

People's Democratic Republic of Algeria

الجمهورية الجزائرية الديمقراطية الشعبية

Ministry of Higher Education and Scientific Research

وزارة التعليم العالي والبحث العلمي

Blida 01 University

Institute of Aeronautics and Space Studies

Construction Department



End-of-study Thesis

In view of obtaining a Master's Diploma in Aeronautics

Option: Avionics

*Design and implementation of a robust controller on
a Quadrotor system*

Presented by:

Ms. SELMANE Nour EL Houda

Supervisors:

Mr. ALLAM Ahmed

Mr. LAGHA Mohand

Defense Jury:

Jury president: Dr. TAHRAOUI Sofiane

Examiner: Dr. AZMEDROUB Boussaad

Promotion: 2022/2023

Acknowledgment

First and foremost, I humbly acknowledge my profound gratitude to Allah, the most merciful and compassionate, for guiding me to the right path and helping me overcome difficulties. His unwavering support has granted me the strength, resilience, and unwavering resolve to undertake and successfully complete this thesis.

I would like to extend my heartfelt gratitude to my supervisors, Dr. ALLAM Ahmed for their guidance, and mentorship throughout the entire duration of this project. Their expertise and constructive feedback have been invaluable. I am also grateful to Prof. LAGHA Mohand for their support, valuable insights and contributions to this research.

I extend my gratitude to the members of the jury for the honor they have bestowed upon us by agreeing to evaluate our work.

Special thanks to all the staff at the Institute of Aeronautics and Space Studies for their invaluable contributions and dedication.

I am deeply thankful to my parents; my mother Faiza for her unwavering encouragement, love and support, my father Ali for his support, patience and understanding throughout my academic journey. To my brothers Oussama, Islam and Nadhir for their love and care, their presence in my life has been a source of strength and joy. To my grandmother for her love, kindness and thoughtfulness that have made my life brighter, my uncle Amine for the lifelong friendship we've built. To my cousin Anissa for helping me, inspiring me to strive for excellence and for being a role model in my life. To my uncle Mohammed (may Allah have mercy on him) for his prayers, constant encouragement and motivation. To all my family for everything they did to make me feel precious. Their support has been my pillar of strength.

Finally, I want to express my gratitude to my best friends; Imane, Soultana, Raounak, Yasmine and Amira for simply being there, through thick and thin. Their presence in my life is a gift I treasure every day. To all my friends from clubs, events and university, for their moral support and for being a source of inspiration during my academic journey.

I dedicate this work to my loving parents, who have always believed in me and supported my dreams. Your love and encouragement have been my guiding light.

Table of contents

LIST OF TABLES	6
LIST OF FIGURES.....	7
ACRONYMS	8
LIST OF SYMBOLS.....	9
ABSTRACT	11
GENERAL INTRODUCTION	13

CHAPTER 1 : EVOLUTION OF QUADROTORS AND ITS CONTROL TECHNIQUES

Introduction	4
1.1. Classification of UAVs and their application	4
1.1.1. Fixed-Wing Drones	4
1.1.2. Multirotor Drones.....	5
1.1.3. Single Rotor Drones	6
1.1.4. Hybrid VTOL Drones.....	6
1.1.5. Nano Drones:.....	7
1.2. Evolution of Drone (quadrotors)	7
1.3. Quadrotor control techniques.....	11
1.4. Hybrid Flight Control Systems.....	15
Conclusion	16

CHAPTER 2 : DYNAMIC MODELING OF A QUADROTOR

Introduction	18
2.1. General description of the quadrirotor	18
2.2. Quadrotor movements	19
2.2.1. Vertical movement.....	20
2.2.2. Roll movement	21
2.2.3. Pitch movement.....	21
2.2.4. Yaw movement	22
2.2.5. Translational movements.....	22
2.3. Dynamic model of the quadrotor	23

2.3.1. Euler Angles	24
2.3.2. Angular velocity	25
2.3.3. Linear velocity.....	25
2.3.4. Development of the Newton-Euler mathematical model	26
2.3.4.1. Equations of translation movement.....	27
2.3.4.2. Equations of rotational movement	27
2.3.5. System state representation	29
2.2.6. Rotor dynamics	30
Conclusion	31

CHAPTER 3 : CONTROLLER DESIGN AND SIMULATION RESULTS

Introduction	34
3.1. Synthesis of quadrotor control laws.....	34
3.2. Linear control for quadrotors.....	35
3.2.1. Classical PID control.....	35
3.2.2. Characteristics of P. I. and D. controllers	35
3.2.3. Classic PID quadrotor control	37
3.2.3.1. Altitude, Attitude and Heading Control.....	39
3.2.3.2. Position Controller.....	39
3.2.4. PID Controller Simulation	39
3.3. Non-linear control for quadrotors.....	43
3.3.1. Sliding Mode Controller (SMC)	43
3.3.1.1. Introduction to SMC	43
3.3.1.2. Principle of Sliding Mode Control (SMC)	44
3.3.1.3. Quadrotor sliding mode control	47
3.3.1.4. Results of sliding mode (SMC) model simulation	49
3.3.2. Non Linear PID (NLPID) (Modified SMC).....	52
3.3.2.1. Introduction to Non Linear PID	52
3.3.2.2. Altitude Control	53
3.3.2.3. Attitude and Heading Control	53
3.3.2.4. Position Controller.....	54
3.3.2.5. Results of NLPID model simulation	54
3.3.3. Active Disturbance Rejection Control (ADRC).....	57
3.3.3.1. ADRC altitude control	57
3.3.3.2. System state representation.....	58

3.3.3.3. Extended state estimator	58
3.3.3.4. Control law	59
3.3.3.5. Results of ADRC model simulation	61
3.4. Interpreting simulation results	62
3.5. Comparison of the four control techniques	63
3.6. Overview of Robustness.....	63
Conclusion	64

CHAPTER 4 : IMPLEMENTATION

Introduction	67
4.1. Overview about the Parrot AR Drone 2.0	67
4.1.1. AR Drone 2.0 system parameters	68
4.1.2. Engines	69
4.1.3. LiPo batteries	69
4.1.4. Motion sensors	69
4.1.5. Assisted control of basic manoeuvres	70
4.1.6. Advanced manoeuvres using host tilt sensors	70
4.1.7. Wifi network and connection	70
4.2. AR Drone tool box kit	71
4.2.1. AR-Drone Simulation Block	71
4.2.2. AR-Drone Wi-Fi Block	71
4.2.3. Waypoint Tracking	72
Conclusion	75
GENERAL CONCLUSION	76
REFERENCES	79

List of tables

Tableau 1-1. Research projects on quadrotors.....	9
Tableau 1-2. Performance evaluation of control techniques.	15
Tableau 3-1. K_p , K_i and K_d controllers.	36
Tableau 3-2. Parameters obtained through identification.....	40
Tableau 3-3. PID Parameters.....	40
Tableau 3-4. SMC Parameters.....	49
Tableau 3-5: Overview of robustness.....	64
Tableau 4-1. Parameters of the Parrot AR. Drone 2.0 model.....	68

List of figures

Figure 1.1. Fixed- wing drone.....	5
Figure 1.2. Multirotor drone.....	6
Figure 1.3. Single rotor drone.....	6
Figure 1.4. Hybrid VTOL drone.....	7
Figure 1.5. Nano drone.....	7
Figure 1.6. Bréguet Richet Gyroplane No. 1.....	8
Figure 1.7. Georges de Bothezat.....	8
Figure 1.8. Oehmichen No.2.	8
Figure 1.9. Curtiss-Wright.....	8
Figure 1.10. Classification of control methodologies for quadrotor control	12
Figure 2.1. Basic quadrotor design.....	19
Figure 2.2. Illustration of vertical movement.....	20
Figure 2.3. Illustration of roll movement.....	21
Figure 2.4. Illustration of pitching movement.....	21
Figure 2.5. Illustration of the yaw movement.....	22
Figure 2.6. Illustration of translational movement.....	23
Figure 2.7. Geometry of the quadrotor.....	24
Figure 3.1. Illustration of quadrotor control structure.....	34
Figure 3.2. Classical PID structure.....	35
Figure 3.3. Quadrotor PID control diagram.....	38
Figure 3.4. PID Controller Simulation Response.....	41
Figure 3.5. PID Control Inputs.....	42
Figure 3.6. Trajectory Response under PID.....	42
Figure 3.7. Sliding mode.....	45
Figure 3.8. Quadrotor SMC control diagram.....	47
Figure 3.9. SMC Controller Simulation Response.....	50
Figure 3.10. SMC Control Inputs.....	51
Figure 3.11. Trajectory Response under SMC.....	51
Figure 3.12. Nonlinear PID control architecture.....	53
Figure 3.13. NLPID Controller Simulation Response.....	55
Figure 3.14. NLPID Control Inputs.....	56
Figure 3.15. Trajectory Response under NLPID.....	56
Figure 3.16. Block diagram of closed-loop ADRC control.....	59
Figure 3.17. Block diagram of ADRC control applied to the quadrotor.....	60
Figure 3.18. ADRC Controller Simulation Response + control signal $u(t)$ for altitude Z	61
Figure 3.19. ADRC ESO error.....	62
Figure 4.1. AR Drone 2.0 without and with indoor hull.....	67
Figure 4.2. Drone Sensors.....	69
Figure 4.3. AR-Drone Simulation Block in yellow.....	71
Figure 4.4. AR-Drone Wi-Fi Block in bleu.....	72
Figure 4.5. Vehicle diagram and frames.....	73
Figure 4.6. Simulation instruction flowchart.....	73
Figure 4.7. Setup waypoints Tracking.....	74
Figure 4.8. ARdrone-PC WiFi connection.....	75

Acronyms

UAV Unmanned Aerial Vehicle

MUAV Multirotor Unmanned Aerial Vehicle

VTOL Vertical takeoff and landing

PID Proportional-Integral-Derivative (controller)

LQR Linear-Quadratic Regulator

ANN Artificial Neural Network

SMC Sliding Mode Control

MPC Model Predictive Control

ADRC Active Disturbance Rejection Control

NSS Nominal System Stability

RSS Robust System Stability

CIC Controller Implementation Complexity

DOF Degrees of Freedom

DC Direct Current

NLPID Non Linear Proportional-Integral-Derivative (controller)

ESO Extended State Observer

PD Proportional-Derivative

EPP Expanded Polypropylene

RAM Random Access Memory

API Application Programming Interface

MEMS Micro-Electro-Mechanical Systems

IMU Inertial Measurement Unit

ESSID Extended Service Set Identifier

DHCP Dynamic Host Configuration Protocol

CPU Central Processing Unit

List of Symbols

ζ Position vector of the quadrotor

$C = \cos$ And $S = \sin$

Ω_i Angular velocities in the fixed frame

v_i^b Linear velocities in the fixed frame

m Total mass of the quadrotor

R Rotation matrix

\wedge The vector product

J Symmetrical inertia matrix of dimension (3x3)

$S(\Omega)$ the antisymmetric matrix

F_f total force generated by the four rotors

F_t The drag force along the axes (x, y, z)

K_{ftx}, K_{fity} and K_{ftz} Drag coefficients for translation

F_g Force of gravity

M_f Moment caused by thrust and drag forces

M_a Moment resulting from aerodynamic friction

$K_{fax}, K_{fay}, K_{faz}$ Aerodynamic friction coefficients

I Identity matrix

U Control input vector

X State vector

ω Angular body rates

ϕ Roll angle

ϕ_d Desired roll ϕ

ψ yaw angle

ψ_d Desired yaw ψ

θ pitch angle

θ_d Desired pitch θ

e Error

x_d Desired x position

y_d Desired y position

- z_d Desired altitude z
- τ_i input torque
- Q_i resistive torque generated by rotor i
- $\omega_{d,i}$ Command values provided by the controller
- M non-singular matrix
- R_a motor resistance
- k_m motor torque constant
- k_g gearbox gain
- K_p proportional controller
- K_i integral controller
- K_d Derivative control
- g Gravitational acceleration
- L Distance to the center of the Quadrotor
- b Thrust factor
- d Drag factor
- I_{xx} Quadrotor moment of inertia around X axis
- I_{yy} Quadrotor moment of inertia around Y axis
- I_{zz} Quadrotor moment of inertia around Z axis
- J_r Rotor inertia
- s sliding surface
- λ_i the sliding surface parameter
- x_i the state of the system
- x_{d_i} the desired state
- $T_{i(\cdot)}$ the integral time constants of the NLPID structure
- $T_{d(\cdot)}$ the derivative time constants of the NLPID structure
- $f(t)$ total disturbance
- $\hat{f}(t)$ estimated total disturbance
- $\hat{x}_i(t)$ estimated state variable
- L Luenberger Observer gain

Abstract

The proposed work consists in designing and implementing four distinct robust controllers namely Proportional-Integral-Derivative (PID), Sliding Mode Control (SMC), Nonlinear PID (NLPID), and Active Disturbance Rejection Control (ADRC), in order to improve the performance of the physical quadrotor system in terms of stability and robustness. The first stage consists in building an equation for the dynamic model of the system, then simulating the four control methods in Simulink and Matlab, with a comparison.

In a second stage, an experimental validation is programmed on an ARdrone 2.0 flying platform, in order to assess the feasibility and effectiveness of the proposed robust control system. This research contributes to the advancement of quadrotor control technology, offering valuable insights into the practical deployment of these controllers in real-world applications.

Keywords: Modeling, Matlab (Simulink), Quadrotor, AR Drone 2.0, PID Controller, Sliding Mode Control (SMC), Nonlinear PID, Active Disturbance Rejection Control (ADRC).

ملخص

العمل المقترح يتضمن تصميم وتنفيذ أربع خوارزميات تحكم قوية، وهم التحكم نسبي تكاملي تفاضلي (PID)، التحكم بالانزلاق (SMC)، التحكم نسبي تكاملي تفاضلي غير الخطي (NLPID)، والتحكم النشط في رفض التشويش (ADRC)، بهدف تحسين أداء النظام الفيزيائي لطائرة درون رباعية الدوران من حيث الاستقرار والقوة. الخطوة الأولى تشمل بناء معادلة للنموذج الديناميكي للنظام، ثم محاكاة الطرق الأربعة للتحكم في بيئة Simulink و Matlab مع إجراء مقارنة.

في الخطوة الثانية، تم برمجة تحقيق تجريبي على منصة الطيران AR Drone 2.0، بهدف تقييم جدوى وفعالية النظام المقترح للتحكم القوي. يساهم هذا البحث في تقدم تكنولوجيا التحكم في طائرات الدرون رباعية الدوران، من خلال توفير مؤشرات قيمة حول تنفيذ نظم التحكم في تطبيقات الواقع.

الكلمات المفتاحية: النمذجة، Matlab(simulink)، طائرة درون AR Drone 2.0، التحكم نسبي تكاملي تفاضلي (PID)، التحكم بالانزلاق (SMC)، التحكم نسبي تكاملي تفاضلي غير الخطي (NLPID)، التحكم النشط في رفض التشويش (ADRC).

Résumé:

Le travail proposé consiste à concevoir et à mettre en œuvre quatre contrôleurs robustes distincts, à savoir le contrôle proportionnel-intégral-dérivé (PID), le contrôle en mode glissant (SMC), le PID non linéaire (NLPID) et le contrôle actif de rejet des perturbations (ADRC), afin d'améliorer les performances du système physique du quadrotor en termes de stabilité et de robustesse. La première étape consiste à construire une équation pour le modèle dynamique

du système, puis à simuler les quatre méthodes de contrôle dans Simulink et Matlab, avec une comparaison.

Dans une deuxième étape, une validation expérimentale est programmée sur une plateforme volante ARdrone 2.0, afin d'évaluer la faisabilité et l'efficacité du système de contrôle robuste proposé. Cette recherche contribue à l'avancement de la technologie de contrôle des quadrotors, en offrant des indications précieuses sur le déploiement pratique de ces contrôleurs dans des applications réelles.

Mots-clés: Modélisation, Matlab (Simulink), Quadrotor, AR Drone 2.0, Contrôleur PID, Contrôle en mode glissant (SMC), PID non linéaire, Contrôle actif de rejet des perturbations (ADRC).

General

Introduction

General Introduction

Unmanned aerial vehicles (UAVs), have attracted a great deal of interest from the scientific community for a number of decades. These flying devices were originally used largely in the military domain for surveillance and assessment of sensitive areas. However, in recent years, the use of these aerial vehicles has spread to a wide range of industries, including industrial, civilian, and emergency situations. When compared to vehicles with a pilot, using these flying robots offers many benefits, including the ability to protect others from risky operations and reduce maintenance costs.

There are different types of drones, which vary based on their specific applications including surveillance, inspection, and aerial photography. The quadrotor discussed in this introduction belongs to the category of Multirotor Unmanned vehicles MUAVs. They offer great maneuverability and agility and can fly in limited spaces, which is different from fixed-wing drones for example, that require special runways for takeoff and landing. However, controlling quadrotors in challenging environments with disturbances and uncertainties remains a significant challenge. Existing control methods often struggle to maintain stability and accurate trajectory tracking in the presence of external disturbances, parameter uncertainties, and nonlinear dynamics.

The problem at hand is the need for a robust command solution that can effectively address the control challenges faced by quadrotors. Specifically, there is a need for a control algorithm that can ensure stability, accurate tracking, and robust performance in the presence of disturbances and uncertainties.

The objective of this thesis is to design and implement robust nonlinear control techniques, namely Proportional Integral Derivative (PID), Modified Sliding Mode Control (SMC), a Modified Sliding Mode Controller (SMC) known as Non Linear PID (NLPID) and an Active Disturbance Rejection Control (ADRC), to enhance the performance of a quadrotor system in terms of stability and robustness.

Structure of the project:

This thesis project is structured into four comprehensive chapters, each contributing to a deeper understanding of quadrotors and their control mechanisms.

1. The First Chapter is structured around two sections; in the first section, an overview of the history of drones is given with a special emphasis on quadrotors. It covers the historical evolution and developments of quadrotor technology, emphasizing significant turning points that influenced their current state. The second section provides an overview of the principal control techniques and commands used for quadrotors. It gives an idea about these techniques, evaluates their performance according to some criteria, to bring and justify the choice and orientations of the thesis.
2. The dynamic modeling of the quadrotor is covered in Chapter 2. First, we give a description of the system, including its fundamental architecture and flight dynamics. The quadrotor's dynamic modeling is then shown, showcasing the main traits and behavior of the device. Finally, we demonstrate the state model for this system's control.
3. The Third Chapter introduces the theoretical background of four control techniques: the PID controller (proportional, integral, derivative), the sliding mode control (SMC), the modified sliding mode controller known as (NLPID) and finally the Active Disturbance Rejection Control (ADRC). Then, we simulate these control laws in Matlab.
4. The final phase of this thesis project is shown in Chapter 4, where the control techniques and theoretical knowledge are implemented. Specifically, an experimental validation is envisaged on a flying platform such as ARdrone 2.0, in order to evaluate the feasibility and the superiority of the proposed robust control in the real-world.

Finally, we conclude our project with a general conclusion and outline our future work.

Chapter 1

Evolution of quadrotors and its control techniques

Introduction

Unmanned Aerial Vehicles (UAVs), better known as "Drones", are aircrafts that are operated remotely without a human pilot on board. They have been widely adopted in the military world over the last decade and the success of these military applications is increasingly driving efforts to establish unmanned aircraft in non-military roles [1]. UAVs have become increasingly popular in recent years due to their adaptability and ability to perform a wide range of tasks. They have been used in various fields such as agriculture, surveying, search and rescue operations [2]. UAVs are composed of several subsystems that work together to achieve a specific mission. These subsystems include the air vehicle, mission planning and control [3]. UAVs come in different sizes, ranging from micro UAVs to large UAVs, and can be classified based on their range and endurance and missions [4]. The quadrotors discussed in this chapter, are the most popular type of UAVs, due to their numerous advantages including; stability, maneuverability(they can fly indoors as well as outdoors due to their small size), cheaper and more durable, better control during flight and their capacity of carrying payloads.

The purpose of this work is to briefly summarize the evolution of one of these MUAVs (known as quadrotors), then describe the synthesis and application of control techniques on modern quadrotors.

1.1. Classification of UAVs and their application

The Classification of Unmanned Aerial Vehicles (UAVs) encompasses a wide array of types, each designed with specific attributes to address various applications. An overview to gain insight into how different UAV classifications align with diverse real-world applications is described in what follows:

1.1.1. Fixed-Wing Drones

These drones have a fixed wing design resembling airplanes. They are known for their efficient flight capabilities and longer endurance (they can fly for a really long time without needing to stop or land to fuel). They have many applications:

- a) **Aerial Mapping:** Fixed-wing drones are used for capturing large geographic areas with high precision, creating detailed maps for various purposes like urban planning and land assessment.
- b) **Surveillance:** They are employed in monitoring remote areas, border control, and security tasks. Their long endurance allows for extended surveillance operations.

- c) **Agriculture:** Fixed-wing drones are used for precision agriculture, monitoring crops, assessing soil conditions, and optimizing irrigation.
- d) **Delivery:** In recent years, fixed-wing drones have been explored for delivering goods to remote or hard-to-reach areas, especially in emergency situations or rural locations.



Figure 1.1. Fixed- wing drone.

1.1.2. Multirotor Drones

Multirotor drones are characterized by having multiple rotors that allow them to take off and land vertically. They are highly maneuverable and commonly used for various aerial tasks such as:

- a) **Photography:** Multirotor drones are popular for capturing stunning aerial photos and videos. They provide unique perspectives for photography and cinematography.
- b) **Videography:** These drones are used for filming events, movies, documentaries, and real estate videos. Their ability to hover and change angles makes them suitable for dynamic shots.
- c) **Inspections:** Multirotor drones are employed for inspecting infrastructure, power lines, buildings, and other difficult-to-reach areas. They provide visual data without endangering workers.
- d) **Search & Rescue:** These drones are used in locating missing persons or objects in remote or hazardous environments, assisting search and rescue operations.



Figure 1.2. Multirotor drone.

1.1.3. Single Rotor Drones

Single rotor drones work a bit like helicopters, with a big rotor for lift and a smaller one for stability. They can be used in:

- a) **Aerial Cinematography:** These drones are equipped with stabilized cameras for capturing cinematic footage in controlled and stable flight conditions.
- b) **Industrial Inspections:** Single rotor drones are used to inspect tall structures like telecommunications towers, wind turbines, and industrial facilities. They can go to places that might be risky for humans to reach.



Figure 1.3. Single rotor drone.

1.1.4. Hybrid VTOL Drones

Hybrid VTOL (Vertical Takeoff and Landing) drones combine features of both fixed-wing and multirotor drones. They can take off and land vertically and fly efficiently like fixed-wing drones during longer missions. They can be applied in:

- a) **Long-Range Surveying:** Hybrid VTOL drones are used to cover large areas in tasks like surveying and mapping. They are great for environmental assessments.

- b) **Environmental Monitoring:** These drones are used to study wildlife, habitats, and natural environments in remote locations.



Figure 1.4. Hybrid VTOL drone.

1.1.5. Nano Drones:

Nano drones are small, lightweight UAVs with sizes around 10 mm. They are designed to navigate tight spaces and perform tasks that are challenging for larger drones.

- a) **Indoor Reconnaissance:** Nano drones are used to navigate indoor spaces, gather information, and assess potentially hazardous environments.
- b) **Research:** In scientific research, nano drones are used to study insects, small environments, and conduct experiments in confined spaces.



Figure 1.5. Nano drone.

1.2. Evolution of Drone (quadrotors)

The history of quadcopters and multirotors dates back to the early pioneers who first attempted rotor flight using multicopters, as using more than one rotor seemed to be the natural solution to the problem of VTOL flight [5].

The first quadrotor was constructed by the Breguet Brothers in 1907. Although the Gyroplane No. 1's flights (shown in Figure 1.6) are considered as the first manned helicopter flights, they were not fully unattached due to issues with stability and control mechanisms [6,

7]. Later attempts at manned quadrotors included Georges de Bothezat's Flying Octopus in 1922 (Figure 1.7) and Etienne Oehmichen's Oehmichen No.2 in the same year (Figure 1.8). Further significant examples are The Convertawings Model A, created in 1922 by Oehmichen and de Bothezat, the Curtiss X-19 (Figure 1.9) created in 1963 by Curtiss-Wright corporation, the Bell X-22A created in 1966 by Bell Aircraft Corporation, and the flying vehicles developed by the Moller Company [8].

