. 42, no. 11, pp. 703–708, 2009, 7th IFAC Symposium on Advanced Control of Chemical Processes, ISSN: 1474-6670.

[93] D. Simon, "Model predictive control in flight control design - stability and reference tracking," Ph.D. dissertation, Mar. 2014.

[94] M. Elhesasy, T. N. Dief, M. Atallah, *et al.*, "Non-linear model predictive control using casadi package for trajectory tracking of quadrotor," *Energies*, vol. 16, no. 5, 2023, ISSN: 1996-1073.

[95] R. Sanz, P. García, Q.-C. Zhong, and P. Albertos, "Robust control of quadrotors based on an uncertainty and disturbance estimator," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, Apr. 2016.

[96] B. J. Emran, M. Al-Omari, M. F. Abdel-Hafez, and M. A. Jaradat, "A cascaded approach for quadrotor's attitude estimation," *Procedia Technology*, vol. 15, pp. 268–277, 2014.

[97] S. Tellex, A. Brown, and S. Lupashin, "Estimation for quadrotors," *arXiv preprint arXiv:1809.00037*, 2018.

[98] M. Sarim, A. Nemati, M. Kumar, and K. Cohen, "Extended kalman filter based quadrotor state estimation based on asynchronous multisensor data," Oct. 2015.

[99] S. Ahmed, B. Qiu, C.-W. Kong, H. Xin, F. Ahmad, and J. Lin, "A data-driven dynamic obstacle avoidance method for liquid-carrying plant protection uavs," *Agronomy*, vol. 12, no. 4, 2022, ISSN: 2073-4395.

[100] N. Xuan-Mung and S. K. Hong, "Robust adaptive formation control of quadcopters based on a leader–follower approach," *International Journal of Advanced Robotic Systems*, vol. 16, 2019.

[101] R. Abbas and Q. Wu, "Improved leader follower formation controller for multiple quadrotors based afsa," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 13, p. 85, Mar. 2015.