République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE AHMED ZABANA DE RELIZANE

Faculté des Sciences et de la Technologie

Département d'Electrotechnique et d'Automatique

THESE DE DOCTORAT LMD 3^{éme} CYCLE

Filière : Electrotechnique
Spécialité : Commande Electrique

Titre de thèse

Contrôle Intelligent, Planification et Optimisation de la Trajectoire d'un Drone Miniaturé

Présenté et soutenu publiquement par :

M^{r} Bouziane GHOUAL

Devant les membres de jury :

Président :	M. Mohamed KHODJA	Professeur	Université de Relizane
Directeur de thèse :	M. Benyssaad YSSAAD	Professeur	Université de Relizane
Co-Directeur de thèse :	M. Youssouf MEDDAHI	MCA	Université de Chlef
Examinateur :	M. Tayeb ALLAOUI	Professeur	Université de Tiaret
Examinateur :	M. Mohamed NOUR	MCA	Centre Universitaire El Bayadh
Examinateur :	M. Abdelkader MOSTEFA	MCA	Université de Relizane

Année universitaire : 2024/2025

Algerian Republic Democratic and Popular

Ministry of Higher Education and Scientific Research

AHMED ZABANA UNIVERSITY OF RELIZANE

Faculty of Science and Technology

Department of Electrical Engineering and Automation

PHD THESIS – LMD 3rd CYCLE

Field: Electrical Engineering
Specialization: Electrical Control

Thesis Title

Intelligent Control, Path Planning, and Optimization of a Miniature Drone's Trajectory

Presented and publicly defended by:

Mr Bouziane GHOUAL

In the presence of the jury members:

President:	M. Mohamed KHODJA	Professor	University of Relizane
Thesis Advisor:	M. Benyssaad YSSAAD	Professor	University of Relizane
Co-Advisor:	M. Youssouf MEDDAHI	MCA	University of Chlef
Examiner:	M. Tayeb ALLAOUI	Professor	University of Tiaret
Examiner:	M. Mohamed NOUR	MCA	University Center of El Bayadh
Examiner:	M. Abdelkader MOSTEFA	MCA	University of Relizane

Academic Year: 2024/2025

ACKNOWLEDGMENTS

Acknowledgments

In the name of Allah, the Most Gracious, the Most Merciful.

All praise is due to Allah, Lord of the worlds, who granted me the strength, patience, and determination to undertake and complete this work. Without His divine guidance and countless blessings, none of this would have been possible.

I would like to express my deepest gratitude to my academic advisor, Professor Benyssaad YSSAAD, for their unwavering support, constructive feedback, and insightful supervision throughout this research. Their expertise and encouragement have been essential to the success of this work.

I am also sincerely thankful to my co-advisor, Mr. Youssouf MEDAHI, for his generous support, valuable input, and continuous guidance during all phases of this study. My heartfelt thanks go to Professor Bouziane MELIANI, whose collaboration, critical insights, and encouragement have been a great source of motivation and help throughout this journey.

It is also my great honor to extend my deepest appreciation to the respected jury members, Mr. Tayeb ALLAOUI and Mr. Mohamed NOUR, who kindly accepted to serve as members of the defense jury and came from distant places to participate in this evaluation. I am truly grateful for their presence, their valuable time, and their contribution to this work.

My sincere thanks also go to Mr. Abdelkader MOSTEFA and Mr. Mohamed KHODJA for their kind acceptance to serve as jury members of this defense. I hold them in high respect both as juries and as my honorable teachers.

I am especially grateful to my dear friend and brother, Dr. Sif eddine BELALIA, for his constant support, kindness, and encouragement. My honorable father Mr Mahmoud MOSTFA TOUNSI His advice and companionship meant a great deal to me and contributed positively to this experience, both personally and academically.

I also extend my appreciation to all members of the Electrical Engineering and Automatic Department of our University of Relizane, whose assistance and academic environment made this research possible.

Finally, I express my love and gratitude to my family, whose unwavering support, patience, and prayers have been my foundation throughout this path. May Allah reward them all.

May this effort be accepted and bring benefit, by the will of Allah.

ABSTRACT

Abstract

Quadcopters are widely applied in surveillance, inspection, monitoring, and delivery, where precise trajectory tracking and energy efficiency are critical under uncertain environments. This work presents advanced control strategies that combine nonlinear design, heuristic optimization, and neural adaptation to address these challenges. First, a nonlinear backstepping controller is developed for the six-degree-of-freedom dynamics of the quadcopter. To overcome manual gain selection, nature-inspired algorithms such as Grey Wolf Optimizer (GWO), Garra Rufa Optimization (GRO), and Pelican Optimization Algorithm (POA) are employed for automatic tuning. Simulations confirm improved accuracy and robustness against wind disturbances compared to traditional methods. Second, energy-efficient flight control is achieved through a Linear Quadratic Regulator (LQR) with adaptive weighting matrix tuning. A hybrid scheme using GWO and feedforward neural networks (FNNs) adjusts performance and control effort weightings, reducing actuator load while maintaining stability. The first framework, particularly with POA, ensures robustness, adaptability, and accurate path tracking, while the second achieves reliable trajectory following with improved energy efficiency based on the enhancement of FNNs.