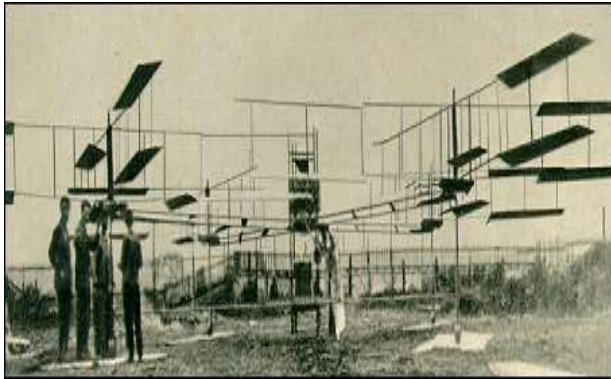


Figure 1.6. Bréguet Richet Gyroplane No. 1



Figure 1.7. Georges de Bothezat.

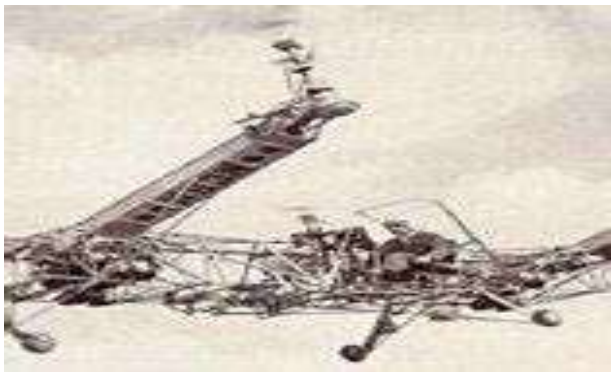


Figure 1.8. Oehmichen No.2.

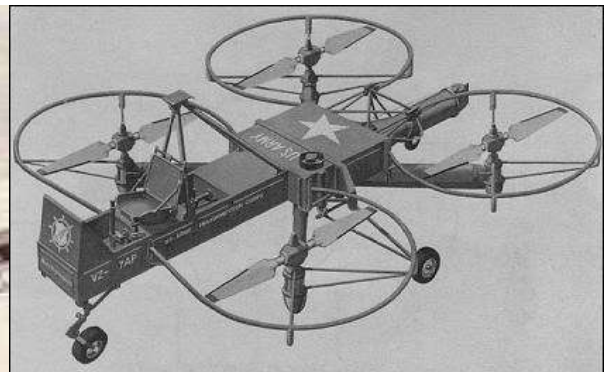











Figure 1.9. Curtiss-Wright.

After these failed attempts, advances in technology and control theory expertise made it possible to create unmanned quadrotors. Several academic institutions and research facilities began using quadrotors in their studies following those advancements. (Table 1) [9].

Tableau 1-1. Research projects on quadrotors.

	University/ Organization	Project	Year	Early studies
	Dragan flyer[45]	V Ti	1998	Commercial product
	Stanford[46]	Mesicopter-I, Kroo	2000-2012	-Feasibility and capability of the vehicle -Design and Manufacturing Techniques
	ANU[47]	P. Pounds's thesis	2002-2014	-Dynamic modeling based on Newton-Euler Method - Control attempt
	FEIT, ANU[47]	X-4 Flyer, P., Pounds		
	Uni. Pennsylvania[49]	E. Altug	2004	Yaw and height control using Visual feedback control techniques
	Uni. Compiègne[50]	P. Castillo's thesis, A. Dzul	2003-2015	-Dynamic modeling using Lagrange approach -Linear trajectory tracking
	Stanford[51]	Starmac	2004-2011	-Altitude and attitudes control in presence of wind
	Stanford[51]	StarmacII		
	EPFL[52]	Bouabdallah & Siegwart	2004-2011	-Autonomous control of the vehicle in indoor environment

Chapter 1 : Evolution of quadrotors and its control techniques

	Cornell University[48]	Eryk Brian Nice's thesis and R. D'Andrea	2004-2015	-Nonlinear dynamic modeling and hover control
	Middle East Technical University, Turkey[53]	F.B. Çamlica's thesis, C Özgen	2004-2014	Hover control
	Uni. Oldenburg[54]	M. Kemper's thesis	2006-2009	Robust control of quadrotor respect to variable center of gravities
	MIT[55]	P. Tournier and J.P. How	2007-2015	Autonomous control of quadrotor by using visual servoing method
	Uni. TUDelft[56]	Menno Wierema and Ir. C. de	2007	Autonomous indoor navigation
	IARC Team - Virginia Tech. Uni[57]	IARC Team Quadrotor	2009	Carrying payload
	Univ. Maryland[58]	AVL's Micro Quad (J. Sean Humbert)	2009-2015	Robust visual navigation]
	Azad University of Ghazvin[59]	Farshid Jafari Harandi	2010	Outdoor navigation
	CrazyFlie[60]	CrazyFlie	2011	Commercial product
	DJI[61]	DJI inspire 3	2023	Commercial product
	Skydio[62]	Sky dio 2 plus	2023	Commercial product

Due to its complex construction, modeling a quadrotor is not a simple process. Quadrotor modeling takes into account a variety of assumptions depending on specific missions. These presumptions are introduced in what follows [9].

1. **Gyroscopic effect:** The gyroscopic effect is a phenomenon that occurs in quadrotors due to the rotation of the propellers. The gyroscopic effect of the propellers causes the quadrotor to experience a body roll-inducing torque when it pitches [10]. This effect can cause instability in the quadrotor and affect its control.
2. **Ground effect:** is defined as the increase in thrust (at constant power) of a rotor operating near the ground [11].
3. **Drag and lift force of propeller rotation:** As the blades of the propellers rotate in the air, a damping force and a lift force are generated [9].
4. **Structure:** Most of the works on quadrotors have assumed that the structure of the quadrotors is rigid and symmetrical, therefore the inertia tensor of the quadrotor must be diagonal [12].
5. **Blade flapping:** Blade flapping effect is very important, as the tilting rotor can introduce significant stability effects for the vehicle [9].
6. **Critical conditions in taking-off and landing:** Critical scenarios in taking off and landing, such as in the presence of sloped terrains and surrounding obstacles [9].
7. **Wind:** Wind has a significant nonlinear effect on the navigation, inertial orientation and rates of the vehicle [9].

Naturally, there are certain control strategies to counteract these effects. These approaches are introduced in what follows.

1.3. Quadrotor control techniques

When quadrotor technology was first being developed, manual control was the main way to fly the drone. Adjusting the quadrotor's throttle, pitch, roll, and yaw required direct input from a human operator.

During the last two decades, there has been a significant amount of study on quadrotor control. For quadrotors, many control approaches and algorithms have been developed. According to the following and as indicated in (Figure 1.10) [15], there are three categories into which control techniques for quadrotors may be divided [15].

- **Linear:** (PID, LQR, Gain Scheduling and H_∞).

- **Nonlinear** (feedback linearization, backstepping, sliding mode, adaptive control, and model predictive control).
- **Learning-based intelligent** (fuzzy logic, human based learning and artificial neural network (ANN)] controllers).

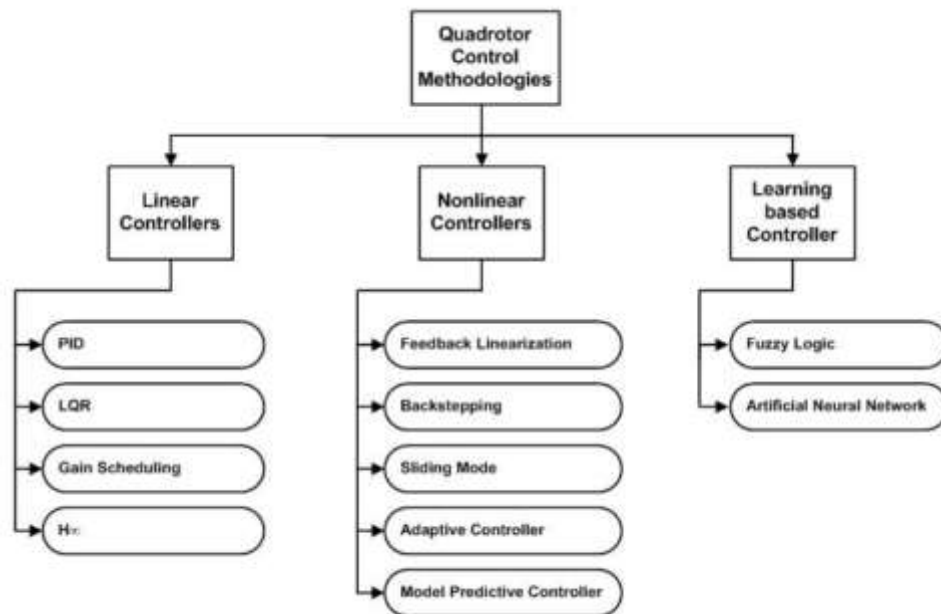


Figure 1.10. Classification of control methodologies for quadrotor control .

The original methods of choice for Multirotor Unmanned aerial vehicle (MUAV) flight control have been linear control approaches. PID, LQR, H_∞ and gain scheduling controllers are some of these methods [15].

1. Linear controllers

They are the most typical and conventional flight control systems, typically based on PID, Linear Quadratic (LQ), or H_∞ algorithms. A full-scale helicopter reportedly accomplished autonomous waypoint navigation in the late 1960s using the classical linear control technique.

PID control: The first control technique used in quadrotors was the classical control, such as PID control [13,14]. For industrial applications, PID controllers are particularly effective and practical. It has also demonstrated success in quadrotor control. PID has the benefit of being implemented without the need for a dynamical model knowledge, and PID gains may be adjusted by hit and trial [15]. Although it is a linear controller used for the nonlinear multivariable quadrotor system, it was proven successful in many literature

LQR: One category of optimal controllers is the linear-quadratic regulator (LQR). The use of LQR to quadrotor control has been effective and it offers robust performance [15].

Gain scheduling: Gain scheduling is a model-based control method that is frequently utilized in nonlinear control designs. The use of linear control methods for nonlinear system control is made possible by this model-based approach [15].

H ∞ : The most popular robust controller in use today is the H ∞ controller. Robust control methodology offers a variety of techniques to control dynamical systems having unmodeled dynamics and parametric uncertainties. In literature, linear H ∞ controllers have been applied on linearised models of quadrotors [15].

Different quadrotors have successfully used linear controllers and demonstrated their effectiveness. Linear controllers, on the other hand, are made for linearized models, and their performance suffers when quadrotors stray from their operating schedules or are subject to certain physical constraints. Despite these limitations, linear controllers are recognized methods for quadrotor control [15].

2. NON- Linear controllers

To address the limitations of linear control techniques used to quadrotors, many nonlinear control approaches have been developed; they are based on nonlinear dynamic models. Some of these techniques include feedback linearization, backstepping, sliding mode control (SMC), model predictive control (MPC), Active Disturbance Rejection Control (ADRC) and adaptive control [15].

Feedback linearisation: As a nonlinear control approach known as feedback linearization, state variables from the dynamical model are transformed into a new coordinate system. Since the resultant system possesses linear dynamics, linear procedures are then used, and later, the coordinates are reverse-transformed to restore the original dynamics [15].

Backstepping: A well-known method for underactuated system control is backstepping. This approach lacks robustness but has a quick convergence rate and is capable of handling external disturbances [15].

Sliding mode control: SMC is a robust nonlinear control technique that is simple to use and guarantees resilience for known disturbances. A detailed overview is presented in the third chapter.

Adaptive control: is an effective robust control method for systems having unmodeled dynamics and model uncertainties [15].

Model Predictive Control: Another nonlinear method that has been used to MUAV control is MPC. By resolving optimum control issues, MPC incorporates a dynamic model of the system to predict future system states while minimizing error [15].

Active Disturbance Rejection Control (ADRC): is a nonlinear control technique that focuses on effectively rejecting disturbances and uncertainties in control systems, even in the presence of nonlinearities and complex dynamics. A detailed overview is presented in the third chapter.

For the control of quadrotors, nonlinear controllers such as feedback linearization, sliding mode controller, backstepping control, adaptive control, ADRC and MPC have been effectively used. When it comes to tracking precision, system uncertainty, and disturbance rejection, nonlinear controllers outperform linear control techniques. However, significant implementation progress has not been witnessed. The implementation of these approaches requires more expensive computing, which is the cause [15].

3. Learning-based intelligent controllers

A dynamic model of the MUAV is not always necessary when using learning-based intelligent controllers; instead, the system is trained using data from the MUAV's flight tests. There are now flight controllers for MUAVs that are based on fuzzy logic and neural networks [15].

Fuzzy logic: Fuzzy logic is an approach to computing that allows “degrees of truth” rather than the usual “true or false” Boolean logic [16]. It has been applied successfully to quadrotor in combination with other control techniques [15].

Artificial Neural Network: ANN is a type of machine learning algorithm that is modeled after the structure and function of the human brain. It is widely used in intelligent controllers. Researchers have employed ANN for quadrotor system identification, estimation and control [15].

Multiple experiments have been used to effectively verify learning-based intelligent controllers. Model-free approach enables these controllers to be used on different MUAV configurations. However, Analyzing the stability and robustness of these approaches is still challenging [15].

(Table 2) [15] Provides a summary of how proposed control approaches for quadrotor control performed when measured against defined evaluation criteria. Control techniques are listed in rows, while evaluation criteria are listed in columns. Table 1 entries for the criterion NSS and RSS are marked with yes or mixed (in combination with other techniques), while CIC is labeled with a low, medium, or high score [15].

Tableau 1-2. Performance evaluation of control techniques.

Category	Control Techniques	NSS	RSS	CIC
<u>Linear</u>	PID	Yes	Mixed	Low
	LQR	Yes	Mixed	Low
	Gain scheduling	Yes	Mixed	Medium
	H∞	Yes	Yes	Medium
<u>Nonlinear</u>	Feedback linearisation	Yes	Mixed	Medium
	Backstepping	Yes	Mixed	Medium
	Sliding mode SMC	Yes	Mixed	Medium
	Adaptive	Yes	Mixed	Medium/high
	MPC	Yes	Mixed	Medium/high
<u>Intelligent</u>	Fuzzy Logic	Mixed	Mixed	Medium/high
	ANN	Mixed	Mixed	High

1.4. Hybrid Flight Control Systems

In recent literature, researchers are recommending using more than one type of flight control algorithms after it was discovered that using just one type of flight control algorithms was not enough to ensure good performance, especially when the quadrotor is not flying close

¹ **Performance Evaluation Criteria:**

- Nominal System Stability (NSS): the system stays stable under normal conditions.
- Robust System Stability (RSS): the system remains stable even when faced with unexpected changes or uncertainties
- Controller Implementation Complexity (CIC): refers to how difficult or complicated it is to put a controller into action

to its nominal condition. A research is based on the trajectory-tracking controller design for the biplane quadrotor by using three different nonlinear control methods: Backstepping Control (BSC), Integral Terminal Sliding Mode Control (SMC), and Hybrid control (ITSMC + BSC) where ITSMC and BSC control the position and attitude subsystems, and the criteria for evaluation is the performance in tracking objectives [43]. A hybrid flight control scheme that combined a backstepping controller with a Neural Networks controller is introduced in [44]. The backstepping controller focused on attitude control, ensuring accurate and responsive maneuvers, while the neural network controller is used to compensate for the uncertainties and nonlinearities in the system. By integrating these controllers, the quadrotor achieved stable flight with improved performance in the presence of uncertainties and nonlinearities. Another research proposed in [35] which is a Sliding Mode based Nonlinear PID Controller for Quadrotor that will be exploited in our project, so an overview of this technique is mentioned in chapter three.

Conclusion

In this chapter, we have given a brief overview about the evolution of one the UAVs, precisely, quadrotors. Along with a brief overview about the principal control techniques and commands frequently used for quadrotors.

This foundation of knowledge and analysis serves as a solid basis for the next chapters, where four of the proposed control techniques which are PID and SMC, a modified SMC and ADRC can be further explored and evaluated.

Now we're going to focus on a configuration that has attracted a great deal of interest over the last ten years, whether on a scientific or industrial scale, i.e. the quadrotor, which we'll model dynamically in the next chapter.

Chapter 2

Dynamic modeling of a quadrotor

Introduction

In order to design a flight controller, we must first have a good understanding of the quadrotor's movements, its dynamics and consequently its dynamic equations. This understanding is necessary not only for controller design, but also to ensure that the simulations of our vehicle behavior are as close as possible to reality when the control is applied.

To represent the studied system mathematically, a group of approaches are combined in modeling. This requires both; deep understanding of the phenomena produced in our system and the ability to express them mathematically with equations.

The synthesis of dynamic system control laws requires an accurate system modelling so that the model can best predict the system's behavior under various excitations (control, interference, etc.) So, the more accurate it is, the more true to the system. However, this has complicated the process of looking for and putting together potential control laws. A compromise must be made by adopting simplified assumptions in order to meet real constraints [21].

2.1. General description of the quadrirotor

A quadrotor is an aerial mobile robot defined in space by 6 DOF, also known as multirotor unmanned aerial vehicle (MUAV), which has many potential and exciting missions, including indoor and urban flight. It is a VTOL (vertical take-off and landing) drone, which means that it can take off and land vertically (unlike aircraft, for example). The quadrotor has four rotors, located at the end of each arm of the cross, and the electronics of control are usually located in the center of the cross. To prevent the vehicle from spinning on itself; its yaw axis, it is necessary for two propellers to spin clockwise, and the other two spin counterclockwise. Each pair of rotors that rotate in the same direction must be placed on the opposite ends of a cross branch in order to steer the quadrotor.

The Quadrotor operation is very special. By modifying the motor's power, you can turn it up and down, tilt left/right (roll) or tilt forward/backward (pitch), and even turn (yaw) on itself, the quadrotor has six degrees of freedom, three rotational and three translational movements. [22].

These six degrees must be controlled by only four commands, so it's an under-activated system (the number of inputs is less than the number of outputs).

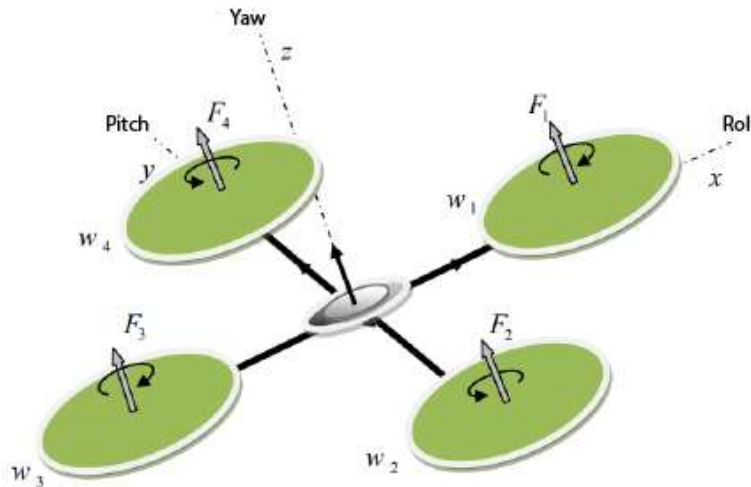


Figure 2.1. Basic quadrotor design.

2.2. Quadrotor movements

When the main rotor of a helicopter rotates, it generates a reactive torque that would cause the helicopter body to rotate in the opposite direction if this torque were not counteracted. This is usually done by adding a tail rotor that produces thrust in a lateral direction. However, this rotor, with its associated power supply, makes no contribution to thrust. With a quadrotor, on the other hand, the right and left rotors turn clockwise, while the front and rear rotors turn counterclockwise, this effectively compensates for unwanted reactive torque and allows the vehicle to glide without turning out of control. What's more, unlike conventional helicopters, all the energy expended in counteracting the rotary motion contributes to the thrust force. [23].

The fundamental maneuvers of a quadrotor involve adjusting the speed of its individual rotors to modify the thrust produced. As a result, the quadrotor tilts towards the slower rotor, influencing its motion in that specific direction. This behavior is similar to that of a conventional helicopter, where movements are interconnected, meaning the quadrotor cannot translate without experiencing roll or pitch. Consequently, altering the speed of a rotor induces motion in at least three degrees of freedom.

For instance, if we increase the speed of the left rotor, it triggers a rolling motion (the quadrotor tilts towards the slower rotor, located to the right), a yawing motion (disrupting the balance between clockwise and counterclockwise rotations, causing horizontal rotation), and a translational movement (the rolling motion inclines the frame, subsequently altering the

direction of the thrust force). This interdependence allows us to control all six possible movements of the quadrotor using only four commands (representing the torque applied to each rotor by the motors). The quadrotor has five main movements:

The quadrotor has five main movements:

1. Vertical movement
2. Roll movement
3. Pitch movement
4. Yaw movement
5. Horizontal translations

2.2.1. Vertical movement

In order to glide, all lift force should only be along the z-axis with a magnitude exactly opposite to the force of gravity. Moreover, the lift force created by each rotor must be equal to prevent the vehicle from tipping over. Consequently, the thrust produced by each rotor must be identical.

Upward and downward movement is achieved by varying the rotational speed of the motors (and consequently the thrust produced); if the lift force is greater than the weight of the quadrotor, the movement is upward, and if the lift force is less than the weight of the quadrotor, the movement is downward.

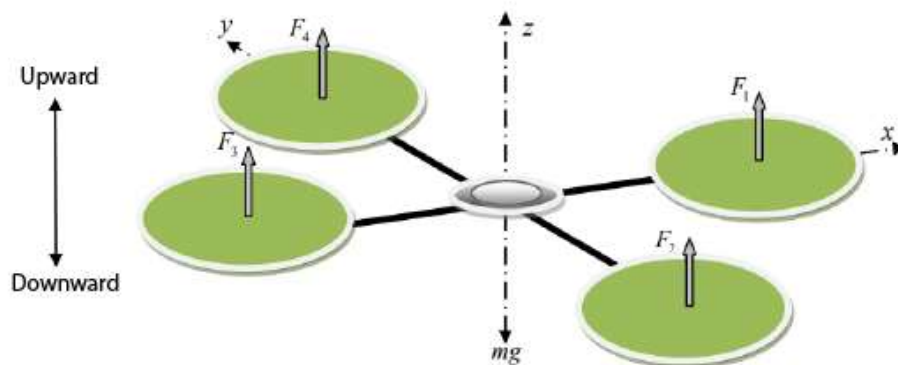


Figure 2.2. Illustration of vertical movement.

2.2.2. Roll movement

Figure (2.3) shows how roll motion is achieved. In this case, a torque is applied around the x-axis, i.e. a difference in thrust is applied between rotor 2 and rotor 4. This movement (rotation around the x-axis) is coupled with a translational movement along the y-axis.

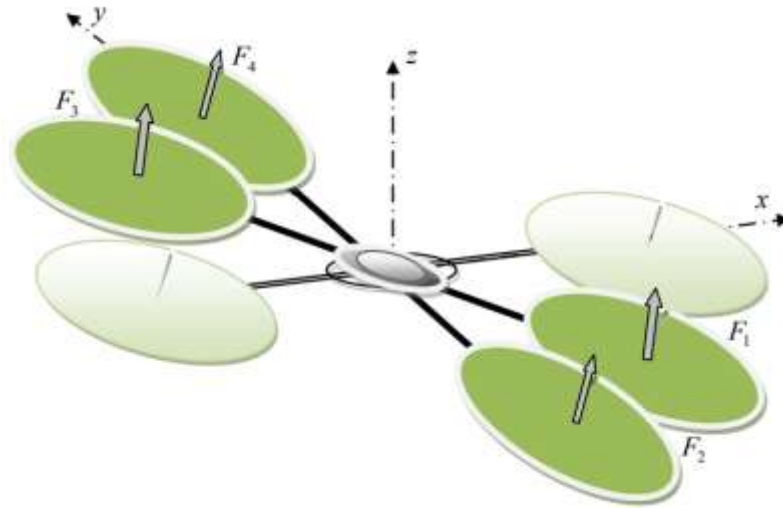


Figure 2.3. Illustration of roll movement.

2.2.3. Pitch movement

Figure (2.4) shows how pitch movement is achieved. In this case, a torque is applied around the y axis, i.e. by applying a thrust difference between rotor 1 and rotor 3. This movement (rotation around y) is coupled with a translational movement along the x axis.

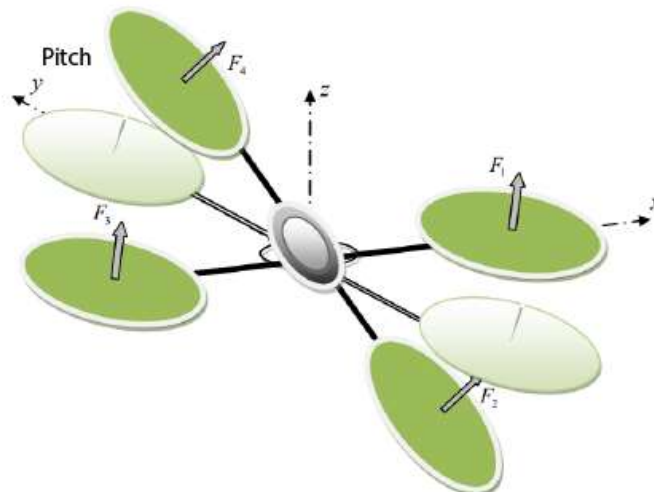


Figure 2.4. Illustration of pitching movement.

2.2.4. Yaw movement

Figure (2.5) shows how yaw motion is achieved. In this case, we want to apply a torque around the z-axis, which is done by applying a speed difference between rotors $\{1,3\}$ and $\{2,4\}$. This movement is not a direct result of the thrust produced by the propellers, but by the reactive torques produced by the rotation of the rotors. The direction of the thrust force does not shift during movement, but the increase in lift force in one pair of rotors must be equal to the decrease in the other pairs to ensure that the entire thrust force remains the same.

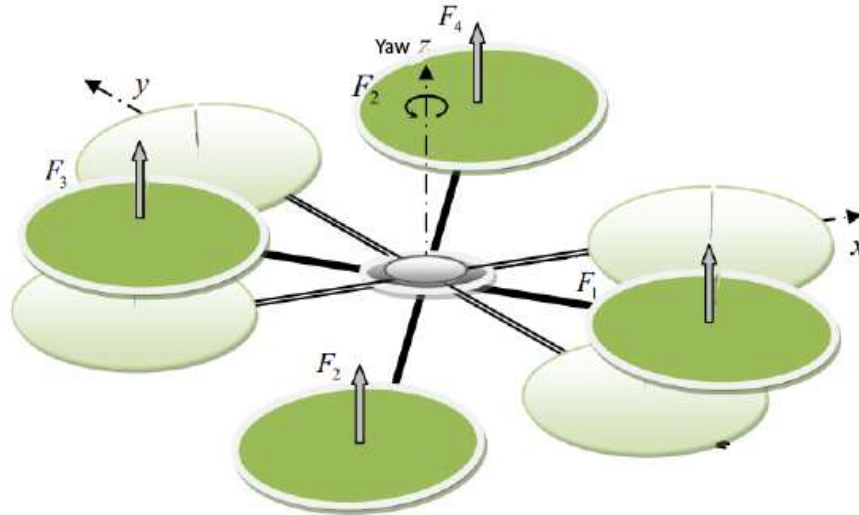


Figure 2.5. Illustration of the yaw movement.

2.2.5. Translational movements

Figure (2.6) shows how horizontal translation is achieved. In this case, we want to apply a force along x or y which is done by tilting the body (by pitching or rolling) and increasing any thrust produced to keep the magnitude of the z component of the thrust equal to the force of gravity.

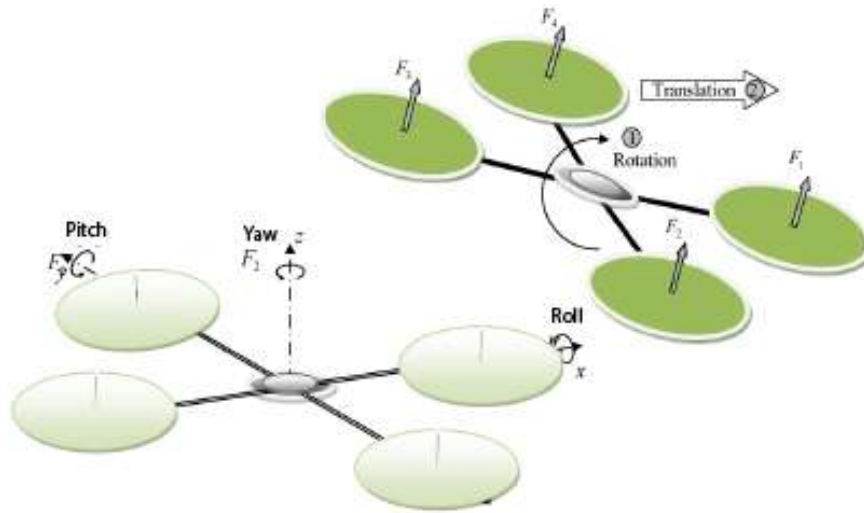


Figure 2.6. Illustration of translational movement.

2.3. Dynamic model of the quadrotor [24]

The motion of the quadrotor can be divided into two subsystems; rotational subsystem (roll, pitch and yaw) and translational subsystem (altitude and x and y position). The rotational subsystem is fully actuated while the translational subsystem is underactuated [32].

Model Propositions and Hypotheses: The quadrotor in fact experiences variety of physical effects. However, initially, it is necessary to make assumptions in order to eliminate those of lower significance dynamics.

- The quadrotor structure is assumed a rigid.
- The quadcopter has a perfectly symmetrical structure, which induces that the inertial matrix will be assumed diagonal.
- Thrust/drag of each motor are proportional to the square of the speed of rotation of the rotors, which is a very close approximation of the aerodynamic behavior.
- The body center axis is in the same position as the center of gravity of quadrotor.
- The inertia of rotors been neglected counter to all gyroscopic effect due to the rotors propulsion.

To evaluate the mathematical model of the quadrotor, we use two reference frames, a fixed frame linked to the earth R^b and a moving frame R^m . The transition between the moving and fixed frame is given by a matrix called the transformation matrix T, which contains the orientation and position of the moving frame relative to the fixed frame.

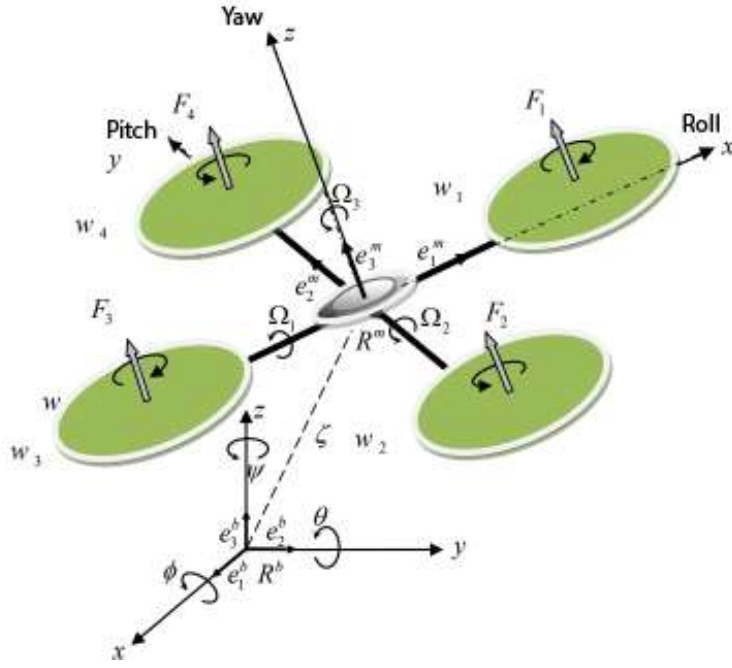


Figure 2.7. Geometry of the quadrotor.

$$T = \begin{bmatrix} R & \zeta \\ 0 & 1 \end{bmatrix} \quad (2.1)$$

With R the rotation matrix (describes the orientation of the moving object), $\zeta = [x \ y \ z]^T$ is the position vector. Euler angles are used to determine the elements of the rotation matrix R .

2.3.1. Euler Angles

At the beginning, the moving frame is aligned with the fixed frame, then the moving frame rotates around the x -axis by a roll angle $(-\frac{\pi}{2} < \phi < \frac{\pi}{2})$, followed by a rotation around the y -axis by a pitch angle $(-\frac{\pi}{2} < \theta < \frac{\pi}{2})$, followed by a rotation around the z -axis by a yaw angle $(-\pi < \Psi < \pi)$, so we have the rotation matrix R as following:

$$\begin{aligned} R &= Rot_z(\Psi) \times Rot_y(\theta) \times Rot_x(\phi) \\ &= \begin{bmatrix} C\Psi & -S\Psi & 0 \\ S\Psi & C\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\phi & -S\phi \\ 0 & S\phi & C\phi \end{bmatrix} \end{aligned} \quad (2.2)$$

$$R = \begin{bmatrix} C\Psi C\theta & S\phi S\theta - S\Psi C\phi & C\phi S\theta C\Psi + S\Psi S\phi \\ S\Psi C\theta & S\phi S\theta S\Psi + C\Psi C\theta & C\phi S\theta S\Psi - S\phi C\Psi \\ -S\theta & S\phi C\theta & C\phi C\theta \end{bmatrix} \quad (2.3)$$

With: $C = \cos$ and $S = \sin$

2.3.2. Angular velocity

We define Ω_1, Ω_2 and Ω_3 as angular velocities in the fixed frame. They're expressed in terms of the velocities $\dot{\phi}, \dot{\theta}$ and $\dot{\psi}$ in the moving frame. The roll rotation takes place when the frames are aligned, followed by a pitch rotation, the speed vector must be expressed in the fixed frame, so we multiply the vector $\dot{\theta}$ by $Rot_x(\phi)^{-1}$, and finally we have the yaw rotation and to express the velocity vector $\dot{\psi}$ in the fixed frame we multiply it by $[Rot_y(\theta) \cdot Rot_x(\phi)]^{-1}$

We therefore have:

$$\Omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + Rot_x(\phi)^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + [Rot_y(\theta) \cdot Rot_x(\phi)]^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \quad (2.4)$$

Finally, we obtain:

$$\Omega = \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\theta} C\phi \\ -\dot{\theta} S\phi \end{bmatrix} + \begin{bmatrix} -\dot{\psi} S\theta \\ \dot{\psi} S\phi C\theta \\ \dot{\psi} C\phi C\theta \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \dot{\psi} S\theta \\ \dot{\theta} C\phi + \dot{\psi} S\phi C\theta \\ \dot{\psi} C\phi C\theta - \dot{\theta} S\phi \end{bmatrix} \quad (2.5)$$

$$\Omega = \begin{bmatrix} 1 & 0 & -S\theta \\ 0 & C\phi & S\phi C\theta \\ 0 & -S\phi & C\phi C\theta \end{bmatrix} \times \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (2.6)$$

When the quadrotor makes small rotations, we can make the following approximations:

$$C\phi = C\theta = C\psi = 1, \text{ and } S\phi = S\theta = S\psi = 0.$$

So the angular velocity will be

$$\Omega = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T \quad (2.7)$$

2.3.3. Linear velocity

Linear velocities v_x^b, v_y^b, v_z^b in the fixed frame according to linear velocities v_x^m, v_y^m, v_z^m in the moving frame are given by:

$$v = \begin{bmatrix} v_x^b \\ v_y^b \\ v_z^b \end{bmatrix} = R \times \begin{bmatrix} v_x^m \\ v_y^m \\ v_z^m \end{bmatrix} \quad (2.8)$$

2.3.4. Development of the Newton-Euler mathematical model [25-26-27-28]

Using the Newton-Euler formulation, the equations are expressed as follows:

$$\begin{cases} \dot{\zeta} = v \\ m\ddot{\zeta} = F_f + F_t + F_g \\ \dot{R} = RS(\Omega) \\ J\dot{\Omega} = -\Omega \wedge J\Omega + M_f - M_a - M_{gh} \end{cases} \quad (2.9)$$

With ζ : is the position vector of the quadrotor

Ω : Angular velocity expressed in the fixed frame

m : Total mass of the quadrotor

R : Rotation matrix

\wedge : The vector product

J : Symmetrical inertia matrix of dimension (3x3), given by:

$$J = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad (2.10)$$

$S(\Omega)$: is the antisymmetric matrix; for a velocity vector it is given by:

$$S(\Omega) = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \quad (2.11)$$

F_f : is the total force generated by the four rotors, given by:

$$F_f = R \times \left[0 \quad 0 \quad \sum_{i=1}^4 F_i \right]^T \quad (2.12)$$

$$F_i = b\omega_i^2 \quad (2.13)$$

F_t : The drag force along the axes (x, y, z), given by:

$$F_t = \begin{bmatrix} -K_{ftx} & 0 & 0 \\ 0 & -K_{f ty} & 0 \\ 0 & 0 & -K_{ftz} \end{bmatrix} \dot{\zeta} \quad (2.14)$$

K_{ftx} , $K_{f ty}$ and K_{ftz} : Drag coefficients for translation

F_g : force of gravity is given by:

$$F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \quad (2.15)$$

M_f : moment caused by thrust and drag forces

$$M_f = \begin{bmatrix} l (F_4 - F_2) \\ l (F_3 - F_1) \\ d (\omega^2_1 - \omega^2_2 + \omega^2_3 - \omega^2_4) \end{bmatrix} \quad (2.16)$$

M_a : moment resulting from aerodynamic friction, it is given by:

$$M_a = \begin{bmatrix} K_{fax} \dot{\phi}^2 \\ K_{fay} \dot{\theta}^2 \\ K_{faz} \dot{\psi}^2 \end{bmatrix} \quad (2.17)$$

$K_{fax}, K_{fay}, K_{faz}$: Aerodynamic friction coefficients.

2.3.4.1. Equations of translation movement

We have:

$$m\ddot{\zeta} = F_f + F_t + F_g \quad (2.18)$$

If we replace each force by its formula, we find:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} c\phi s\psi s\theta + s\phi s\psi \\ c\phi s\theta s\psi - s\phi s\psi \\ c\phi c\theta \end{bmatrix} \sum_{i=1}^4 F_i - \begin{bmatrix} K_{ftx}\dot{x} \\ K_{fty}\dot{y} \\ K_{ftz}\dot{z} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \quad (2.19)$$

We then obtain the differential equations that define the translational movement:

$$\begin{cases} \ddot{x} = \frac{1}{m} (c\phi s\theta s\psi + s\phi s\psi) \left(\sum_{i=1}^4 F_i \right) - \frac{K_{ftx}}{m} \dot{x} \\ \ddot{y} = \frac{1}{m} (c\phi s\theta s\psi - s\phi s\psi) \left(\sum_{i=1}^4 F_i \right) - \frac{K_{fty}}{m} \dot{y} \\ \ddot{z} = \frac{1}{m} (c\phi c\theta) \left(\sum_{i=1}^4 F_i \right) - \frac{K_{ftz}}{m} \dot{z} - g \end{cases} \quad (2.20)$$

2.3.4.2. Equations of rotational movement

We have:

$$J\dot{\Omega} = -\Omega \wedge J\Omega + M_f - M_a - M_{gh} \quad (2.21)$$

If we replace each moment by the corresponding formula, we find:

$$\begin{aligned} \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\Psi} \end{bmatrix} = - \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\Psi} \end{bmatrix} \wedge \left(\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\Psi} \end{bmatrix} \right) - \begin{bmatrix} J_r \overline{\Omega_r} \dot{\theta} \\ -J_r \overline{\Omega_r} \dot{\phi} \\ 0 \end{bmatrix} \\ - \begin{bmatrix} K_{fax} \dot{\phi}^2 \\ K_{fay} \dot{\theta}^2 \\ K_{faz} \dot{\Psi}^2 \end{bmatrix} + \begin{bmatrix} lb(\omega_4^2 - \omega_2^2) \\ lb(\omega_3^2 - \omega_1^2) \\ d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \end{aligned} \quad (2.22)$$

We then obtain the differential equations defining the rotational motion:

$$\begin{cases} I_x \ddot{\phi} = -\dot{\theta} \dot{\Psi} (I_z - I_y) - J_r \overline{\Omega_r} \dot{\theta} - K_{fax} \dot{\phi}^2 + lb(\omega_4^2 - \omega_2^2) \\ I_y \ddot{\theta} = \dot{\phi} \dot{\Psi} (I_z - I_x) - J_r \overline{\Omega_r} \dot{\phi} - K_{fay} \dot{\theta}^2 + lb(\omega_3^2 - \omega_1^2) \\ I_z \ddot{\Psi} = -\dot{\phi} \dot{\theta} (I_y - I_x) - J_r \overline{\Omega_r} \dot{\theta} - K_{faz} \dot{\Psi}^2 + d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \quad (2.23)$$

With:

$$\overline{\Omega_r} = \omega_1 - \omega_2 + \omega_3 - \omega_4 \quad (2.24)$$

Consequently, the complete dynamic model governing the quadrotor is as follows:

$$\begin{cases} \ddot{\phi} = \frac{(I_y - I_z)}{I_x} \dot{\theta} \dot{\Psi} - \frac{J_r}{I_x} \overline{\Omega_r} \dot{\theta} - \frac{K_{fax}}{I_x} \dot{\phi}^2 + \frac{l}{I_x} u_2 \\ \ddot{\theta} = \frac{(I_z - I_x)}{I_y} \dot{\phi} \dot{\Psi} + \frac{J_r}{I_y} \overline{\Omega_r} \dot{\phi} - \frac{K_{fay}}{I_y} \dot{\theta}^2 + \frac{l}{I_y} u_3 \\ \ddot{\Psi} = \frac{(I_x - I_y)}{I_z} \dot{\theta} \dot{\phi} - \frac{K_{faz}}{I_z} \dot{\Psi}^2 + \frac{1}{I_z} u_4 \\ \ddot{x} = -\frac{K_{ftx}}{m} \dot{x} + \frac{1}{m} u_x u_1 \\ \ddot{y} = -\frac{K_{fty}}{m} \dot{y} + \frac{1}{m} u_y u_1 \\ \ddot{z} = -\frac{K_{ftz}}{m} \dot{z} - g + \frac{\cos(\phi) \cos(\theta)}{m} u_1 \end{cases} \quad (2.25)$$

With :

$$\begin{cases} u_x = (c\phi c\Psi s\theta + s\theta s\Psi) \\ u_y = (c\phi s\theta s\Psi - s\phi c\Psi) \end{cases} \quad (2.26)$$

And :

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ -lb & 0 & lb & 0 \\ d & -d & d & -d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_3^2 \end{bmatrix} \quad (2.27)$$

from (2.26) we find :

$$\begin{cases} \phi_d = \arcsin (u_x \sin(\Psi_d) - u_y \cos(\Psi_d)) \\ \theta_d = \arcsin \left(\frac{(u_x \sin(\Psi_d) - u_y \cos(\Psi_d))}{\cos(\phi_d)} \right) \end{cases} \quad (2.28)$$

○ **Control Input Vector U [34]:**

A control input vector, u , consisting of four inputs; u_1 through u_4 , u_1 is the resulting upwards force of the four rotors which is responsible for the altitude of the quadrotor and its rate of change (z, \dot{z}). u_2 is the difference in thrust between rotors 2 and 4 which is responsible for the roll rotation and its rate of change ($\phi, \dot{\phi}$). u_3 on the other hand represents the difference in thrust between rotors 1 and 3 thus generating the pitch rotation and its rate of change ($\theta, \dot{\theta}$). Finally u_4 is the difference in torque between the two clockwise turning rotors and the two counterclockwise turning rotors generating the yaw rotation and ultimately its rate of change ($\Psi, \dot{\Psi}$). This choice of the control vector u decouples the rotational system, where u_1 will generate the desired altitude of the quadrotor, u_2 will generate the desired roll angle, the desired pitch angle will be generated by u_3 whereas u_4 will generate the desired heading.

If the rotor velocities are needed to be calculated from the control inputs, an inverse relationship between the control inputs and the rotors' velocities is needed, which can be acquired by inverting the matrix in Equation (2.27) then taking the square root of that.

2.3.5. System state representation

For a physical system, there are many state representations, in our case, we choose the state vector as follows [25-26-27-28]:

$$\begin{aligned} X &= [\phi \quad \dot{\phi} \quad \theta \quad \dot{\theta} \quad \Psi \quad \dot{\Psi} \quad x \quad \dot{x} \quad y \quad \dot{y} \quad z \quad \dot{z}]^T \\ &= [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_7 \quad x_9 \quad x_{10} \quad x_{11} \quad x_{12}]^T \end{aligned}$$

We obtain the following state representation:

$$\left\{ \begin{array}{l} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1 x_4 x_6 + a_2 x_2^2 + a_3 \bar{\Omega}_r x_4 + b_1 u_2 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = a_1 x_4 x_6 + a_2 x_2^2 + a_3 \bar{\Omega}_r x_4 + b_1 u_2 \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = a_7 x_2 x_4 + a_8 x_6^2 + b_3 u_4 \\ \dot{x}_7 = x_8 \\ \dot{x}_8 = a_9 x_8 + \frac{1}{m} u_x u_1 \\ \dot{x}_9 = x_{10} \\ \dot{x}_{10} = a_{10} x_{10} + \frac{1}{m} u_y u_1 \\ \dot{x}_{11} = x_{12} \\ \dot{x}_{12} = a_{11} x_{12} + \frac{\cos(\phi) \sin(\theta)}{m} u_1 - g \end{array} \right. \quad (2.29)$$

With:

$$\left\{ \begin{array}{l} a_1 = \frac{(I_y - I_z)}{I_x}, a_2 = -\frac{K_{fax}}{I_x}, a_3 = -\frac{J_r}{I_x}, a_4 = \frac{(I_z - I_x)}{I_y}, a_5 = -\frac{K_{fay}}{I_y}, a_6 = \frac{J_r}{I_y} \\ a_7 = \frac{(I_x - I_y)}{I_z}, a_8 = -\frac{K_{faz}}{I_z}, a_9 = -\frac{K_{ftx}}{m}, a_{10} = -\frac{K_{f ty}}{m}, a_{11} = -\frac{K_{ftz}}{m}, b_1 = \frac{l}{I_x}, \\ b_2 = \frac{l}{I_y}, b_3 = \frac{l}{I_z}. \end{array} \right. \quad (2.30)$$

2.2.6. Rotor dynamics [30-31]

The motors used in quadrotors are generally DC motors. Rotor dynamics are approximated to the dynamics of a DC motor, and is given by the following differential equations:

$$J_r \dot{\omega}_i = \tau_i - Q_i, i \in \{1,2,3,4\} \quad (2.31)$$

Where τ_i is the input torque, and $Q_i = d\omega_i^2$ is the resistive torque generated by rotor i .

To achieve the objectives of quadrotor control, a speed control loop is often required. First, we need to determine the desired speeds $\omega_{d,i}$ corresponding to the command values provided by the controller, calculated as follows:

$$\varpi_d = M^{-1}U \quad (2.32)$$

Where : $\varpi_d = (\omega_{d1}^2, \omega_{d2}^2, \omega_{d3}^2, \omega_{d4}^2)^T$, $U=(u_1, u_2, u_3, u_4)^T$, and M is a non-singular matrix, obtained from (2.27).

The aim is to synthesize a controller so that $\omega_i \rightarrow \omega_{d,i}$ when $t \rightarrow \infty$ using torques τ_i .

We define the speed error:

$$\tilde{\omega}_i = \omega_i - \omega_{d,i} \quad (2.33)$$

A control law is developed in [30-31], it is given by:

$$\tau_i = Q_i + J_r \dot{\omega}_{d,i} - k_i \tilde{\omega}_i \quad (2.34)$$

With $k_i, i \in \{1,2,3,4\}$ are positive gains.

Replacing the control law in (2.31), we obtain:

$$\dot{\tilde{\omega}}_i = -\frac{k_i}{J_r} \tilde{\omega}_i \quad (2.35)$$

This relationship represents the dynamics of the error, showing us the exponential convergence of ω_i to $\omega_{d,i}$, when $t \rightarrow \infty$. This indicates the convergence of the quadrotor towards these desired values, ensuring the quadrotor's stability.

In reality, the quadrotor is controlled by the power supply voltages of its four motors. To regulate these motors, it is necessary to acquire the input voltage for each individual motor. Assuming a low motor inductance and utilizing identical motors, the input voltage for each motor can be derived through the following method:

$$v_i = \frac{R_a}{k_m k_g} \tau_i + k_m k_g \omega_i \quad (2.36)$$

With: R_a is the motor resistance, k_m is the motor torque constant, k_g is the gearbox gain.

Conclusion

In conclusion, this chapter has provided a comprehensive exploration of the dynamic model of quadrotors, we started with a general description, gaining insights into their structure and functionality. We then delved into the complexities of quadrotor movements, elucidating how they achieve various flight behaviors through precise control of rotor dynamics.

The application of the Newton-Euler formalism has enabled us to establish the dynamic model of the quadrotor. From this model, we draw the conclusion that the quadrotor is a bit challenging to control because it doesn't have enough actuators for all its movements. Moreover, the complexity of this model, its non-linearity, and the complex interactions among its various states are easily noticeable.

In the forthcoming chapter, we will unveil some control techniques that promise to navigate the challenges posed by the quadrotor's dynamic nature, providing effective strategies for precision control, maneuverability and robustness.

Chapter 3

Controller design and simulation results

Introduction

In the preceding chapter, we modeled dynamically our quadrotor from physical and aerodynamic laws using the Euler-Newton formalization. This chapter establishes the control of a quadrotor using both linear and non-linear methods.