Résumé

Les quadricoptères sont largement utilisés dans la surveillance, l'inspection, le suivi environnemental et la livraison, où la précision du suivi de trajectoire et l'efficacité énergétique sont essentielles dans des environnements incertains. Ce travail propose des stratégies de contrôle avancées combinant conception non linéaire, optimisation heuristique et adaptation neuronale pour relever ces défis. Tout d'abord, un contrôleur backstepping non linéaire est développé pour la dynamique à six degrés de liberté du quadricoptère. Afin de dépasser les limites du réglage manuel des gains, des algorithmes inspirés de la nature tels que Grey Wolf Optimizer (GWO), Garra Rufa Optimization (GRO) et Pelican Optimization Algorithm (POA) sont utilisés pour un ajustement automatique. Les simulations confirment une amélioration de la précision et de la robustesse face aux perturbations dues au vent par rapport aux méthodes traditionnelles. Ensuite, un contrôle de vol écoénergétique est obtenu grâce à un régulateur quadratique linéaire (LQR) avec ajustement adaptatif des matrices de pondération. Un schéma hybride combinant GWO et réseaux de neurones feedforward (FNNs) ajuste les pondérations liées à la performance et à l'effort de contrôle, réduisant ainsi la charge des actionneurs tout en maintenant la stabilité. Le premier cadre, en particulier avec le POA, assure robustesse, adaptabilité et suivi précis des trajectoires, tandis que le second permet un suivi fiable avec une meilleure efficacité énergétique grâce à l'amélioration des FNNs.

ملخص

تُستَخدم الطائرات الرباعية على نطاق واسع في المراقبة، التفتيش، المتابعة البيئية، وخدمات التوصيل، حيث يُعَد تتبع المسار بدقة مع كفاءة الطاقة أمراً أساسياً في البيئات غير المؤكدة. يقدّم هذا العمل استراتيجيات تحكم متقدمة تجمع بين التحكم غير الخطي، الخوارزميات المستوحاة من الطبيعة، والتعلم العصبي. في الجزء الأول تم تطوير متحكم Backstepping غير خطي لمعالجة ديناميكيات الطائرة ذات ست درجات حرية، مع اعتماد خوارزميات مثل GWO و GRO وخاصة POA لضبط المكاسب تلقائياً وتحسين الدقة والقدرة على مقاومة الاضطرابات كالرياح. في الجزء الثاني تم اعتماد منظم رباعي خطي (LQR) لتحقيق تحكم موفر للطاقة، حيث تم تعديل مصفوفات الوزن بشكل تكيفي باستخدام مزيج من GWO والشبكات العصبية FNNs مما ساهم في تقليل جهد المشغلات مع الحفاظ على الاستقرار. وقد بينت النتائج أن الإطار الأول، خاصة معPOA ، يوفر المتانة والتكيف وتتبعاً دقيقاً للمسار، بينما يحقق الإطار الثاني تتبعاً موثوقاً مع كفاءة طاقية محسّنة بفضل FNNs .

TABLE OF CONTENTS

List of Figures and Tables

List of Figures :

List of Tables :

List of Symbols / Abbreviations

LIST OF SYMBOLS / ABBREVIATIONS

Symbol	Description	Unit / Notes
m	Mass of the quadcopter	kg
g	Acceleration due to gravity	m/s^2
F_i	Thrust force produced by rotor i	N
ω_i	Angular velocity of rotor i	rad/s
k_f	Thrust coefficient	N·s^2
k_τ	Torque coefficient	N·m·s^2
l	Distance from the center to each rotor	m
τ	Torque applied on the drone	N·m
T	Total thrust from all rotors	N
τ_ϕ, τ_θ, τ_ψ	Roll, pitch, and yaw torques	N·m
x, y, z	Position coordinates in inertial frame	m
ϕ, θ, ψ	Euler angles (roll, pitch, yaw)	rad
R_{xyz}	Rotation matrix (Body to Inertial frame)	—
p, q, r	Angular velocity components	rad/s
$\dot{x}$	Derivative of variable x with respect to time	—
u_1, u_2, u_3, u_4	Control inputs: thrust and torques	N or N·m
E	Energy stored in the battery	J or Wh
V	Battery voltage	V
Q	Battery capacity	Ah
T_{flight}	Estimated flight time	s
I	Moment of inertia matrix	kg·m^2
F_d	Aerodynamic drag force	N
k_d	Aerodynamic drag coefficient matrix	N·s/m
Q	State weighting matrix in LQR	—
R	Control effort weighting matrix in LQR	—
K	Optimal state feedback gain matrix (LQR)	—
J	Performance index or cost function	—
ANN	Artificial Neural Network	—
FNN	Fuzzy Neural Network	—
w_{ij}	Weight from neuron j to neuron i	—
b_i	Bias of neuron i	—
a_i	Activation output of neuron i	—
$\sigma(\cdot)$	Activation function (e.g., sigmoid, ReLU)	—
RL	Reinforcement Learning	—
DQN	Deep Q Network (RL algorithm)	—