In the first place, we'll demonstrate how to perform a PID-corrected loop system control. Next, we'll go into more details about how the sliding mode and other nonlinear controllers are developed.

At the end, a simulation of the dynamics will be displayed using the Matlab and Simulink tools.

3.1. Synthesis of quadrotor control laws

In this section, we propose a control strategy mainly based on two loops (inner loop and outer loop). The inner loop contains four control laws: roll control (ϕ), pitch control (θ), yaw control (ψ) and Z altitude control. The outer loop includes two control laws for positions x and y. The outer control loop uses a correction block to generate desired values for roll (θ_d) and pitch (ϕ_d) motions. This block corrects roll and pitch rotation based on the desired yaw (ψ_d). The diagram below shows the control strategy adopted:

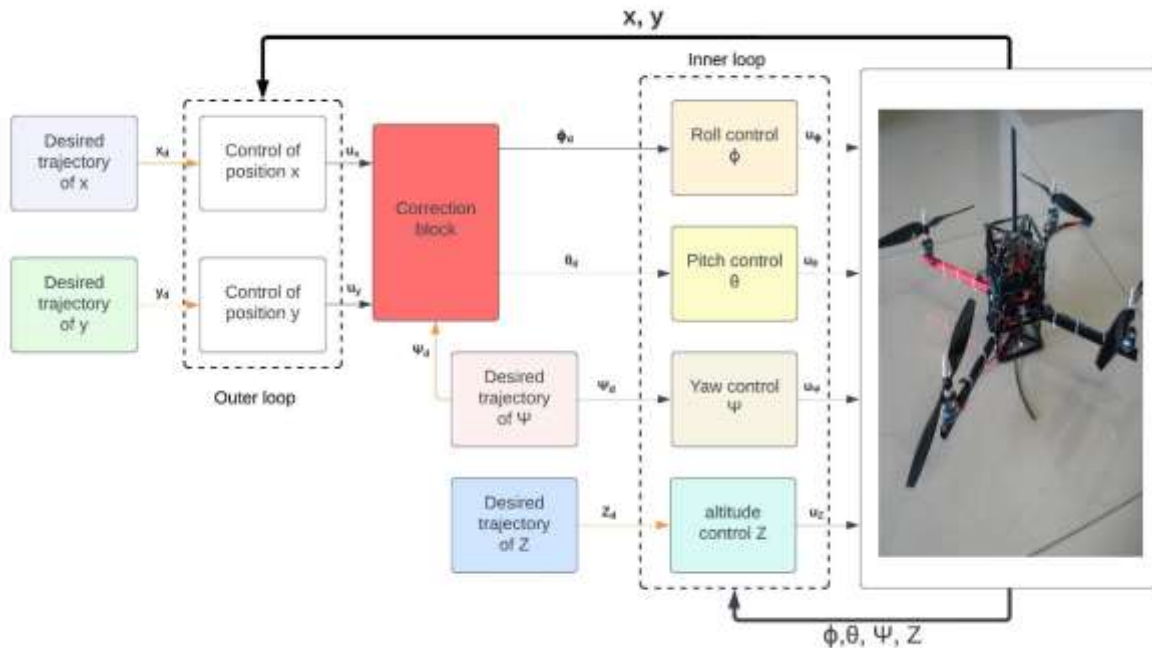


Figure 3.1. Illustration of quadrotor control structure.

3.2. Linear control for quadrotors

3.2.1. Classical PID control [33]

In the process industries, the PID algorithm is the most widely utilized feedback controller. For more than 60 years, it has been used successfully. It is a reliable, simple-to-understand method that can produce outstanding gains in servo performance dynamic features.

PID control, also known as proportional-integral-derivative control, is a common technique in the design of control systems. It is a form of feedback control that modifies an output variable according to the discrepancy between the desired set point and the actual measured value.

To calculate the control signal, the PID controller combines three terms: **proportional**, **integral**, and **derivative**. Each word makes a unique contribution to the total control action.

Prior to specifying the parameters (or settings) for each mode that will be utilized in this method, it is required to choose between utilizing P, I, or D as the mode to start with.

P, PI, or PID are three commonly used fundamental algorithms. The classic PID structure is shown in **Figure 3.2**:

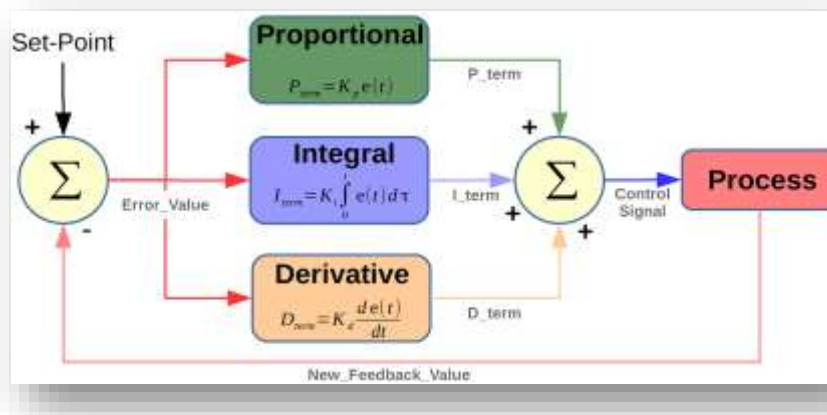


Figure 3.2. Classical PID structure.

3.2.2. Characteristics of P, I, and D controllers

- A **proportional controller (K_p)** will shorten the rising time but never completely eliminate steady-state inaccuracy.

The effect of proportional action is to amplify the error by a constant gain so that the system reacts more quickly to set-point changes. This proportional action is represented as follows:

$$C(t) = K_p \cdot \varepsilon(t) \tag{3.1}$$

The greater the value of K_p , the faster the response, but at the expense of a deterioration in system stability to instability for larger values.

- **An integral controller (K_i)** will remove steady-state error but may exacerbate transient response. The aim of integral action is to reduce or even eliminate the static error in steady state. To achieve this, the controller integrates the error with respect to time and multiplies the result by a constant K_i as follows:

$$C(t) = K_i \cdot \int_0^t \varepsilon(t) dt \tag{3.2}$$

The larger the value of K_i , the faster the static error will be corrected, but we lose a little in stability and there is a risk of overshoot occurring.

- **Derivative control (K_d)** enhances transient response, lowers overshoot, and overshoot while increasing system stability. To obtain a derivative action, we multiply the derivative of the error by a coefficient K_d this action eliminates response overshoot and improves system stability. Its relationship is given as follows:

$$C(t) = K_d \cdot \frac{d\varepsilon(t)}{dt} \tag{3.3}$$

The greater the value of K_d the more the overshoot is attenuated, but if it is too great the system is slowed down until it risks becoming unstable for very large values.

The table below summarizes how each of the controllers K_p , K_i , and K_d affects a closed-loop system.

Tableau 3-1. K_p , K_i and K_d controllers.

Response	Rise time	Over shoot	Setteling time	Static error
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Increase	Increase	Elimination
K_d	Small change	Decrease	Decrease	Small change

Due to the dependence between K_p , K_i , and K_d , these correlations may not be exact.

In fact, altering one of these factors could affect how the other two behave. The table should only be used as a reference when figuring out the values of K_p , K_i , and K_d .

For speed, stability, and precision, the PID controller integrates the three mentioned actions. This is the given expression for a PID controller:

$$C(t) = K_p \cdot \varepsilon(t) + K_i \cdot \int_0^t \varepsilon(t) dt + K_d \cdot \frac{d\varepsilon(t)}{dt} \quad (3.4)$$

In this way, the controller achieves its objectives in terms of speed, stability and precision by finding the optimum configuration of the values of the various gains K_p , K_i , and K_d .

3.2.3. Classic PID quadrotor control

The quadrotor is a sub-actuated system with 6 DOF having 4 inputs to control 6 outputs. The system is connected to two loops:

- **Inner loop:** that regulates rotational motions (roll, pitch, and yaw).
- **Outer loop:** that regulates translational movements.

The inner loop must be quicker.

The PID controller was chosen due to its:

- **Simple structure**
- **Good performance in many processes**
- **Reliable even without a specific control system model.**

So this system is made up of 6 PIDs, 3 regulators of rotational movements in the inner loop and 3 in the outer loop,

The synoptic model of the quadrotor is shown in **Figure 3.3**:

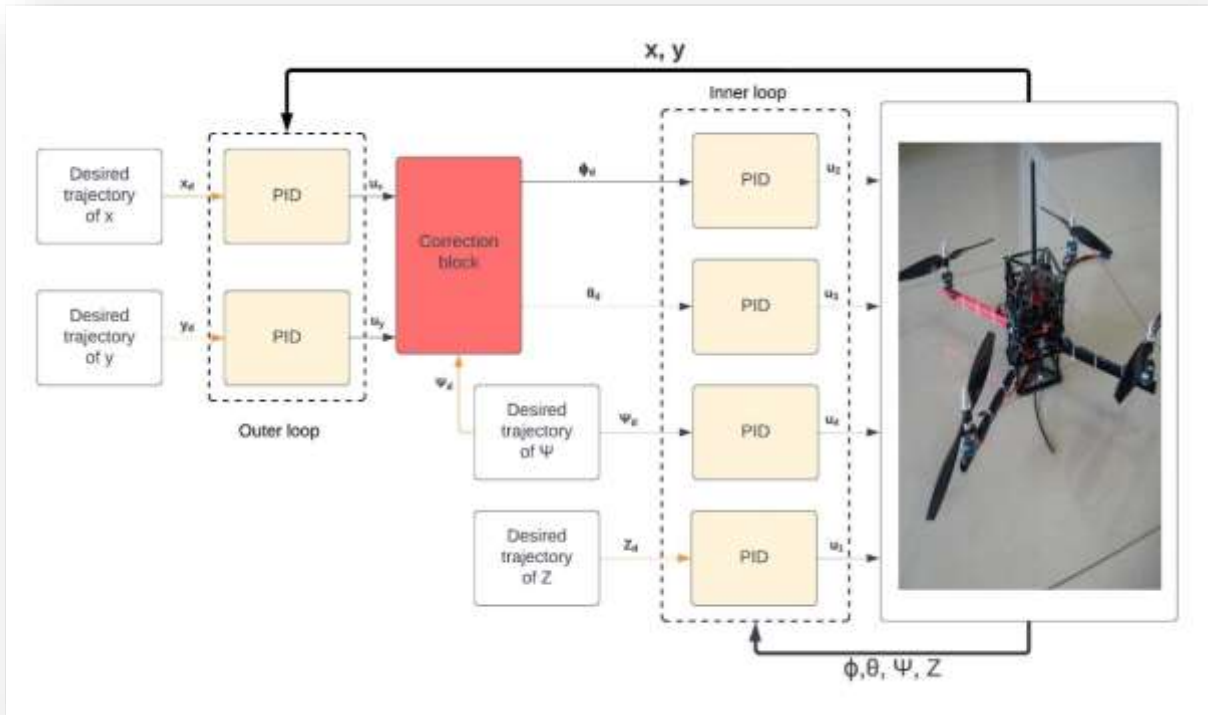


Figure 3.3. Quadrotor PID control diagram.

The dynamic model presented in chapter 2 includes gyroscopic effects. The influence of effects in our case is less important than the action of the motors. We can neglect these effects by considering a quasi-steady state. A model approximation is rewritten as follows:

$$\left\{ \begin{array}{l} \ddot{\phi} = \frac{l}{I_{xx}} U_2 \\ \ddot{\theta} = \frac{l}{I_{yy}} U_3 \\ \ddot{\psi} = \frac{1}{I_{zz}} U_4 \\ \ddot{X} = \frac{u_x}{m} U_1 \\ \ddot{Y} = \frac{u_y}{m} U_1 \\ \ddot{Z} = g - \frac{1}{m} \cos(\theta) \cos(\phi) \end{array} \right. \quad (3.5)$$

In our case we want our controller's outputs to follow the position trajectory X, Y, Z , therefore the general expression of the classical PID structure is written as follows:

$$U_i = K_{p-j} \cdot e_j(t) + K_{I-j} \cdot \int e_j(\tau) d\tau + K_{D-j} \cdot \frac{de_j(t)}{dt} \quad (3.6)$$

With $i = 1, 2, 3, 4$ and $j = Z, \phi, \theta, \Psi$

The correction is based on the observed error, which is the difference between the desired value and the actual measured value. $e_j = \text{desired value} - \text{measured value} = j_d - j$

The two controllers for X, Y to generate ϕ_d, θ_d are developed as follows:

$$U_{\phi_d, \theta_d} = K_{p-X,Y} \cdot e_{X,Y}(t) + K_{I-X,Y} \cdot \int e_{X,Y}(\tau) d\tau + K_{D-X,Y} \cdot \frac{de_{X,Y}(t)}{dt} \quad (3.7)$$

$$e_{X,Y} = X, Y_d - X, Y \quad (3.8)$$

3.2.3.1. Altitude, Attitude and Heading Control

A PID controller is developed to control the altitude z , the roll angle ϕ , the pitch angle θ and the yaw Ψ of the quadrotor. It generates the control input U_1, U_2, U_3, U_4 successively. The derived control laws are as follow:

$$\begin{cases} U_1 = K_p(z - z_d) + K_d(\dot{z} - \dot{z}_d) + K_i \int (z - z_d) dt \\ U_2 = K_p(\phi_d - \phi) + K_d(\dot{\phi}_d - \dot{\phi}) + K_i \int (\phi_d - \phi) dt \\ U_3 = K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta}) + K_i \int (\theta_d - \theta) dt \\ U_4 = K_p(\Psi_d - \Psi) + K_d(\dot{\Psi}_d - \dot{\Psi}) + K_i \int (\Psi_d - \Psi) dt \end{cases} \quad (3.9)$$

3.2.3.2. Position Controller

A complete position controller is created after stable controllers are acquired for the quadrotor's altitude and attitude. The desired accelerations \ddot{x}_d and \ddot{y}_d are calculated using PID controllers.

$$\ddot{x}_d = K_p(x_d - x) + K_d(\dot{x}_d - \dot{x}) + K_i \int (x_d - x) dt \quad (3.10)$$

$$\ddot{y}_d = K_p(y_d - y) + K_d(\dot{y}_d - \dot{y}) + K_i \int (y_d - y) dt \quad (3.11)$$

3.2.4. PID Controller Simulation

In order to test the correctors on hard dynamics, we have identified the parameters listed in the following table:

Tableau 3-2. Parameters obtained through identification.

	Parameter	description	Unit	value
01	m	Mass of the Quadrotor	Kg	0.480
02	g	Gravitational acceleration	m/s^2	9.806
03	L	Distance to the center of the Quadrotor	M	0.25
04	b	Thrust factor	$N.sec^2$	2.9842×10^{-5}
05	d	Drag factor	$N.m.sec^2$	3.2320×10^{-7}
06	I_{xx}	Quadrotor moment of inertia around X axis	$Kg.m^2$	8.1×10^{-3}
07	I_{yy}	Quadrotor moment of inertia around Y axis	$Kg.m^2$	8.1×10^{-3}
08	I_{zz}	Quadrotor moment of inertia around Z axis	$Kg.m^2$	14.2×10^{-3}
09	J_r	Rotor inertia	$Kg.m^2$	6×10^{-5}

We have succeeded in stabilizing the trajectories of the quadrotor model thanks to the PID parameters adjusted by attentives. These values are summarized in the following table:

Tableau 3-3. PID Parameters.

PID Parameter	X	Y	Z	ϕ	θ	ψ
K_p	2.2	2.2	5	2	2	2.3
K_i	0.02	0.02	2.5	1.1	1.2	0.01
K_d	0.85	0.85	2	2	2	1.1

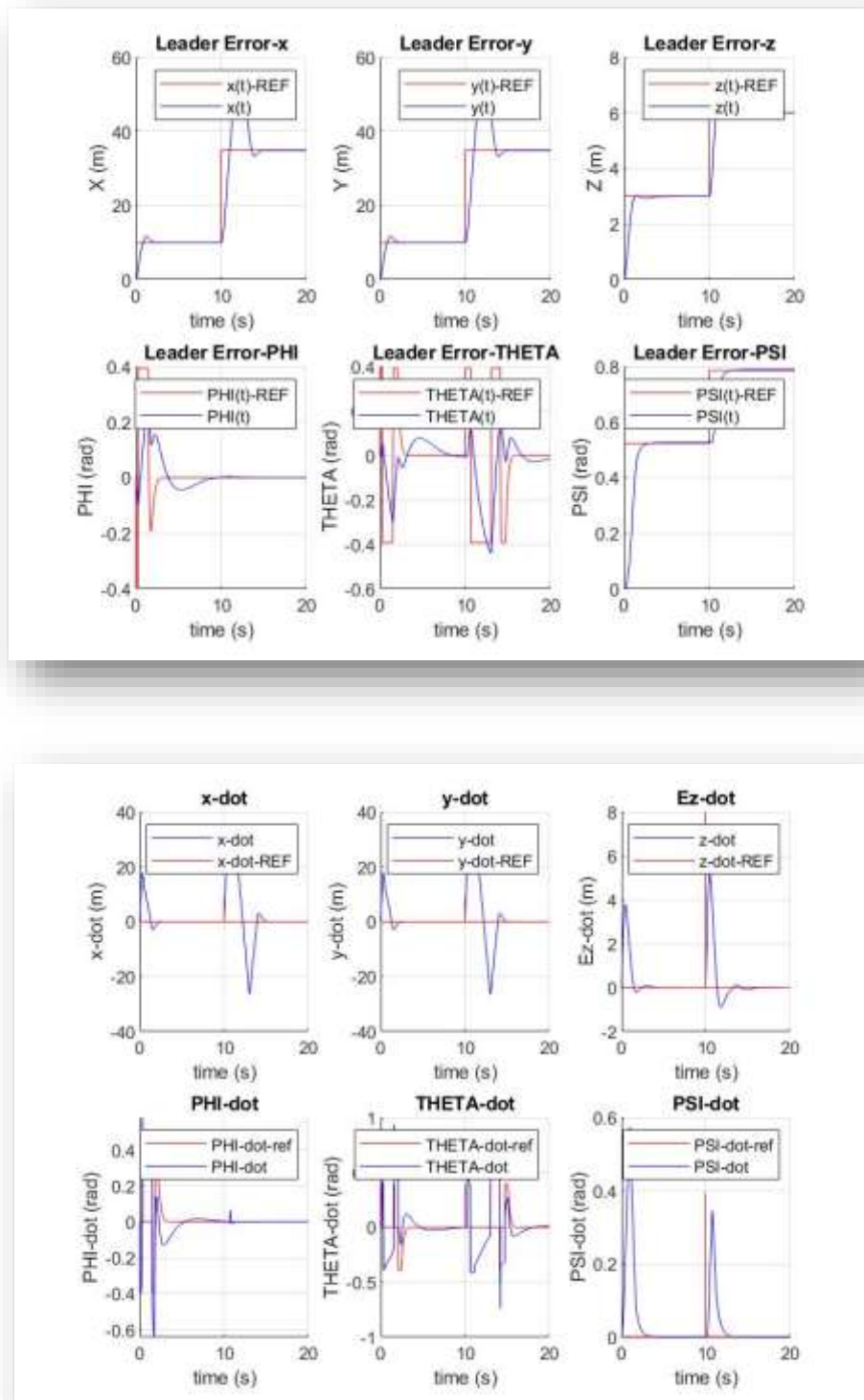


Figure 3.4. PID Controller Simulation Response.

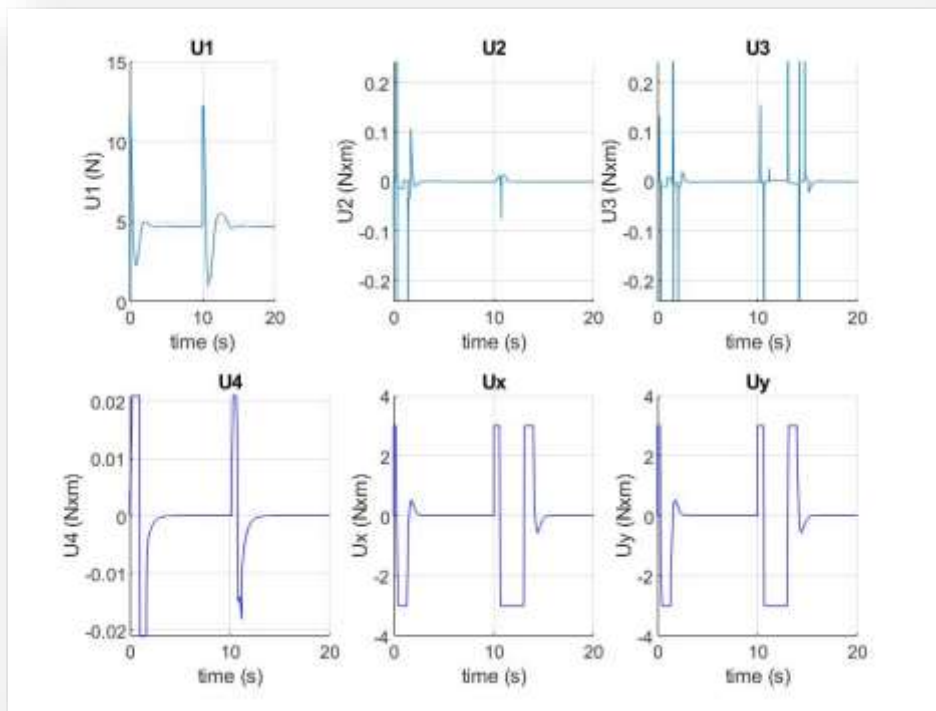


Figure 3.5. PID Control Inputs.

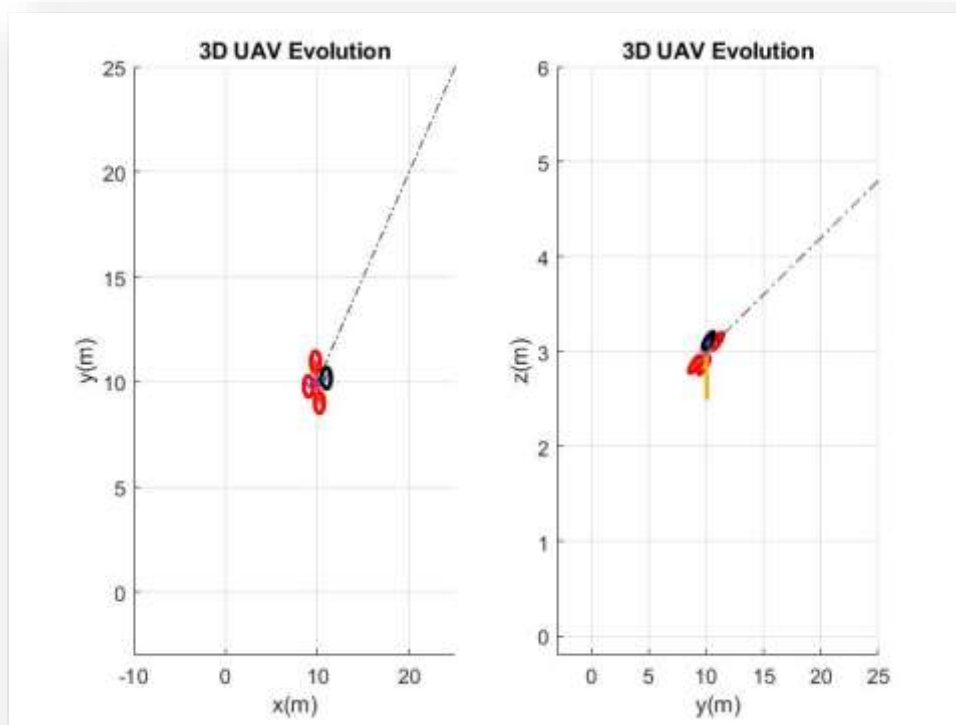


Figure 3.6. Trajectory Response under PID.

These graphs provide a comprehensive view of the quadrotor's performance under PID control. Figure 3.7 compares the actual trajectory with the desired reference trajectory in six dimensions. Figure 3.5 displays the derivatives of position and orientation parameters, reflecting the quadrotor's velocity and angular velocity. Figure 3.6 showcases the control inputs ($U_1, U_2, U_3, U_4, U_x,$ and U_y) that the PID controller applies to maintain stability, altitude, and precise positioning of the quadrotor. Together, they offer insights into the controller's effectiveness and dynamic behavior.

3.3. Non-linear control for quadrotors

Since the quadrotor system is a nonlinear type system, traditional correctors become insufficient and often give less efficient results. To overcome this problem, the current research trend is towards robust nonlinear controls which give acceptable results over wide functional ranges. Among these techniques are sliding mode control SMC, ADRC and Non-linear PID, which have long been the subject of a great deal of research, either on their own or in combination with other control techniques.

3.3.1. Sliding Mode Controller (SMC)

We proposed using a nonlinear Sliding Mode Controller (SMC) to control the states of the quadrotor.

3.3.1.1. Introduction to SMC

The theory of sliding modes has been the subject of over the last thirty years, both by Soviet researchers as well as by researchers in other countries. It was developed by Russian engineer Emelyanov in the 1950s and improved upon by American engineer Utkin in the 1970s, it was originally created for the guidance and control of missiles.

SMC has become well known for its capacity to manage uncertainties and disturbances, making it appropriate for real-world uses. The concept of sliding mode is forcing the system's state trajectory to slide along a defined surface. Designing a discontinuous control law that switches the system's dynamics and ensures that the system stays on the sliding surface despite uncertainty is one of the key principles of SMC. SMC has become widely used throughout time in many industries, including robotics, aircraft, and power systems [17].

Sliding-mode control is a special type of Variable Structure Systems (VSS). In sliding-mode control, the control system is designed to drive and then constrain the system state to lie in a proximity of the switching function. There are two advantages to this approach.

Firstly, the dynamic behaviour of the system can be adapted by a particular choice of the switching function. This implies that we can control the behaviour of the system by choosing when and how to switch between different control modes or strategies.

Secondly, the closed-loop response becomes totally insensitive to a particular class of uncertainty. This means, even when there are unknown factors influencing the system, the control system can handle uncertainties in a robust way, providing stable and reliable performance.

3.3.1.2. Principle of Sliding Mode Control (SMC)

Sliding control can be divided into two parts:

1. Attracting the system states: The goal is to bring the system's states into a predefined region, ensuring convergence towards this region regardless of initial conditions or disturbances.
2. Designing a control law: Once the system's states are within the desired region, a control law is implemented to keep them there. This control law continuously adjusts the system's inputs to counteract any deviations and maintain stability.

By combining these two parts, sliding control aims to stabilize and regulate the system's behavior, allowing it to operate within the defined region and meet specific performance requirements.

Notion:

The fundamental concept of sliding control is first to bring the system's states (such as position, velocity, etc.) into a carefully chosen region. The selection of this region is based on specific criteria or requirements for the system's behavior. Then once the system's states are within this region, a control law is designed to ensure that the system remains in this region continuously. In short, sliding mode control is divided into two parts:

$$u = u_{gliss} + u_{eq} \quad (3.12)$$

u_{gliss} : The Sliding is useful to compensate the uncertainties of the model. It consists of the “Sign” function of the sliding surface s , multiplied by a constant K_{gliss} .

The sliding surface is defined in the error state space to guarantee states convergence.

u_{eq} : Equivalent or nominal control is determined by the model of the system, a linear or a non-linear model, it is designed to drive the system dynamics onto the sliding surface or within a desired region. This part is designed using the equivalent control method, the principle of which is based on determining the system's behaviour when it is on the sliding surface s .

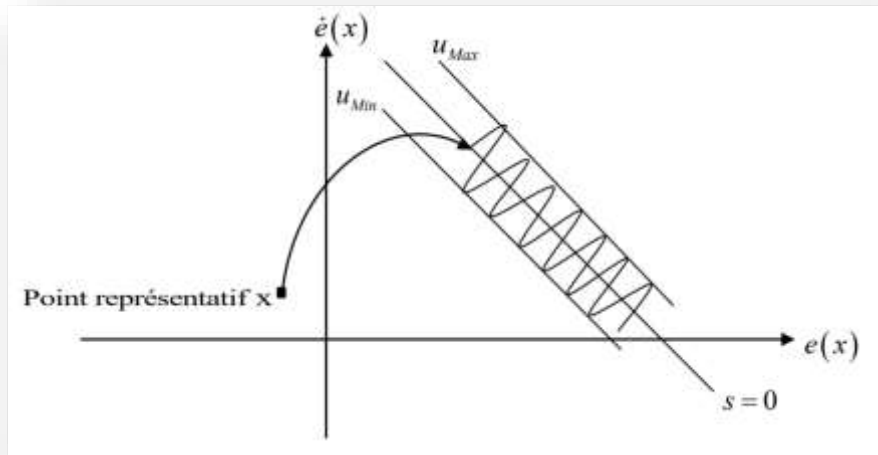


Figure 3.8. Sliding mode.

❖ **Sliding surface:**

In sliding mode control (SMC), a sliding surface is a key component of the control design. It is a trajectory or manifold that the system's state is driven towards and made to slide along, its design is based on the dynamics of the system and the control objectives. The control law is intended to make sure that the system maintains a sliding motion on the surface. In the presence of uncertainties or disturbances, this sliding behavior offers robustness and performance. Generally, the sliding surface $s(x)$ is selected as a hyperplan where the system dynamics change unexpectedly.

The surface s is given by:

$$s_i = \dot{e}_i + \lambda_i e_i \quad (3.13)$$

With:

$$e_i = x_i - x_{d_i} \quad (3.14)$$

Where: λ_i is the sliding surface parameter, x_i is the state of the system, and x_{d_i} is the desired state.

❖ **Switching Term:**

The switching term in SMC refers to the control action that actively moves the system's state onto the sliding surface. This control action switches between different control laws or modes of operation to achieve this goal.

- The switching term generates a control action to get the system's state back onto the sliding surface when it is far away.

- The switching term continuously modifies the control action while the system is on or close to the sliding surface to guarantee that it stays there.

❖ **Chattering in sliding mode control (SMC):**

Chattering is a common problem in sliding mode control (SMC). Chattering refers to high-frequency oscillations or rapid switching behavior observed in the control signals or system states while the system is in the sliding mode [18, 19, 20]. It appears as an undesired problem that might cause:

- Poor Performance
- Reduce Precision
- More Fatigue, Wear And Tear On The System's Components
- Produce A High energy loss

Chattering in SMC can be caused by several factors, for example for quadrotors; dynamics like motor response time, saturation limits, and delays can interact with the control algorithm and cause chattering, improper choice of control gains can also lead to chattering, for example gains that are too high, can make the control signal chatter and switch excessively between different control modes. Large or rapidly changing uncertainties or disturbances can cause the control signal to chatter. Several methods may be used to reduce chattering in SMC such as filters, mentioning that there isn't a specific solution that works for all quadrotor systems, however, the choice of these methods depends on the specific system characteristics and requirements.

3.3.1.3. Quadrotor sliding mode control

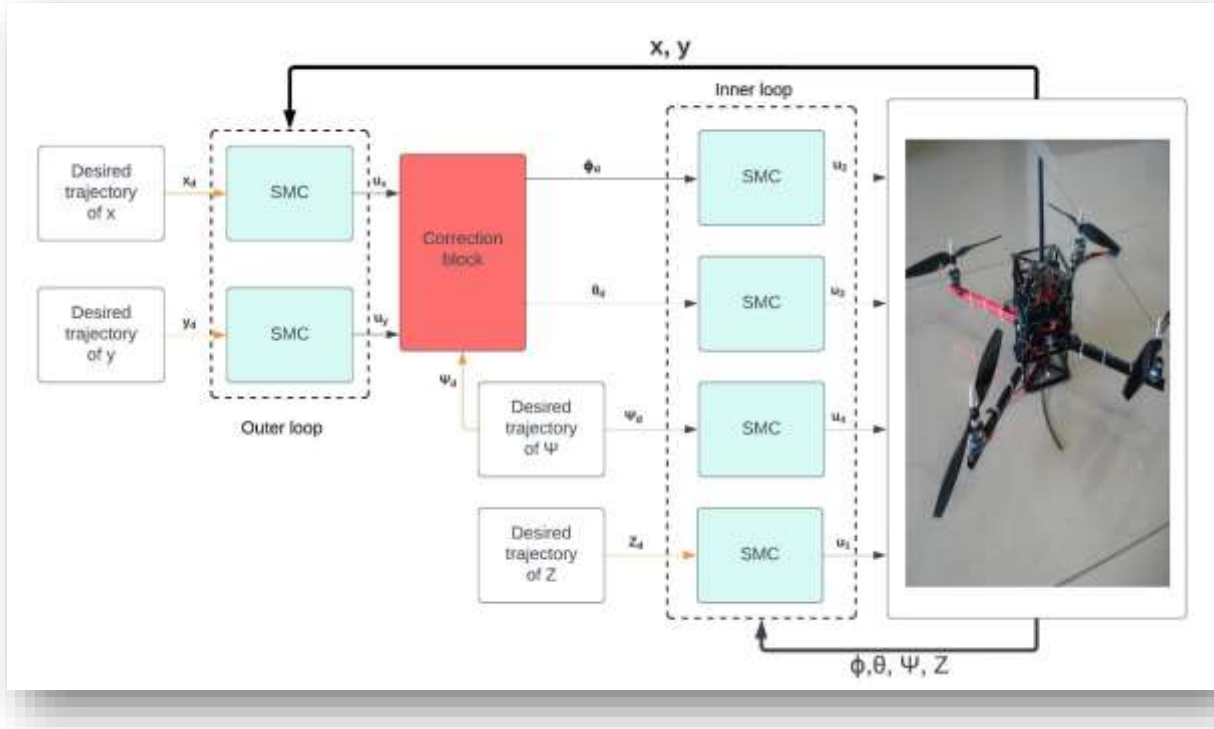


Figure 3.9. Quadrotor SMC control diagram.

We have:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1 x_4 x_6 + a_2 x_2^2 + a_3 \overline{\Omega}_r x_4 + b_1 u_2 \end{cases} \quad (3.15)$$

To design a control law for subsystem \$U_2\$.

➤ **Roll control \$\varphi\$:**

For subsystem (3.15), the relative degree is \$r = 2\$. The sliding surface then is defined as follows:

$$s_\varphi = \dot{e}_1 + \lambda_1 e_1 \quad (3.16)$$

So that:

$$\lambda_1 > 0, \text{ and } e_1 = \varphi_d - \varphi$$

Using sliding mode control theory, the control law is as follows:

$$u_2 = u_{2gliss} + u_{2eq} \quad (3.17)$$

With: \$u_{2gliss}\$ is the correction term defined by:

$$u_{2gliss} = -k_{s1} \text{sign}(s_\varphi), k_{s1} > 0 \quad (3.18)$$

u_{2eq} is the equivalent control, calculated when:

$$s_\varphi = 0, \text{ and } \dot{s}_\varphi = 0 \quad (3.19)$$

\dot{s}_φ is the derivative of s_φ , such that:

$$\dot{s}_\varphi = \ddot{e}_1 + \lambda_1 \dot{e}_1 \quad (3.20)$$

From (3.19) and (3.20), we have:

$$s_\varphi = 0 \Leftrightarrow \ddot{e}_1 + \lambda_1 \dot{e}_1 = 0 \quad (3.21)$$

By replacing \dot{e}_1 and \ddot{e}_1 in (3.21), we find:

$$\ddot{\varphi}_d - \dot{x}_2 + \lambda_1(\dot{\varphi}_d - x_2) = 0 \quad (3.22)$$

From (3.22), we get:

$$u_{2eq} = \frac{-1}{b_1} (a_1 x_4 x_6 + a_2 x_2^2 + a_3 \overline{\Omega_r} x_4 - \ddot{\varphi}_d - \lambda_1(\dot{\varphi}_d - x_2)) \quad (3.23)$$

Therefore:

$$u_2 = -k_{s1} \text{sign}(s_\varphi) + \frac{1}{b_1} (-a_1 x_4 x_6 - a_2 x_2^2 - a_3 \overline{\Omega_r} x_4 + \ddot{\varphi}_d + \lambda_1(\dot{\varphi}_d - x_2)) \quad (3.24)$$

By following the same procedures for pitch and yaw control, altitude control and position control,

With: $e_i = x_{id} - x_i$, we find:

➤ **Pitch control θ :**

$$u_3 = -k_{s2} \text{sign}(s_\theta) + \frac{1}{b_2} (-a_4 x_2 x_6 - a_5 x_4^2 - a_6 \overline{\Omega_r} x_2 + \ddot{\theta}_d + \lambda_2(\dot{\theta}_d - x_4)) \quad (3.25)$$

With s_θ the sliding surface is selected as follows:

$$s_\theta = \dot{e}_3 + \lambda_2 e_3 \quad (3.26)$$

➤ **Yaw control Ψ :**

$$u_4 = -k_{s3} \text{sign}(s_\Psi) + \frac{1}{b_3} (-a_7 x_2 x_4 - a_8 x_6^2 + \ddot{\Psi}_d + \lambda_3(\dot{\Psi}_d - x_6)) \quad (3.27)$$

With s_Ψ the sliding surface is selected as follows:

$$s_\Psi = \dot{e}_5 + \lambda_3 e_5 \quad (3.28)$$

➤ **Altitude control Z:**

$$u_1 = -k_{s6} \text{sign}(s_z) + \frac{m}{\cos(x_1) \cos(x_3)} (-a_{11}x_{12} + \ddot{z}_d + \lambda_6(\dot{z}_d - x_{12}) + g) \quad (3.29)$$

With s_z the sliding surface is selected as follows:

$$s_z = \dot{e}_{11} + \lambda_6 e_{11} \quad (3.30)$$

➤ **Position control X:**

$$u_x = -k_{s4} \text{sign}(s_x) + \frac{m}{u_1} (-a_9 x_8 + \ddot{x}_d + \lambda_4(\dot{x}_d - x_8)) \quad /u_1 \neq 0 \quad (3.31)$$

With s_x the sliding surface is selected as follows:

$$s_x = \dot{e}_7 + \lambda_4 e_7 \quad (3.32)$$

➤ **Position control Y:**

$$u_y = -k_{s5} \text{sign}(s_y) + \frac{m}{u_1} (-a_{10}x_{10} + \ddot{y}_d + \lambda_5(\dot{y}_d - x_{10})) \quad /u_1 \neq 0 \quad (3.33)$$

With s_y the sliding surface is selected as follows:

$$s_y = \dot{e}_9 + \lambda_5 e_9 \quad (3.34)$$

3.3.1.4. Results of sliding mode (SMC) model simulation

In order to test the correctors on hard dynamics, we have identified the parameters listed in the previous table. **Table 3.2.**

We have succeeded in stabilizing the trajectories of the quadrotor model, thanks to the SMC parameters adjusted by attentives. These values are summarized in the following table:

Tableau 3-4. SMC Parameters.

SMC Parameters	X	Y	Z	ϕ	θ	Ψ
$\lambda_{surface}$	1	1	1	1	1	1
$K_{surface}$	0.2	0.2	0.5	0.1	0.1	0.1
$Width_{band}$	0.2	0.2	0.2	0.2	0.2	0.2

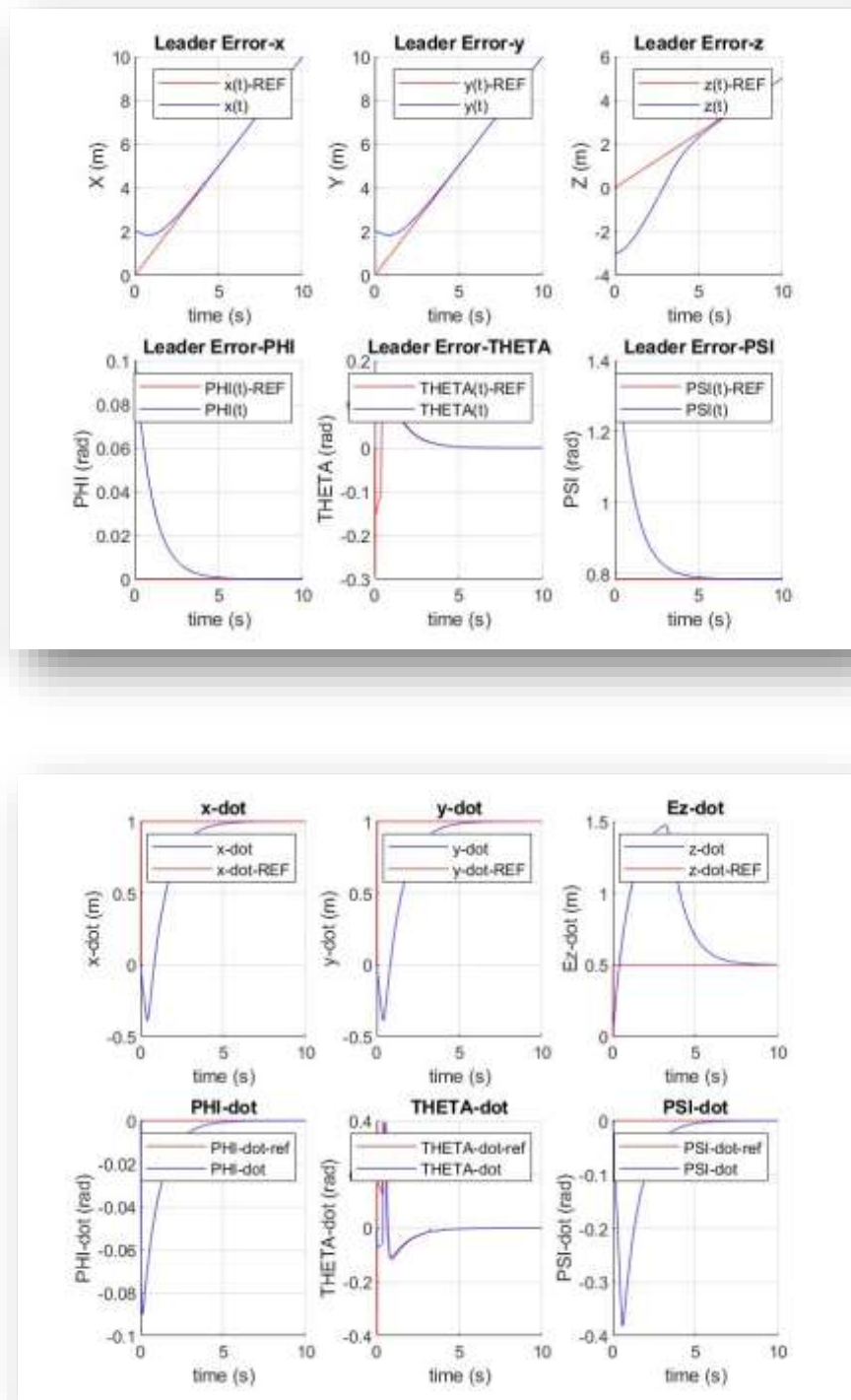


Figure 3.10. SMC Controller Simulation Response.

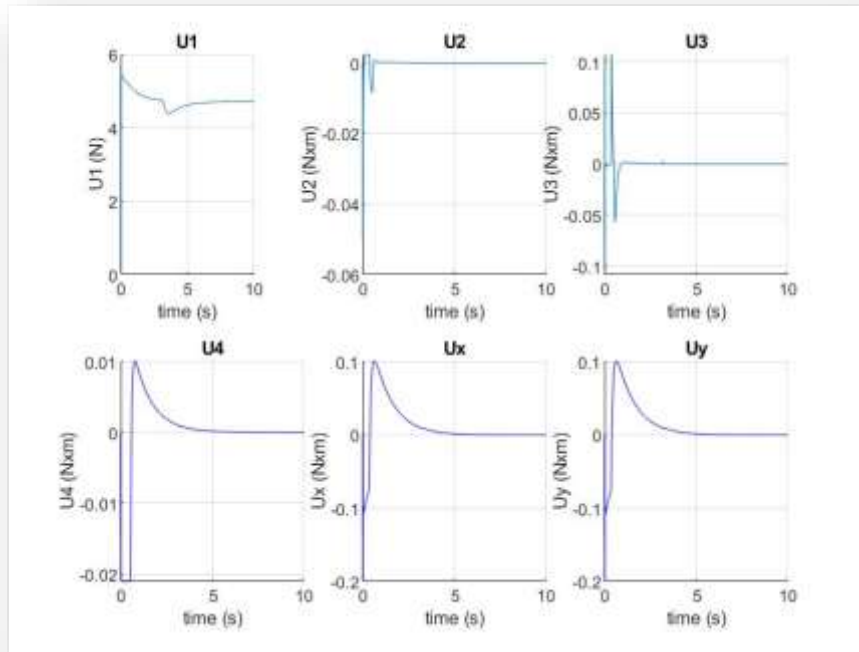


Figure 3.11. SMC Control Inputs.

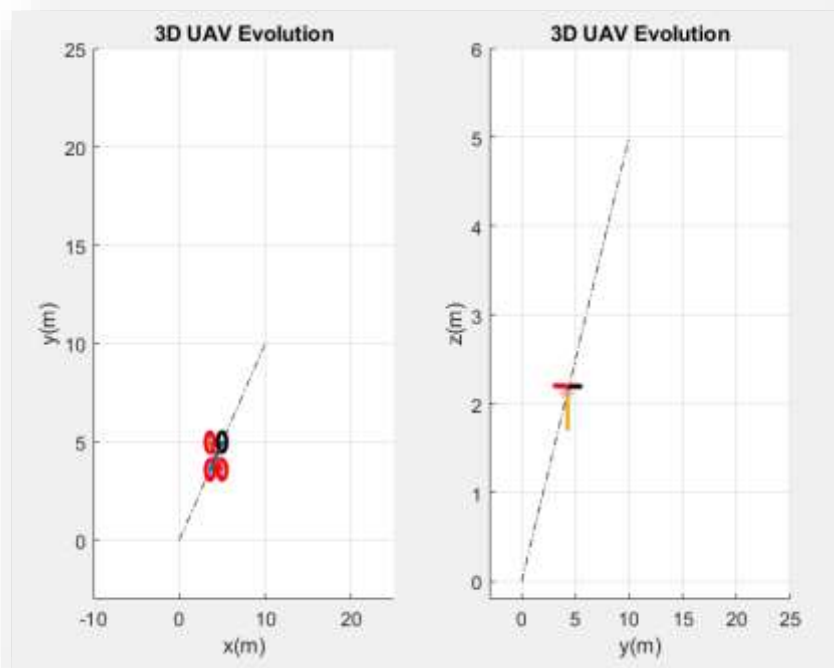


Figure 3.12. Trajectory Response under SMC.

These graphs provide an overview of the quadrotor's behavior under SMC (Sliding Mode Control). Figure 3.13 compares the actual trajectory with the desired reference trajectory in

six dimensions, evaluating the controller's ability to track the desired path accurately. It also displays the derivatives of position and orientation parameters, reflecting the quadrotor's velocity and angular velocity under SMC control. Figure 3.14 illustrates the control inputs ($U_1, U_2, U_3, U_4, U_x,$ and U_y) applied by the SMC controller, revealing its adjustments to ensure stability and precise positioning of the quadrotor. These insights collectively assess the effectiveness of the SMC controller in achieving control and tracking objectives.

3.3.2. Non Linear PID (NLPID) (Modified SMC)

To better control our quadrotor, we propose a combination of the two previous controllers, PID and SMC, into something novel called Non-linear PID. This helps in managing the quadrotor's complex movements because it is not a simple system. We're sure that Non-linear PID, which combines the best features of PID and SMC controllers, would improve the performance of the quadrotor in practical applications.

3.3.2.1. Introduction to Non Linear PID [35]

In the detailed study [35]. A modification to the existing controllers has been made, helping to solve various problems that the traditional linear or nonlinear approaches failed to adequately address using what is called NLPID. The term "NLPID control" typically refers to a type of regulator where linearity of the coefficients is not assumed. It's an innovative approach consisting of a PID structure combined with a nonlinear controller that is created from a technique based on Sliding Mode Control (SMC) theory. To design a sliding mode controller, a sliding surface that has a PID structure is used. In this context, instead of using a traditional switching term within the SMC framework, they have replaced it with a PID (Proportional-Integral-Derivative) controller. This PID controller takes the sliding surface as an input and generates control actions based on the error between the current state and the sliding surface, rather than the traditional tracking error between a reference signal and the measured signal. This new approach is clarified by **Figure 3.12**.

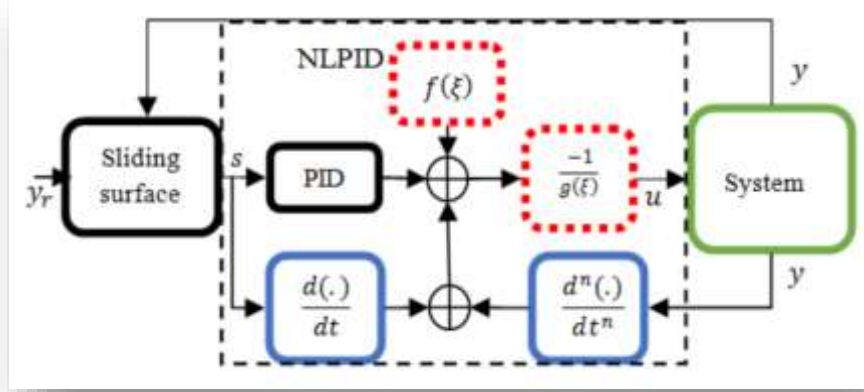


Figure 3.15. Nonlinear PID control architecture[35].

3.3.2.2. Altitude Control

The Non Linear PID altitude controller is based on the approach used by [35]. The controller is developed to control the altitude of the quadrotor. It generates the control input u_1 which is responsible for the altitude for the quadrotor. The derived control law is as follows:

$$u_1 = \frac{-m}{\cos \theta \cos \varphi} \left\{ K_{pz}(s_z(t) + \frac{1}{T_{iz}} \int_0^t s_z(\tau) \cdot d\tau + T_{dz} \frac{ds_z(t)}{dt}) + \beta_{0z} \dot{\varepsilon}_z - g - \ddot{z}_r \right\} \quad (3.35)$$

Note that the sliding surfaces are set to:

$$s_i = \dot{\varepsilon}_i + \beta_{0i} \varepsilon_i \quad |i = x, y, z \quad (3.36)$$

Also $K_{p(\cdot)}, T_{i(\cdot)}, T_{d(\cdot)}$ denote the proportional gain, the integral and derivative time constants of the NLPID structure respectively.

3.3.2.3. Attitude and Heading Control

Note that the sliding surfaces are set to:

$$\begin{cases} s_\varphi = \dot{\varepsilon}_\varphi + \beta_{0\varphi} \varepsilon_\varphi \\ s_\theta = \dot{\varepsilon}_\theta + \beta_{0\theta} \varepsilon_\theta \\ s_\psi = \dot{\varepsilon}_\psi + \beta_{0\psi} \varepsilon_\psi \end{cases} \quad (3.37)$$

Where $(\beta_{0\varphi}, \beta_{0\theta}, \beta_{0\psi})$ are positive constants.

1. Roll Controller:

Another NLPID controller is developed to control the roll angle φ of the quadrotor. The derived control law generates the input u_2 that controls the roll angle as follows:

$$u_2 = -I_x \left\{ K_{p\varphi}(s_\varphi(t) + \frac{1}{T_{i\varphi}} \int_0^t s_\varphi(\tau) \cdot d\tau + T_{d\varphi} \frac{ds_\varphi(t)}{dt}) + \beta_{0\varphi} \dot{\varepsilon}_\varphi + J_r \dot{\theta} \Omega_r \right. \\ \left. + \dot{\theta} \Psi \left(\frac{I_y - I_z}{I_x} \right) - \ddot{\varphi}_r \right\} \quad (3.38)$$

2. Pitch Controller:

A NLPID controller is developed to control the pitch angle θ of the quadrotor. The derived control law generates the input u_3 that controls the pitch angle as follows:

$$u_3 = -I_y \left\{ K_{p\theta}(s_\theta(t) + \frac{1}{T_{i\theta}} \int_0^t s_\theta(\tau) \cdot d\tau + T_{d\theta} \frac{ds_\theta(t)}{dt}) + \beta_{0\theta} \dot{\varepsilon}_\theta + J_r \dot{\varphi} \Omega_r \right. \\ \left. + \dot{\varphi} \Psi \left(\frac{I_z - I_x}{I_y} \right) - \ddot{\theta}_r \right\} \quad (3.39)$$

3. Yaw Controller :

Identical to the pitch and roll controllers, a yaw controller was developed to generate the control input u_4 based on the following control law:

$$u_4 = -I_z \left\{ K_{p\psi}(s_\psi(t) + \frac{1}{T_{i\psi}} \int_0^t s_\psi(\tau) \cdot d\tau + T_{d\psi} \frac{ds_\psi(t)}{dt}) + \beta_{0\psi} \dot{\varepsilon}_\psi + \dot{\varphi} \dot{\theta} \left(\frac{I_x - I_y}{I_z} \right) \right. \\ \left. - \ddot{\psi}_r \right\} \quad (3.40)$$

3.3.2.4. Position Controller

Once stable controllers for both the quadrotor's altitude and attitude have been obtained, a comprehensive position controller is created as follows:

$$u_x = \frac{-m}{u_1} \left\{ K_{px}(s_x(t) + \frac{1}{T_{ix}} \int_0^t s_x(\tau) \cdot d\tau + T_{dx} \frac{ds_x(t)}{dt}) + \beta_{0x} \dot{\varepsilon}_x - \ddot{x}_r \right\} \quad (3.41)$$

$$u_y = \frac{-m}{u_1} \left\{ K_{py}(s_y(t) + \frac{1}{T_{iy}} \int_0^t s_y(\tau) \cdot d\tau + T_{dy} \frac{ds_y(t)}{dt}) + \beta_{0y} \dot{\varepsilon}_y - \ddot{y}_r \right\} \quad (3.42)$$

Note that the stability analysis in the detailed study [35] shows that the origin of the studied system is exponentially stable.

3.3.2.5. Results of NLPID model simulation

The following section presents the results of our simulations, showcasing the NLPID controller's performance using the same previous conditions.

Note that the following NLPID simulations results are compared to the classical SMC.

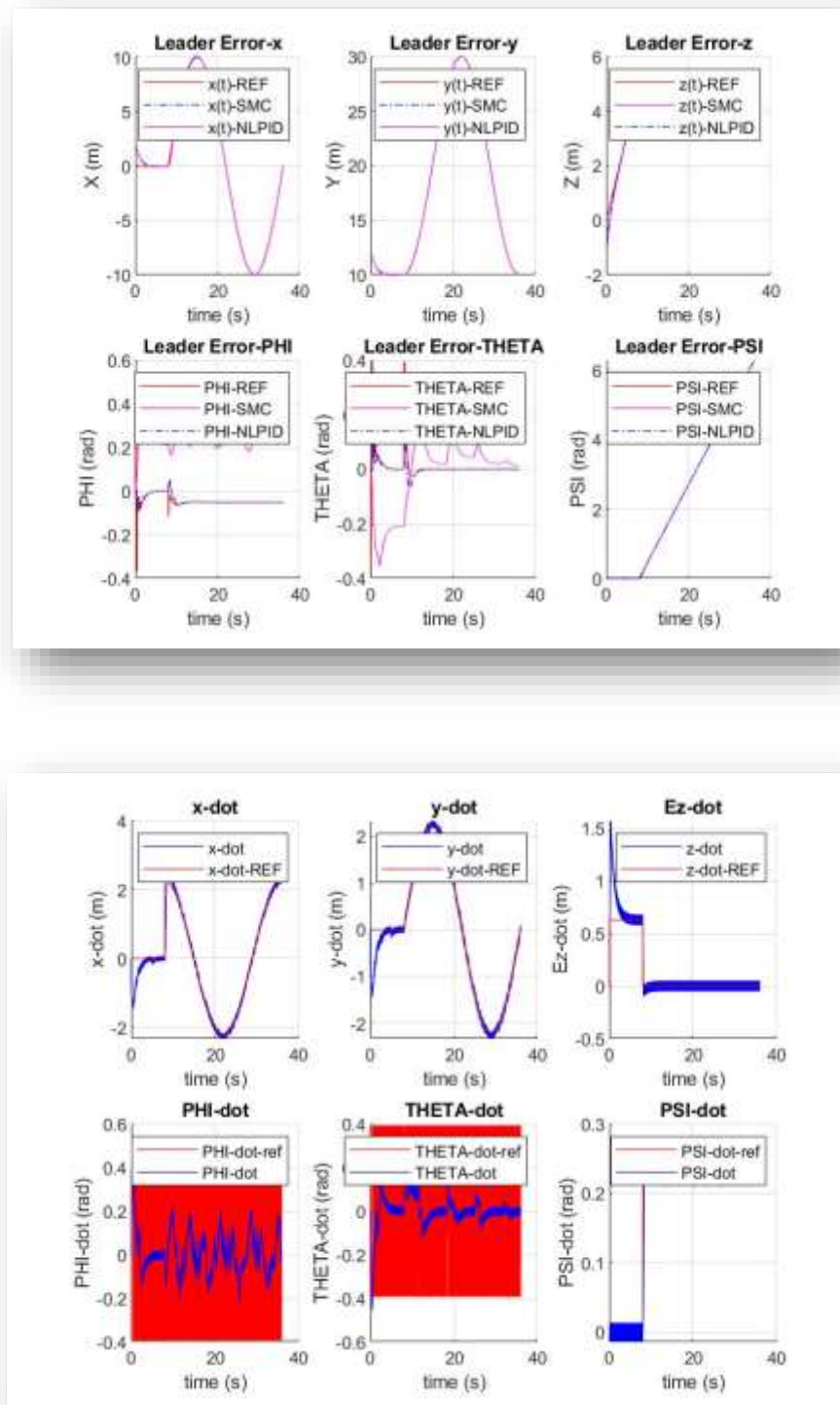


Figure 3.16. NLPID Controller Simulation Response.

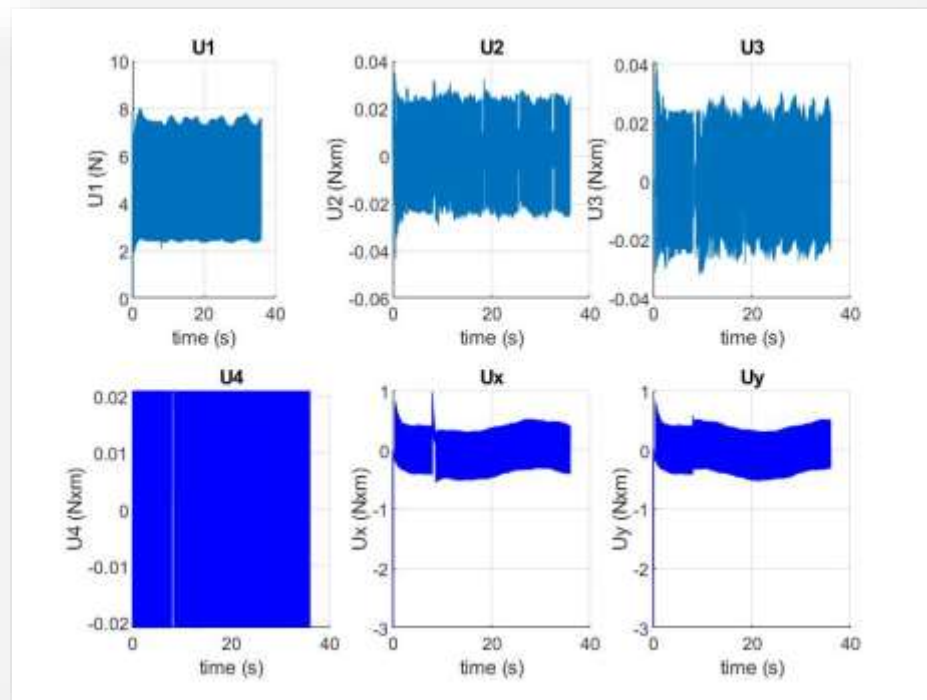


Figure 3.17. NLPID Control Inputs.

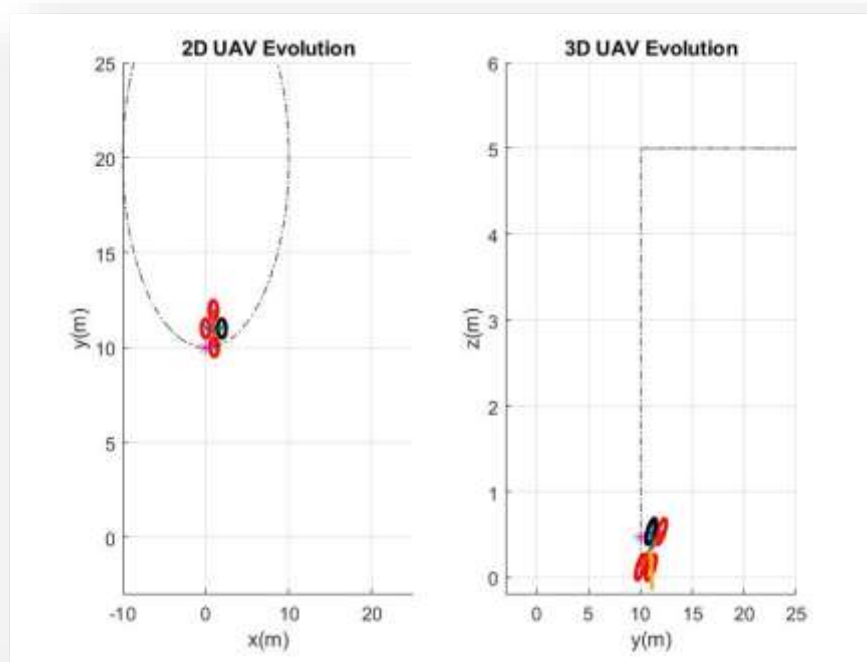


Figure 3.18. Trajectory Response under NLPID.

These graphs offer a comprehensive view of the quadrotor's behavior when controlled by the NLPID approach. Figure 3.19 compares the actual trajectory with the desired reference trajectory in six dimensions, assessing the controller's precision in tracking the desired path. The second graph presents the derivatives of position and orientation, revealing the quadrotor's velocity and angular velocity under the NLPID control. Figure 3.14 provides insights into the control inputs ($U_1, U_2, U_3, U_4, U_x,$ and U_y) applied by the NLPID controller to maintain stability and achieve precise positioning. Together, these graphs highlight the NLPID controller's effectiveness in addressing control and tracking objectives with combined SMC and PID techniques.

3.3.3. Active Disturbance Rejection Control (ADRC) [29, 41]

Over the years, control systems have evolved to address the challenges posed by disturbances and uncertainties in various industrial processes. In this context, the Active Disturbance Rejection Control (ADRC) has emerged as an alternative that combines the ease of implementation of classical PID control methods with the power of modern model-based approaches. The foundation of ADRC lies in an observer that jointly handles real disturbances and modelling uncertainties. As a result, only a coarse process model is required to design a control loop, making ADRC an appealing choice for automation professionals and offering robustness against process variations.

3.3.3.1. ADRC altitude control [29]

We take the altitude dynamics:

$$\ddot{z} = -g + \frac{C\phi C\theta}{m}U_1 - \frac{k_{ftz}}{m}\dot{z} \quad (3.43)$$

We add an external disturbance term $d(t)$ to the altitude dynamics and denote $b = \frac{C\phi C\theta}{m}$:

$$\ddot{z} = -g + bU_1 - \frac{k_{ftz}}{m}\dot{z} + d(t) \quad (3.44)$$

As a final reformulation step, we substitute $b = b_0 + \Delta b$, where b_0 represents the known part of b and Δb an (unknown) modeling error. Finally, we obtain the following equation:

$$\ddot{z} = -g + \Delta bU_1 - \frac{k_{ftz}}{m}\dot{z} + d(t) + b_0U_1 \quad (3.45)$$

We'll see later that all we need to know about our second-order process to design an ADRC is $b_0 \approx b$, i.e. an approximate value of $\frac{C\phi C\theta}{m}$.

The function $f(t)$ represents the generalized (total) disturbance, the altitude dynamics z becomes a double integrator :

$$\ddot{z} = f(t) + b_0 U_1 \quad (3.46)$$

3.3.3.2. System state representation

The basic idea behind ADRC is to use an extended state observer (ESO) that estimates the total disturbance $\hat{f}(t)$. This allows us to compensate for the impact of $f(t)$ on our model using disturbance rejection techniques. Thus, the controller's task is reduced to managing a process presenting approximately an integrator behavior, which can be easily achieved, for example, by using a proportional term.

In order to design the estimator, a state representation of the disturbed process is defined:

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}}_A \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ b_0 \\ 0 \end{pmatrix}}_B u(t) + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} f(t) \quad (3.47)$$

$$y(t) = \underbrace{(1 \quad 0 \quad 0)}_C \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} \quad (3.48)$$

With: $x_1(t) = z$, $x_2(t) = \dot{z}$, $x_3(t) = f(t)$, $u(t) = U_1(t)$

3.3.3.3. Extended state estimator

The ESO is the core part of the ADRC strategy [41]. Since the total disturbance $f(t)$, cannot be measured, an extended state observer (ESO) for this type of model cannot be built except using the input $u(t)$ and output $y(t)$ of the model. However, an estimated state $\hat{x}_3(t)$ will provide an approximate value of $f(t)$, i.e. $\hat{f}(t)$, if the real generalized disturbance $f(t)$ can be considered as constant. The equations of the extended state observer (double integrator extended by a generalized perturbation) are given in equation (3.49).

Note that for linear ADRC, a Luenberger observer is used.

$$\begin{aligned} \begin{pmatrix} \dot{\hat{x}}_1(t) \\ \dot{\hat{x}}_2(t) \\ \dot{\hat{x}}_3(t) \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{x}_1(t) \\ \hat{x}_2(t) \\ \hat{x}_3(t) \end{pmatrix} + \begin{pmatrix} 0 \\ b_0 \\ 0 \end{pmatrix} u(t) + \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} (y(t) - \hat{x}_1(t)) \\ &= \underbrace{\begin{pmatrix} -l_1 & 1 & 0 \\ -l_2 & 0 & 1 \\ -l_3 & 0 & 0 \end{pmatrix}}_{A-LC} \begin{pmatrix} \hat{x}_1(t) \\ \hat{x}_2(t) \\ \hat{x}_3(t) \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ b_0 \\ 0 \end{pmatrix}}_B u(t) + \underbrace{\begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix}}_L y(t) \end{aligned} \quad (3.49)$$

3.3.3.4. Control law

Using the estimated state variables, we can implement disturbance rejection and a linear controller for the behavior of the remaining double integrator, as illustrated in **Figure 3.16**.

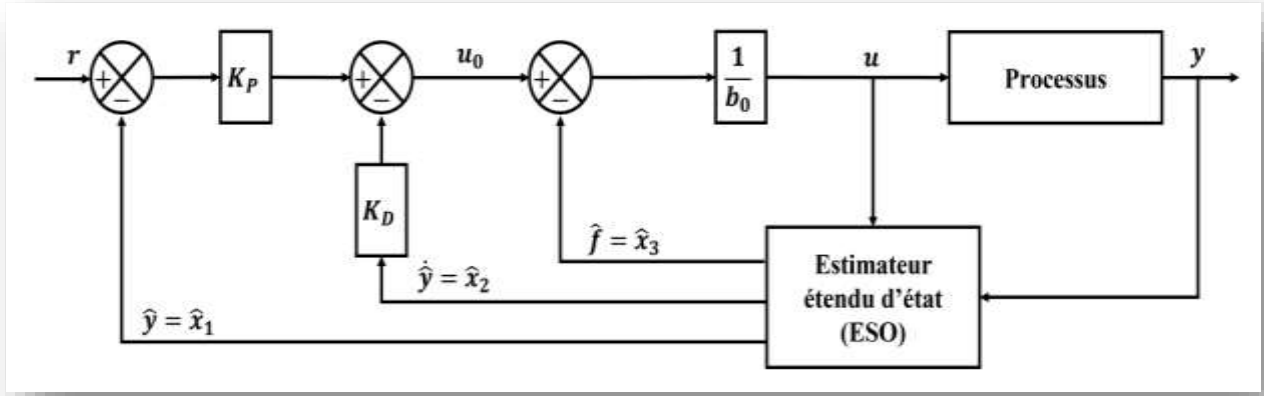


Figure 3.20. Block diagram of closed-loop ADRC control.

A modified PD controller (without the derivative part for the reference value $r(t)$) will lead to second-order closed-loop behavior with adjustable dynamics. It is a state feedback controller based on estimation.

$$\begin{cases} u(t) = \frac{u_0(t) - \hat{f}(t)}{b_0} \\ u_0(t) = K_P(r(t) - \hat{y}(t)) - K_D\dot{\hat{y}}(t) \end{cases} \quad (3.50)$$

If the estimator provides good estimations, specifically $\hat{x}_1(t) = \hat{y}(t) \approx y(t)$, $\hat{x}_2(t) = \dot{\hat{y}}(t) \approx \dot{y}(t)$, and $\hat{x}_3(t) = \hat{f}(t) \approx f(t)$, then by inserting equation (3.50) into equation (3.46), we obtain:

$$\ddot{y} = (f(t) - \hat{f}(t)) + u_0(t) \approx u_0(t) \approx K_P(r(t) - y(t)) - K_D\dot{y}(t) \quad (3.51)$$

Under ideal conditions, this results in:

$$\frac{1}{K_P}\ddot{y}(t) + \frac{K_D}{K_P}\dot{y}(t) + y(t) = r(t) \quad (3.52)$$

Applying the Laplace transform function to obtain the following closed-loop transfer function:

$$G(s) = \frac{Y(s)}{R(s)} = \frac{K_P}{s^2 + K_D s + K_P} \quad (3.53)$$

There are six parameters to find for a second-order ADRC, the control law parameters b_0, K_P and K_D and the observer parameters l_1, l_2 and l_3 .

The ADRC control structure for the quadrotor is shown in **Figure 3.17**.

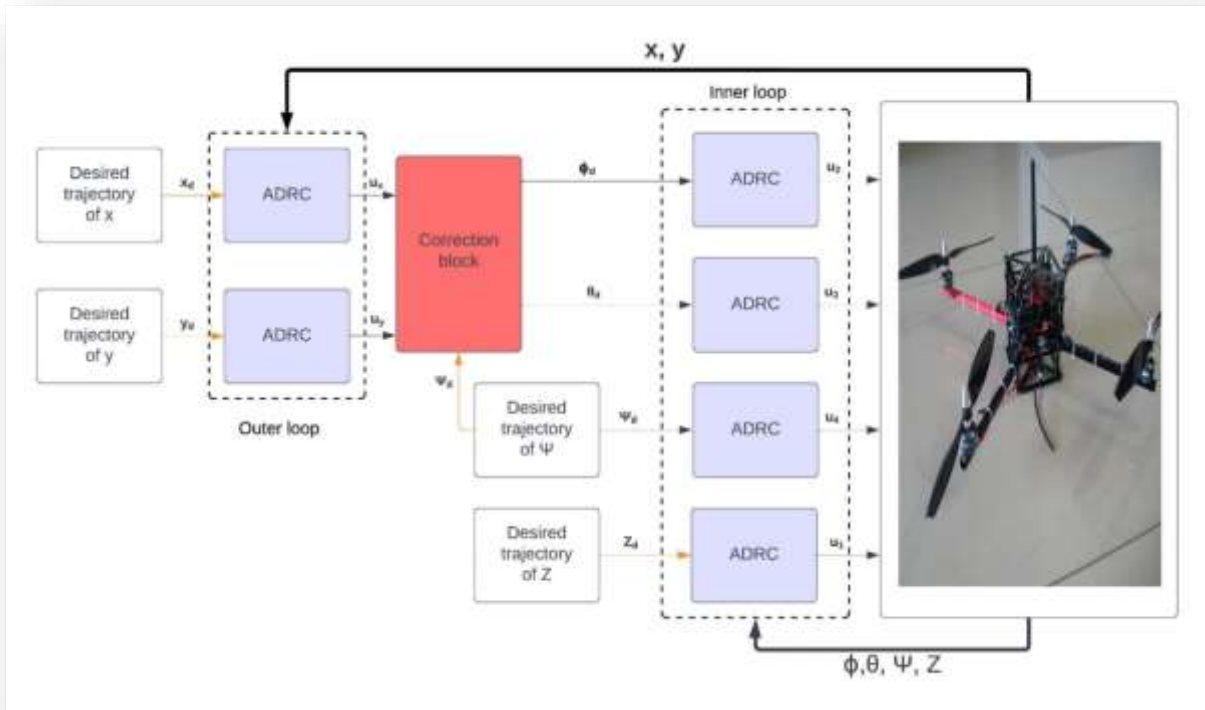


Figure 3.21. Block diagram of ADRC control applied to the quadrotor.

3.3.3.5. Results of ADRC model simulation

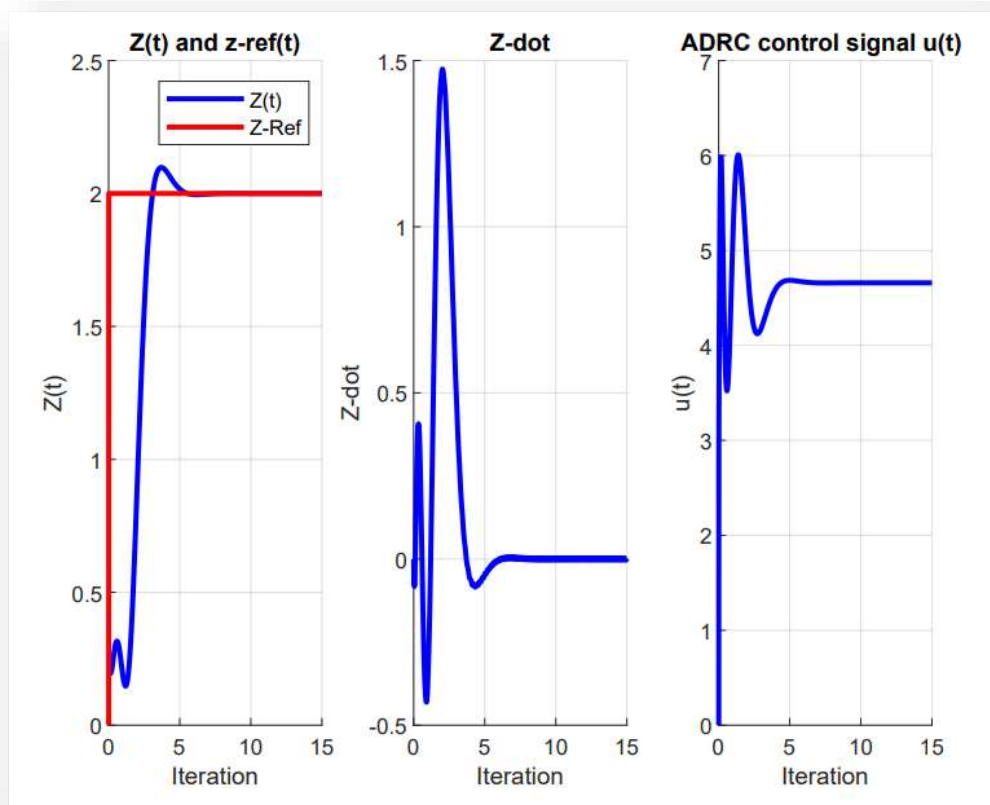


Figure 3.22. ADRC Controller Simulation Response + control signal $u(t)$ for altitude Z .

The figure 3.18 summarizes three key graphs. The first graph shows the quadrotor's altitude (Z) alongside the desired reference altitude (Z reference), assessing altitude tracking precision. The second graph displays the rate of change of altitude, reflecting the quadrotor's vertical velocity. The third graph illustrates control inputs (U) applied by the ADRC controller, providing insights into altitude stability and thrust management. Together, these graphs offer a comprehensive view of the ADRC controller's performance in altitude control and stability.

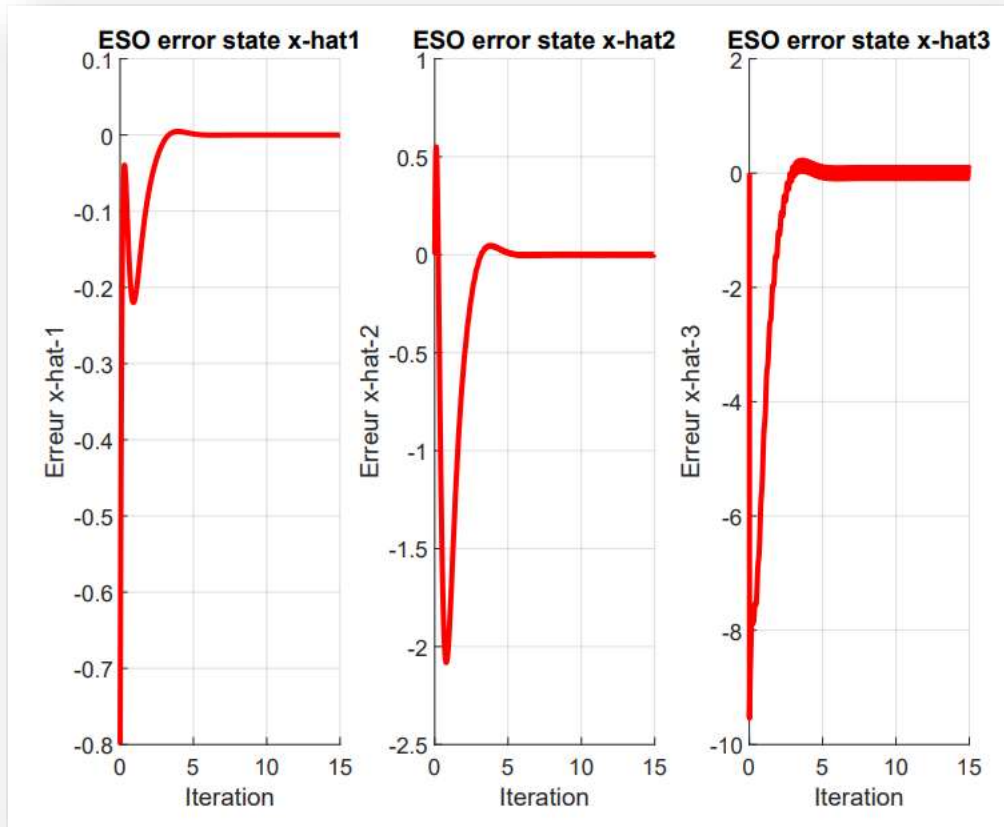


Figure 3.23. ADRC ESO error.

This figure 3.19 presents three distinct graphs, each focusing on an ESO error state. These error states provide critical insights into the performance of the ESO in estimating unmeasured system dynamics and disturbances. Analyzing these graphs allows for an assessment of the observer's accuracy and effectiveness in capturing deviations from the expected system behavior.

3.4. Interpreting simulation results

In our study, we assessed the performance of different controllers, namely PID (Proportional Integral Derivative), SMC (Sliding Mode Control), NLPID (Nonlinear PID), and ADRC (Active Disturbance Rejection Control), for quadrotor control using MATLAB simulations. The simulation results demonstrate the following:

While the PID controller offers satisfactory stability with minor oscillations, it is sensitive to disturbances and may require frequent tuning adjustments. In the other hand, although the SMC controller has increased precision and stability, it still requires a thorough understanding

of the system and may have control chattering. We confirmed the power of combining two methods when simulating the NLPID, that provides accurate trajectory tracking and reasonable robustness against disturbances, but it requires occasional tuning adjustments. The three preceding methods demonstrated accurate trajectory tracking. Finally, the ADRC shows an exceptional stability since the ESO error is converging to zero an active disturbance rejection, and remarkable robustness against disturbances, in our case we used ramp, impulse and constant disturbances.

3.5. Comparison of the four control techniques

PID, SMC, NLPID, and ADRC are examples of several control strategies, each addressed to specific characteristics, applications, control challenges and system dynamics. PID, is a widely used linear control technique, relies on the principles of proportionality, integration, and differentiation. It is well regarded for being simple and efficient for controlling linear systems. Often applied in Industrial processes (e.g., chemical, manufacturing, and power plants), motor control in robotics and automation... In contrast, SMC is a nonlinear approach that employs a sliding surface concept, ensuring robustness against uncertainties and disturbances. This makes it a favoured choice in domains like robotics, Electrical power systems and aerospace where nonlinear dynamics are prevalent. On the other hand, NLPID is a novel strategy that combines elements from both PID and SMC, introducing nonlinear elements within the PID structure to enhance performance and adaptability, particularly suitable for systems displaying nonlinear behaviors. Lastly, ADRC stands out in actively countering external disturbances using an extended state observer, making it a valuable tool in situations with unknown disturbances and modeling uncertainties, such as automotive control, Wind turbine control and robotics. The selection of these strategies depends on specific system characteristics and control needs, allowing engineers to choose the most suitable approach for their applications. Each method has its strengths, making it more suitable for certain applications and system types.

3.6. Overview of Robustness

The table 3.5 below gives an overview of the robustness characteristics of SMC (Sliding Mode Control), the innovative approach NLPID, and ADRC (Active Disturbance Rejection Control):

Tableau 3-5: Overview of robustness.

Control Technique	Robustness strengths	Robustness challenges
SMC (Sliding Mode Controller)	<ul style="list-style-type: none"> - Robust against disturbances and uncertainties. - Manages deviations from desired path 	<ul style="list-style-type: none"> - Suffers from chattering. - Demand significant tuning effort
Modified SMC (Sliding Mode Controller)	<ul style="list-style-type: none"> - Combines robustness of SMC with adaptability of PID 	<ul style="list-style-type: none"> - Robustness balance between PID and SMC components
ADRC (Active Disturbance Rejection Controller)	<ul style="list-style-type: none"> - Highly robust against external disturbances and model uncertainties. - Adapts to changing system dynamics and disturbances 	<ul style="list-style-type: none"> - Robustness depends on the quality of tuning and the accuracy of the observer.

Conclusion

In this chapter, we have designed four important control techniques applied to the quadrotor which are PID, SMC, NLPID, and ADRC. We started first by explaining the theory behind each method, we used PID controllers to stabilize our system and improve its accuracy and speed, next, the second non-linear control based on sliding mode SMC to improve its robustness, then, we tried to combine the two methods(the classical PID+ SMC) getting what we called NLPID to improve the system's performance. Moreover, the fourth non-linear control: ADRC which is particularly effective at rejecting disturbances and uncertainties in the system. And finally, used MATLAB to create simulations, showing how they work in different scenarios.

In order to obtain simulation behaviour as close as possible to real-world behaviour, we'll take things to the next level by implementing these controllers on an ARDrone 2.0 in the real world. And that's what we're going to see in the next chapter.

Chapter 4

Implementation

Introduction

In the previous chapter, we delved into the design and simulation of four distinct control methods: PID, SMC, NLPID, and ADRC. Now we move from theory to application. Some of these control techniques are used in this chapter to operate the ARDrone 2.0 platform.

As we describe the implementation process and the interactions between the controller and the ARDrone, you will see the progression from theory to practical application. When we reveal the outcomes of our ARDrone flights, which offer verifiable proof of the viability of our control strategies in real-world settings.

4.1. Overview about the Parrot AR Drone 2.0

The quadcopter used for implementing and experimentation for the thesis is the AR Drone 2.0 by Parrot. This quadcopter has established itself as a quick and adaptable quadrotor. It is the perfect option for both recreation and research activities because it blends modern technology with simplicity of usage, that's why it was chosen.



Figure 4.1. AR Drone 2.0 without and with indoor hull [36].

The AR.Drone 2.0 is a remote, durable, robust controlled consumer quadcopter developed by Parrot. The body is made of a carbon fibre tube structure and high resistance plastic. A protection hull is made of Expanded Polypropylene (EPP) foam which is durable, light in weight, and recyclable. The propellers are powered by four brushless motors (35 000 rpm, 15W power), and on-board energy is provided by a Lithium polymer battery with a capacity of 1000 mAh. This allows a life time of approximately 10 minutes. The internal processor of the Drone is a 468 MHz ARM9-processor and comes included with 128 Mb of RAM memory.

The processor runs a minimalist version of Linux on board. Moreover an integrated 802.11g wireless card provides network connectivity with an external computing device via WiFi [38].

The Parrot AR.Drone 2.0 comes with forward and bottom facing cameras, bottom facing sonar, a gyroscope, and an accelerometer. The drone hosts a WiFi connection, which a WiFi-enabled device can use to connect to the drone. The drone is 'black-boxed'; the user has access to a number of control commands and limited sensor information. For instance, direct access to the gyroscope or accelerometer is not allowed. Instead, the drone periodically sends a 'navdata' packet, containing information about the drone's velocity, rotation, battery life, and other settings. Parrot has published an application programming interface (API), which allows for simplified control of the AR.Drone. Instead of having to write complex feedback controllers for each of the four rotors, the computer onboard the AR.Drone handles all of the basic control for flight. The API gives access to commands such as forwards, backwards, turn left, and the AR.Drone will translate and execute these simple commands. The AR.Drone API also gives access to information gathered from sensors onboard the AR.Drone. A navdata packet contains information about the AR.Drone's x, y, and z velocities, as well as its current yaw, pitch, and roll. The AR.Drone does not store its own position estimate internally, as the developers decided that simply integrating the velocities over time would be too inaccurate. This is one of the major issues that this project attempts to overcome [37].

The AR.Drone 2.0 can be configured for light and low wind drag when flying outside. The drone must have exterior bumpers to safeguard it when flying indoors as shown in **Figure 4.1**.

When flying indoors, tags can be placed to the exterior of the hull to make it simple for several drones to identify one another using their cameras.

4.1.1. AR Drone 2.0 system parameters

Tableau 4-1. Parameters of the Parrot AR. Drone 2.0 model [42].

Parameter	Indoor hull	Outdoor hull	Unit
m	0.445	0.475	[kg]
l	0.125	0.125	[m]
I_{xx}	$2.7 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$	[kg m ²]
I_{yy}	$2.9 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$	[kg m ²]
I_{zz}	$5.3 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$	[kg m ²]
J_r	$2.03 \cdot 10^{-5}$	$2.03 \cdot 10^{-5}$	[kg m ²]

4.1.2. Engines [39]

The AR.Drone 2.0 is powered with brushless engines with three phases current controlled by a micro-controller. The AR.Drone 2.0 automatically detects the type of engines that are plugged and automatically adjusts engine controls. The AR.Drone 2.0 detects if all the engines are turning or are stopped. In case a rotating propeller encounters any obstacle, the AR.Drone 2.0 detects if any of the propeller is blocked and in such case stops all engines immediately. This protection system prevents repeated shocks.

4.1.3. LiPo batteries [39]

The AR.Drone 2.0 uses a charged 1000mAh, 11.1V LiPo batteries to fly. While flying the battery voltage decreases from full charge (12.5 Volts) to low charge (9 Volts). The AR.Drone 2.0 monitors battery voltage and converts this voltage into a battery life percentage (100% if battery is full, 0% if battery is low). When the drone detects a low battery voltage, it first sends a warning message to the user, then automatically lands. If the voltage reaches a critical level, the whole system is shut down to prevent any unexpected behaviour.

4.1.4. Motion sensors [39]

The AR.Drone has many motions sensors. They are located below the central hull. The AR.Drone 1.0 features a 6 DOF, MEMS-based, miniaturized inertial measurement unit. It provides the software with pitch, roll and yaw measurements. Inertial measurements are used for automatic pitch, roll and yaw stabilization and assisted tilting control. They are needed for generating realistic augmented reality effects. An ultrasound telemeter provides with altitude measures for automatic altitude stabilization and assisted vertical speed control. A camera aiming towards the ground provides with ground speed measures for automatic hovering and trimming. The AR.Drone 2.0 Add 3 DOF to the IMU with a 3 axis magnetometer (mandatory for Absolute Control mode). It also adds a pressure sensor to allow altitude measurements at any height.

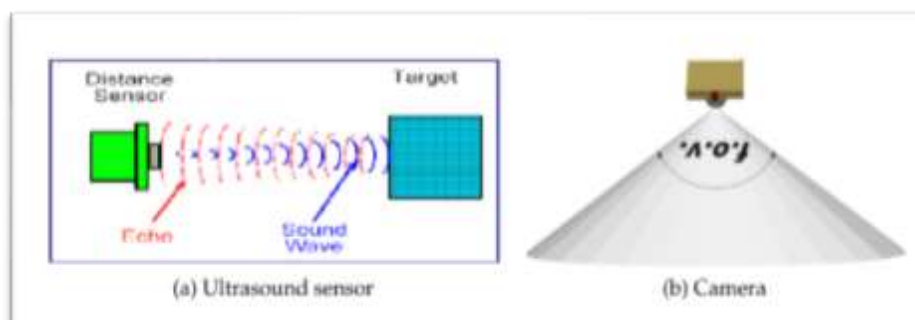


Figure 4.2. Drone Sensors.

4.1.5. Assisted control of basic manoeuvres [39]

Usually quadrotor remote controls feature levers and trims for controlling UAV pitch, roll, yaw and throttle. Basic manoeuvres include take-off, trimming, hovering with constant altitude, and landing. It generally takes hours to a beginner and many UAV crashes before executing safely these basic manoeuvres. Thanks to the AR.Drone 2.0 onboard sensors take-off, hovering, trimming and landing are now completely automatic and all manoeuvres are completely assisted.

A number of flight control parameters can be tuned:

- Altitude limit
- Yaw speed limit
- Vertical speed limit
- AR.Drone 2.0 tilt angle limit
- Host tilt angle limit

4.1.6. Advanced manoeuvres using host tilt sensors [39]

Many hosts now include tilt motion sensors. Their output values can be sent to the AR.Drone 2.0 as the AR.Drone 2.0 tilting commands. One tilting button on the host activates the sending of tilt sensor values to the AR.Drone 2.0. Otherwise hovering is a default command when the user does not input any manoeuvre command. This dramatically simplifies the AR.Drone 2.0 control by the user. The host tilt angle limit and trim parameters can be tuned.

4.1.7. Wifi network and connection [39]

The AR.Drone 2.0 can be controlled from any client device supporting Wifi. The following process is followed:

1. The AR.Drone creates a WIFI network with an ESSID usually called *ardrone2_xxx* (*ardrone_xxx* for AR.Drone 1.0) and self allocates a free, odd IP address (typically 192.168.1.1).
2. The user connects the client device to this ESSID network.
3. The client device requests an IP address from the drone DHCP server.
4. The AR.Drone DHCP server grants the client with an IP address which is :
 - The drone own IP address plus 1 (for AR.Drone 1.0 prior to version 1.1.3) 11
 - The drone own IP address plus a number between 1 and 4 (for AR.Drone 2.0 and AR.Drone 1.0 after 1.1.3 version)

- The client device can start sending requests the AR.Drone IP address and its services ports.

4.2. AR Drone tool box kit [40]

Library blocks are located in the Simulink library file *lib/ARBlocks.slx*. The ‘ARDrone Simulation Block’ and ‘ARDrone Wi-Fi Block’ are the main components of the development kit. These blocks have the same input and output to enable quick and easy transition from design and simulation to implementation.

4.2.1. AR-Drone Simulation Block

This block contains a model of the vehicle dynamics obtained via system identification. Navigation, control, and guidance algorithms should be initially validated using the simulation block.

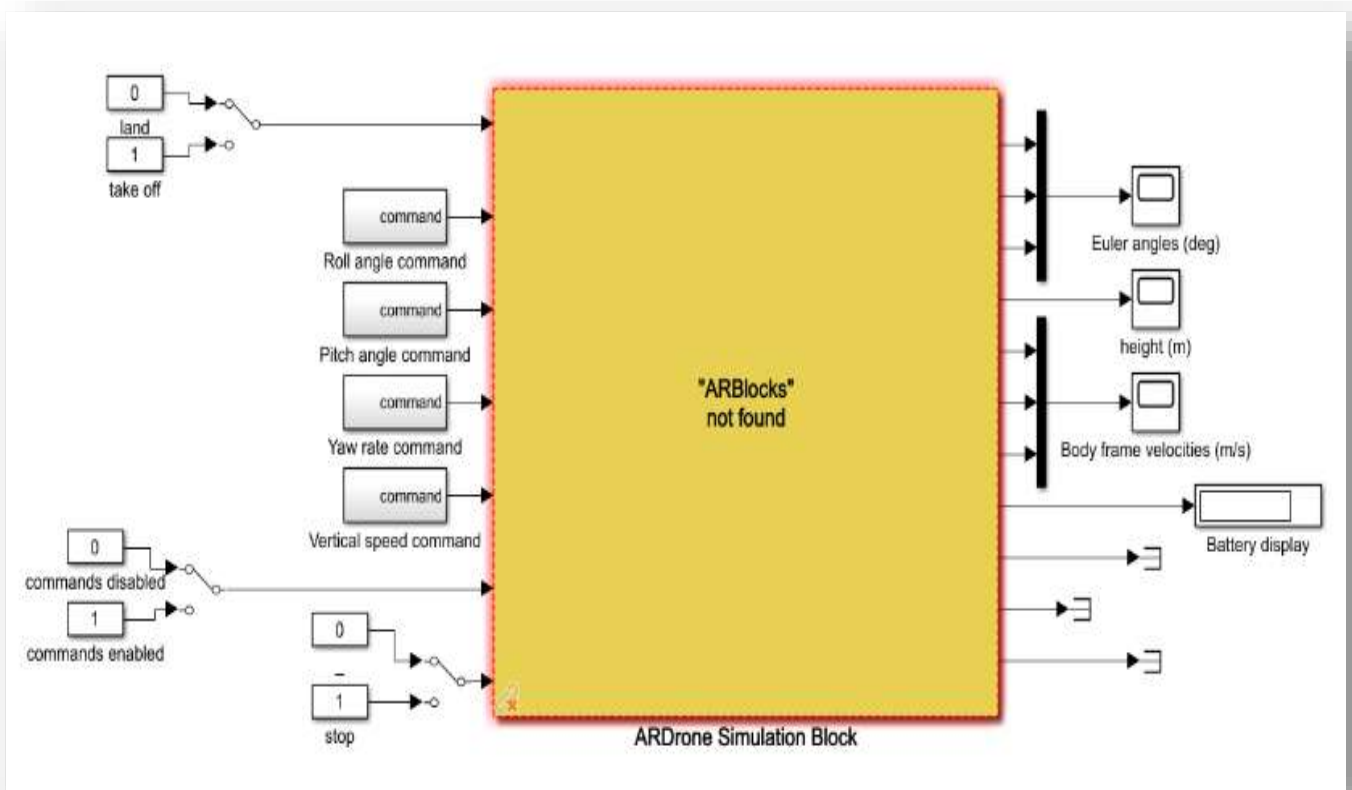


Figure 4.3. AR-Drone Simulation Block in yellow.

4.2.2. AR-Drone Wi-Fi Block

This block is an interface with the Parrot AR.Drone that allows users to simultaneously send commands to the vehicle and read the states from the vehicle. This block was constructed

using Real-Time Windows Target blocks. The AR.Drone block runs in external mode via Real-Time Windows Target to provide periodicity of navigation, control, and guidance tasks.

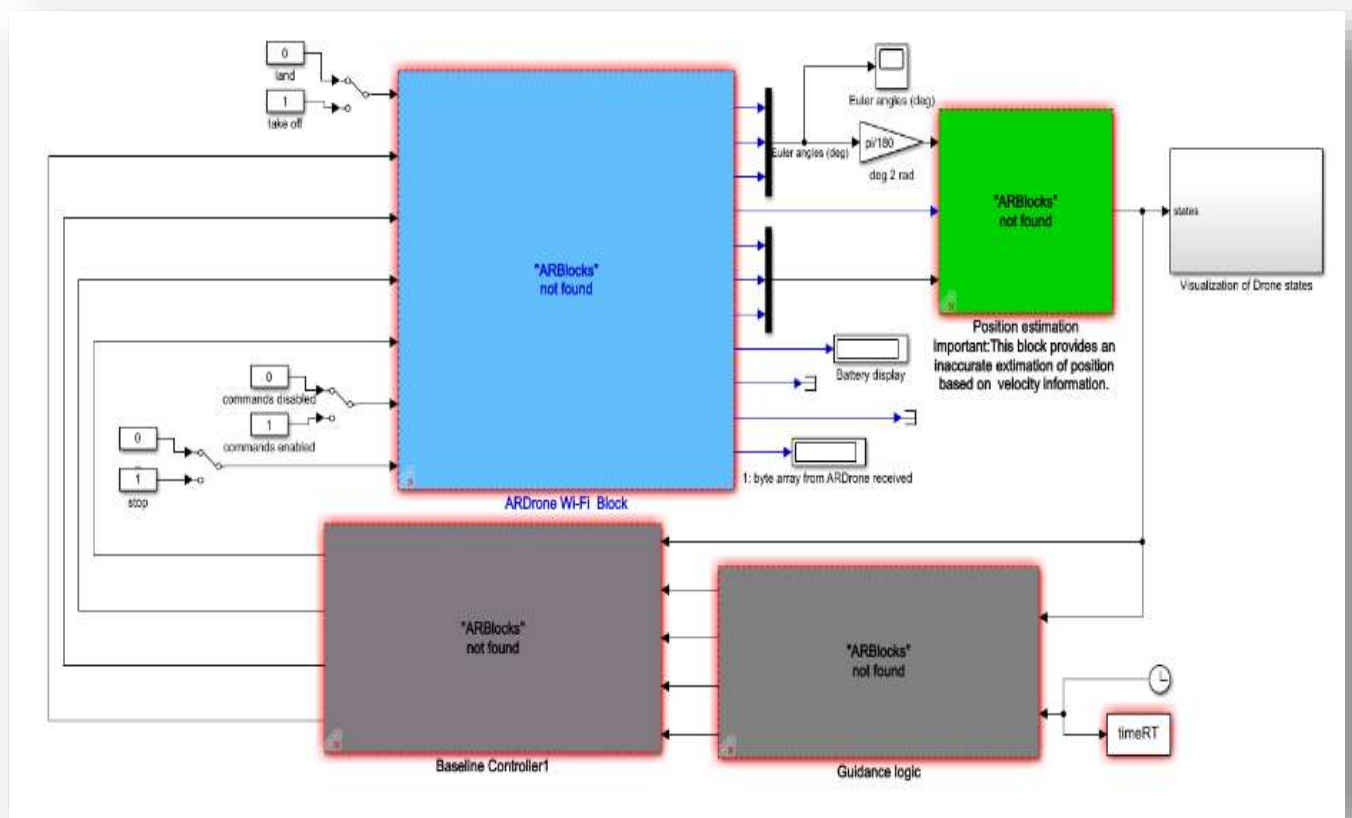


Figure 4.4. AR-Drone Wi-Fi Block in bleu.

4.2.3. Waypoint Tracking

Simulink models for both simulation and Wi-Fi control of the AR.Drone are provided to illustrate the capabilities of the blockset. The simulation and Wi-Fi control models implement a control and guidance algorithm that tracks a list of waypoints in space. The waypoints consist of the vehicle position (X_e, Y_e) in the inertial frame, height, desired heading angle, and waiting time. A diagram of the vehicle, inertial frame, and heading angle is presented in Figure 4.5. The simulation model is used to validate proper operation of algorithms before executing the algorithms on the ARDrone via the Wi-Fi control Simulink model.

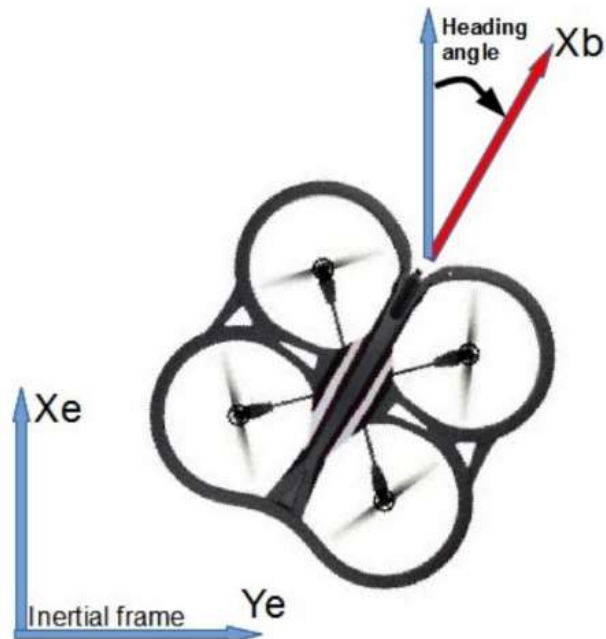


Figure 4.5. Vehicle diagram and frames.

Instructions for simulation:

1. Edit the list of waypoints located at *lib/getWaypoints()*
2. Run the simulation setup file *simulation/setupWPTrackingSim.m*
3. Run the Simulink model *simulation/ARDroneWPTracking.slx*

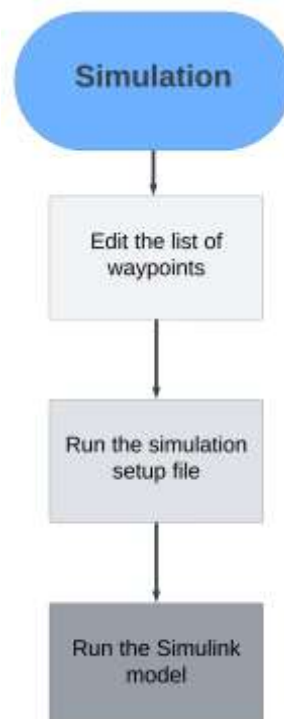


Figure 4.6. Simulation instruction flowchart.

Instructions for Wi-Fi control:

1. Connect to the ARDrone Wi-Fi network
2. Edit the list of waypoints located at *lib/getWaypoints()*
3. Run the setup script *wifiControl/setupWPTracking.m*
4. Build (ctrl+b) the Simulink model *wifiControl/ARDroneWPTracking.slx*
5. Make sure inputs fly, enable reference commands, and emergency stop are set to zero
6. Connect to the target and run *wifiControl/ARDroneWPTracking.slx*
7. Switch input fly to 1 to take off. Switch input enable reference commands to 1 to start sending commands from the Simulink controller. The heading angle is set to zero when input *input enable reference commands* is set to 1.

```

%% Cleaning the workspace

bdclose all;
clear all;
close all;

%% Adding ARDrone library to the path
addpath ../lib/ ;

%%
% Sample time of Simulink model. In demo mode, the ARDrone sends data via
% UDP every 0.065 secs.
sampleTime = 0.065;

%% Waypoints.
% Function getWaypoints returns a list of waypoints to be visited by the
% ARDrone. Waypoints should be selected within the wi-fi connection range.

waypoints = getWaypoints();

%%
% Loading Simulink model
ARDroneWPTracking ;

```

Figure 4.7. Setup waypoints Tracking.

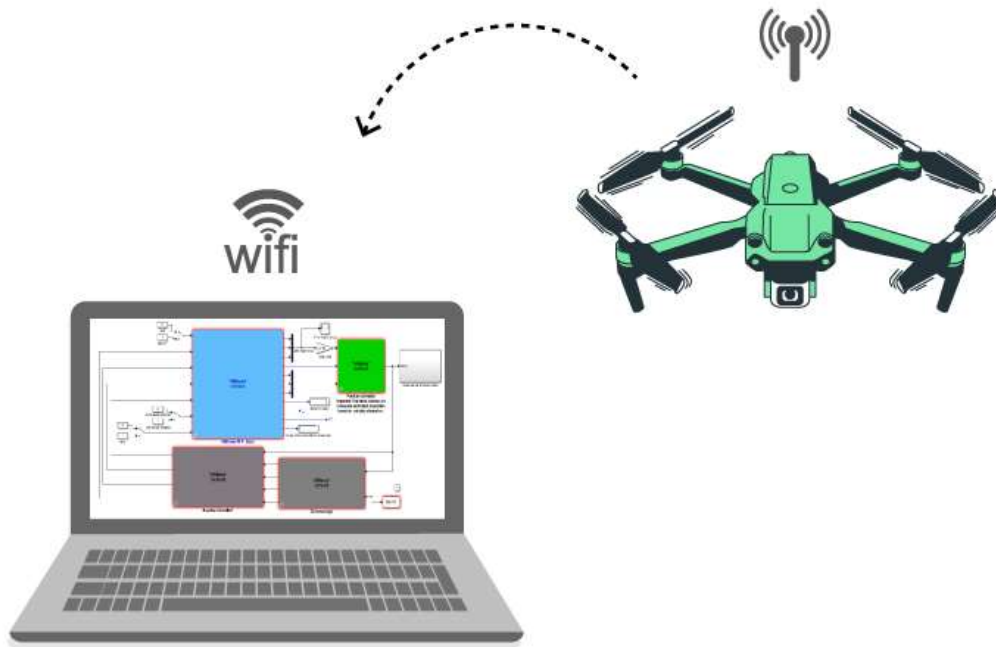


Figure 4.8. ARdrone-PC WiFi connection.

- **Control computer**

All of the command and control code of the system is run in a windows 11 on a Dell latitude 7300 laptop. The laptop has a Core(TM) i5-8365U CPU and 16 GB of RAM.

Conclusion

In this chapter, we've provided an overview of the ARdrone 2.0 starting with defining its critical system parameters and components. The ARdrone 2.0's architecture and hardware components and its basic manoeuvres, have been mentioned to provide a clear picture of its inner workings.

Moving forward, we introduced the ARdrone toolbox kit that serves as the backbone for unleashing the drone's full potential. This kit encompasses the most important steps for setting it up with WiFi network and connection and interfacing it with a personal computer. By understanding these setup procedures, we've laid the groundwork for utilizing the ARdrone 2.0's capabilities effectively. With this information, we can confidently start to explore the various uses and opportunities that this cutting-edge unmanned aerial vehicle has offer.

General

Conclusion

General Conclusion:

The objectives of this thesis work were to design a robust controller for a quadrotor UAV and test it. In the first stage, we presented a general overview of quadrotor and some of the control techniques used for UAVs in Chapter 1. In addition, these theoretical considerations were taken into account to develop the dynamic quadrotor model presented in Chapter 2. In Chapter 3, the structure of the control algorithm was explained by the PID controller, the sliding mode controller SMC, the modified sliding mode controller NLPID. A Matlab simulation was adopted to test both the dynamics and the controllers, it was necessary to identify the physical constants used in the model. Finally, Chapter 4 gave an overview of the AR drone 2.0 and their interconnections, as well as the implementation of the control laws.

Perspective and future work:

During the realization of our project, we have faced some challenges, especially in the implementation phase, i wanted to focus on the main objectives, i added a new controller which is ADRC just to explore different non linear controllers to check their characteristics. implementing and tuning complex control strategies were time-consuming and were not feasible within the scope of my project. It was essential to prioritize achievable goals within my timeframe. However, we remain committed to its continuous evolution. These challenges fuel our drive to enhance and expand our project's scope in the future, as we are dedicated to ongoing improvement and innovation. In the next phase of our research, we intend to:

- Explore optimization techniques to fine-tune controller parameters automatically. Adaptive control algorithms, reinforcement learning, or genetic algorithms can be used to optimize control gains for improved performance.
- Integrate advanced path planning and navigation algorithms. Explore algorithms that enable the quadrotor to autonomously navigate complex environments, avoid obstacles, and optimize trajectories for specific tasks.
- Investigate the integration of machine learning techniques, such as neural networks, for adaptive control. Develop controllers that can adapt and optimize their behavior based on real-time sensor data
- Focus on specific practical applications for quadrotor technology, such as a quadrotor capable of autonomously inspecting infrastructure, like bridges, power lines, or

pipelines. Use AI for anomaly detection and predictive maintenance. Integrate thermal imaging for enhanced inspection capabilities.

References

References:

- [1] F Fahlstrom, P. G. & Gleason, T. J. (2012). "Introduction to UAV Systems," Fourth Edition. <https://doi.org/10.1002/9781118396780>
- [2] Fahlstrom, P. G. & Gleason, T. J. (2012). "Introduction to UAV Systems." In Google Books. John Wiley & Sons. https://books.google.dz/books/about/Introduction_to_UAV_Systems.html?id=uLsNtm99IWYC&redir_esc=y
- [3] Paul Gerin Fahlstrom, Gleason, T. J. & Sadraey, M. H. (2022). "Introduction to UAV Systems." Wiley.
- [4] Kamesh Namuduri, Serge Chaumette, Kim, J. H., & Sterbenz, J. P. G. (2018). "UAV Networks and Communications." Cambridge University Press.
- [5] "History of Quadcopters and Multirotors — Krossblade Aerospace Systems" (2014). Krossblade Aerospace Systems. <https://www.krossblade.com/history-of-quadcopters-and-multirotors>
- [6] J. G. Leishman, "Principles of Helicopter Aerodynamics with CD Extra," Cambridge University Press, Londres, 2006.
- [7] Wikipedia, "Curtiss-Wright VZ-7, Wikipedia, The Free Encyclopedia," 2012. Available: http://en.wikipedia.org/wiki/Curtiss-Wright_VZ-7. Accessed on 01/10/2012.
- [8] S. Bouabdallah, P. Murrieri, et R. Siegwart, "Design And Control Of An Indoor Micro Quadrotor," In Actes de la Conférence Internationale sur la Robotique et l'Automatisation (ICRA'04), Vol. 5, pp. 4393-4398, La Nouvelle-Orléans, États-Unis, 26 avril - 1er mai 2004.
- [9] Ghazbi, S. N., Aghli, Y., Alimohammadi, M., & Akbari, A. A. (2016). "Quadrotors Unmanned Aerial Vehicles: A Review." International Journal on Smart Sensing and Intelligent Systems, 9(1), 309–333. <https://doi.org/10.21307/ijssis-2017-872>
- [10] mtimmons. (2018, 27 juillet). "Quadrotor Dynamic Model: Propeller Gyroscopic Effect." MTwallets. <https://www.mtwallets.com/quadrotor-dynamic-model-propeller-gyroscopic-effect/>
- [11] Cont, D., Mattia Giurato, Riccardi, F., & Lovera, M. (2018). "Ground Effect Analysis for a Quadrotor Platform," pp. 351–367. https://doi.org/10.1007/978-3-319-65283-2_19
- [12] A. Güçlü, "Attitude And Altitude Control Of An Outdoor Quadrotor," Masters Thesis, Atilim University, 2012.
- [13] Saraf, P., Gupta, M., Aivelu, M., & Parimi. (s.d.). "A Comparative Study Between a Classical and Optimal Controller for a Quadrotor." <https://arxiv.org/ftp/arxiv/papers/2009/2009.13175.pdf>
- [14] "A Literature Review of Learning and Optimization Methods Applied to Quadrotor Control." (n.d.). Scholar.google.com. Retrieved May 28, 2023, from https://scholar.google.com/citations?view_op=view_citation&hl=en&user=LIJQ_ZYAAAAJ&citation_for_view=LIJQ_ZYAAAAJ:d1gkVwhDpl0C
- [15] Amin, R., Aijun, L., & Shamshirband, S. (2016). "A Review of Quadrotor UAV: Control Methodologies and Performance Evaluation." International Journal of Automation and Control, 10(2), 87.

https://www.academia.edu/90503495/A_review_of_quadrotor_UAV_control_methodologies_and_performance_evaluation#:~:text=Int.%20J.%20Automation%20and%20Control%2C%20Vol.%2010%2C%20No.

[16] "What is Fuzzy Logic | IGI Global."(n.d.). Wwww.igi-Global.com. <https://www.igi-global.com/dictionary/fuzzy-logic/11740>

[17] Mouloud Bouchoucha, "Conception d'un contrôleur à logique floue basée sur la théorie des modes glissants," Thèse de Magister, École Militaire Polytechnique, septembre 1999.

[18] Utkin, V., & Lee, H. (2006). "Chattering Problem in Sliding Mode Control Systems." IFAC Proceedings Volumes, 39(5), 1. <https://doi.org/10.3182/20060607-3-it-3902.00003>

[19] Zenteno-Torres, J., Cieslak, J., Dávila, J., & Henry, D. (2021). "Sliding Mode Control with Application to Fault-Tolerant Control: Assessment and Open Problems." Automation, 2(1), 1–30. <https://doi.org/10.3390/automation2010001>

[20] Ejaz, M., & Chen, M. (2017). "Sliding Mode Control Design of a Ship Steering Autopilot with Input Saturation." International Journal of Advanced Robotic Systems, 14(3), 172988141770356. <https://doi.org/10.1177/1729881417703568>

[21] W. Lei and C. Li, "On-line Aerodynamic Identification of Quadrotor and Its Application to Tracking Control," IET Control Theory & Applications, vol. 11, no. 17, pp. 3097–3106, septembre 2017, doi: 10.1049/iet-cta.2017.0664.

[22] M. T. Alkowitz, V. M. Becerra, and W. Holderbaum, "Body-centric Modelling, Identification, and Acceleration Tracking Control of a Quadrirotor UAV," International Journal of Modeling, Identification, and Control, vol. 24, no. 1, p. 29, 2015, doi: 10.1504/IJMIC.2015.071697.

[23] L. Besnard, "Control of a Quadrotor Vehicle Using Sliding Mode Disturbance Observer," Master Thesis, Alabama university, 2006.

[24] "Tolérance aux défauts à travers l'utilisation de la méthode de backstepping dans les systèmes non linéaires : étude de cas sur les quadrirotors. (17 avril 2018). "http://dspace.univ-setif.dz:8888/jspui/handle/123456789/1300

[25] H. Bouadi, M. Bouchoucha et M. Tadjine, "Sliding Mode Control Based on Backstepping Approach for an UAV Type-Quadrotor," International Journal of Applied Mathematics and Computer Sciences, Barcelone, Espagne, Vol. 4, No. 1, pp. 12-17, 2007.

[26] H. Bouadi, M. Bouchoucha et M. Tadjine, "Modelling and Stabilizing Control Laws Design Based on Backstepping for an UAV Type-Quadrotor," Actes du 6e Symposium de l'IFAC sur les Véhicules Aériens Autonomes (IAV), Toulouse, France, 2007.

[27] H. Bouadi, M. Bouchoucha, and M. Tadjine, "Modelling and Stabilizing Control Laws Design Based on Sliding Mode for an UAV Type-Quadrotor," Engineering Letters, Londres, Angleterre, Vol. 15, No. 2, pp. 15-24, 2007.

[28] H. Bouadi et M. Tadjine, "Nonlinear Observer Design and Sliding Mode Control for Four Rotors Helicopter," Actes de la World Academy of Science, Engineering and Technology, Venise, Italie, Vol. 25, pp. 225-230, 2007.

[29] Lnt BENHAYA Sifeddine (2022), "Modélisation et Synthèse d'une Loi de Commande en Vue de la Stabilisation d'un UAV de Type Hexarotor," École Supérieure des Techniques de l'Aéronautique, CHAHID MOUSSA RAHALI.

- [30] A. Tayebi et S. McGilvray, "Attitude Stabilisation of a Four Rotor Aerial Robot," Proceedings of the 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, December 2004.
- [31] A. Tayebi et S. McGilvray, "Attitude Stabilization of a VTOL Quadrotor Aircraft," IEEE Transactions on Control Systems Technology, Vol. 14, No. 3, pp. 562-571, mai 2006.
- [32] Amr Nagaty, Sajad Saeedi, Carl Thibault, Mae Seto et Howard Li, "Control and Navigation Framework for Quadrotor Helicopters," Journal of Intelligent and Robotic Systems, vol. 70, nos 1-4, pp. 1–12, 2013. ISSN 0921-0296. doi: 10.1007/s10846-012-9789-z. URL <http://dx.doi.org/10.1007/s10846-012-9789-z>.
- [33] Machet Hammou (2021), "Réalisation et Commande d'un Drone Quadrirotor". <http://173.13.1.9:8080/xmlui/handle/123456789/1064>
- [34] Elkholy, H. (2014). "Dynamic Modeling and Control of a Quadrotor Using Linear and Nonlinear Approaches" [Thèse de maîtrise, l'American University in Cairo]. AUC Knowledge Fountain. <https://fount.aucegypt.edu/etds/1292>
- [35] Yasser Bouzid, Houria Siguerdidjane, Yasmina Bestaoui. "Sliding Modes based Nonlinear PID Controller for Quadrotor: Theory and Experiment." 14th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2017), Jul 2017, Madrid, Spain. p. 286-294, ff10.5220/0006433402860294ff. fhal-01589518f
- [36] Mangsatabam, R. (2018). "Control Development for Autonomous Landing of Quadcopter on Moving Platform." [online] Available at: <https://repository.lib.fit.edu/bitstream/handle/11141/2783/MANGSATABAM-THESIS-2018.pdf> [Accessed 8 Sep. 2023].
- [37] Williams, M. (2014). "Sonar-Based Autonomous Navigation and Mapping of Indoor Environments Using Micro-Aerial Vehicles." <https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=10534&context=theses>
- [38] Parrot Drone, "Parrot Drone SDK," www.msh-tools.com/ardrone/ARDrone_Developer_Guide.pdf, 2014
- [39] "Developer Guide SDK 2.0." (n.d.). <https://jpchanson.github.io/ARdrone/ParrotDevGuide.pdf>
- [40] David Escobar Sanabria (2023). "AR Drone Simulink Development-Kit V1.1". <https://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-1> , https://github.com/justynazander/AR_Drone_Simulink MATLAB Central File Exchange. Retrieved September 9, 2023.
- [41] H. Wang, Y. Huang and C. Xu, "ADRC methodology for a quadrotor UAV transporting hanged payload," 2016 IEEE International Conference on Information and Automation (ICIA), Ningbo, China, 2016, pp. 1641-1646, doi: 10.1109/ICInfA.2016.7832081.
- [42] Identification and control implementation of an A.R.Drone 2.0 Jeurgens, N. L. M. (Author). 23 Feb 2017.

- [43] Dalwadi, N., Deb, D., Ozana, S. (2023). Nonlinear Controllers for Hybrid UAV: Biplane Quadrotor. In: Adaptive Hybrid Control of Quadrotor Drones. Studies in Systems, Decision and Control, vol 461. Springer, Singapore. https://doi.org/10.1007/978-981-19-9744-0_3
- [44] Lee, T., & Kim, Y. (2001). Nonlinear Adaptive Flight Control Using Backstepping and Neural Networks Controller. *Journal of Guidance, Control, and Dynamics*, 24(4), 675–682. <https://doi.org/10.2514/2.4794>
- [45] Draganflyer Company, Available: <http://www.draganfly.com>, Accessed on 01/09/2015.
- [46] I. Kroo, F. Prinz, M. Shantz, and P. Kunz, "The Mesicopter: A Miniature Rotorcraft Concept–Phase Ii Interim Report," Stanford university, Technical report, July 2000. Available:<http://adg.stanford.edu/mesicopter/progressreports/mesicopterinterimreport.pdf>, Accessed on 25/01/2016.
- [47] P. Pounds and R. Mahony, "Design of A Four-Rotor Aerial Robot," in Australasian Conference on Robotics & Automation, pp. 145-150, Auckland, New Zealand, Dec. 06- 08, 2002.
- [48] E. Brian Nice, "Design of a Four Rotor Hovering Vehicle," Masters Thesis, Faculty of the Graduate School Cornell University, 2004.
- [49] E. Altug, J. P. Ostrowski, and R. Mahony, "Control Of A Quadrotor Helicopter Using Visual Feedback," in Proceedings. IEEE International Conference on Robotics and Automation, ICRA '02. Vol.1, pp. 72-77, 2002.
- [50] P. Castillo, A. Dzul, and R. Lozano, "Real-Time Stabilization And Tracking Of A FourRotor Mini Rotorcraft," IEEE Transactions on Control Systems Technology, Vol. 12, pp. 510-516, 2004.
- [51] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. Jung Soon, and C. J. Tomlin, "The Stanford Testbed Of Autonomous Rotorcraft For Multi Agent Control (STARMAC)," in Proceeding. 23rd Digital Avionics Systems Conference (DASC'04), Vol.2, Salt Lake City, USA, Oct. 24-28, 2004.
- [52] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design And Control Of An Indoor Micro Quadrotor," In Proceedings. International Conference on Robotics and Automation (ICRA '04), Vol.5, pp. 4393-4398, New Orleans, USA, April 26- May 01, 2004.
- [53] F. B. ÇAmlica, "Demonstration Of A Stabilized Hovering Platform For Undergraduate Laboratory," Masters Thesis, School of natural and applied science, Middle east Technical University, Dec., 2004.
- [54] M. Kemper, "Impact Of Center Of Gravity In Quadrotor Helicopter Controller Design," 4th IFAC Symposium on Mechatronic Systems, Vol. 4, pp. 157-162, Ruprecht-KarlsUniversity, Germany, Sep. 12-14, 2006
- [55] T. Glenn, V. Mario, H. Jonathan, and F. Eric, "Estimation and Control of a Quadrotor Vehicle Using Monocular Vision and Moiré Patterns," in AIAA Guidance,

- Navigation, and Control Conference and Exhibit, ed: American Institute of Aeronautics and Astronautics, pp. 21-24, , Keystone, Colorado, Aug. 21-24, 2006.
- [56] M. Wierema, "Design, Implementation And Flight Test Of Indoor Navigation And Control System For A Quadrotor UAV," Masters Thesis, Faculty of Aerospace Engineering, Delft University of Technology, 2008.
- [57] B. C. N. Carlos, J. Cook, J. Forest, S. Johnson, Ed. Massie, C. Rogers, "Technical Report Of IARC Team Quadrotor," Technical report, Virginia Tech University, 2008-2009.
- [58] J. Conroy, G. Gremillion, B. Ranganathan, and J. S. Humbert, "Erratum To: Implementation Of Wide-Field Integration Of Optic Flow For Autonomous Quadrotor Navigation," *Autonomous Robots*, Vol. 27, pp. 199-200, 2009.
- [59] F. J. Harandi and A. M. S. V. Khorani, "Modeling, Simulation and Implementation of a Quadrotor," RoboCup Iran Open 2010 Symposium, April 07- April 09, 2010, Tehran, Iran, Available: http://www.academia.edu/2220051/Modeling_Simulation_and_Implementation_of_a_Quadrotor, Accessed on 25/01/2016.
- [60] S. Bouabdallah, "Design And Control Of Quadrotors With Application To Autonomous Flying," PhD thesis, university of EPFL, 2007.
- [61] DJI Inspire 3 - Master the Unseen - DJI
- [62] Skydio 2+ | Skydio
- [63] What Is Robust Control? | Robust Control, Part 1 Video - MATLAB ([mathworks.com](https://www.mathworks.com))