GENERAL INTRODUCTION

GENERAL INTRODUCTION

In recent years, Unmanned Aerial Vehicles (UAVs), and particularly quadcopters, have emerged as a prominent area of research and development within the fields of robotics, aerospace engineering, and autonomous systems. Their capacity for vertical takeoff and landing (VTOL), stable hovering, and operation in confined or complex environments has made them indispensable across numerous civil and industrial domains. Applications now include environmental monitoring, disaster response, precision agriculture, infrastructure inspection, and surveillance. Within this context, the quadcopter platform is especially notable for its relatively simple mechanical structure, agile flight characteristics, and high potential for fully autonomous operation.

Despite these advantages, quadcopters present a distinct set of control challenges. From a systems engineering perspective, they exhibit nonlinear, coupled, and underactuated dynamics. Their performance is particularly sensitive to external influences such as wind disturbances, payload variations, and actuator faults. As a result, the design of reliable and robust control strategies remains a central concern in their development [1]. Traditional control techniques—most notably the Proportional–Integral–Derivative (PID) controller—are still widely employed due to their intuitive structure and suitability for real-time implementation. However, the limitations of PID-based control become apparent in the presence of high uncertainty or strongly nonlinear behavior. Consequently, advanced methods such as the Linear Quadratic Regulator (LQR), Sliding Mode Control (SMC), Backstepping, and Model Predictive Control (MPC) have been adopted. These techniques offer improved robustness, optimality, and stability across a broader range of operating conditions [2].

Alongside these model-based strategies, recent years have seen a growing interest in intelligent and adaptive control methodologies. Artificial intelligence–inspired approaches, including Artificial Neural Networks (ANNs), Fuzzy Logic Controllers, and Reinforcement Learning (RL), have shown promise in dealing with uncertain or time-varying dynamics [3]. These methods are capable of adapting to unforeseen conditions and learning complex behaviors without requiring explicit dynamic models. Hybrid control architectures that combine classical control theory with machine learning or metaheuristic optimization techniques are of particular interest. Such systems capitalize on the stability and predictability of analytical control models while incorporating the flexibility and adaptability of data-driven approaches.

This doctoral work contributes to the field of quadcopter control by addressing the dual challenges of trajectory tracking and intelligent adaptation. The thesis is structured into four main chapters. Chapter I provides a comprehensive review of UAV classifications, propulsion technologies, and established control frameworks. It critically examines the advantages and limitations of existing approaches, identifying key challenges, particularly in achieving real-time adaptation and resilience to disturbances.

Chapter II develops a detailed nonlinear dynamic model of the quadcopter using the Newton–Euler formalism. The modeling incorporates both translational and rotational dynamics, along with actuator constraints and aerodynamic effects such as drag. Two modeling approaches are explored: the direct formulation through motion equations and the linear state-space representation. This model is implemented in MATLAB/Simulink to facilitate the design and evaluation of various control strategies. Supporting details are provided in Annex I.

Chapter III is dedicated to the design and analysis of a nonlinear Backstepping controller for trajectory tracking. While Backstepping offers improved stability over linear controllers, it depends on fixed gain parameters that may not perform effectively under diverse flight conditions or environmental disturbances. To address this limitation, three metaheuristic optimization algorithms are introduced: the Grey Wolf Optimizer (GWO), the Garra Rufa Optimizer (GRO), and the Pelican Optimization Algorithm (POA). These algorithms are used to automatically tune the Backstepping controller's gains. The optimized controllers are tested across multiple simulation scenarios, including spiral and zigzag trajectories and cases with external disturbances such as wind. Results, presented in Annex II, demonstrate significant improvements in tracking accuracy, system stability, and energy efficiency.

Building on the work of Chapter III, Chapter IV presents the central contribution of this thesis: the development of a hybrid controller that combines the LQR technique with a Feedforward Neural Network (FNN). This approach leverages the optimal control capabilities of LQR in linear regions while enabling real-time adaptability through the learning features of the FNN. The neural network is trained to predict optimal LQR gains based on the current state of the quadcopter, thus enabling dynamic adjustment of the control parameters. This hybrid controller is evaluated through simulation and benchmarked against conventional LQR and other advanced control methods. The results, detailed in Annex II, show that the FNN-enhanced LQR controller delivers superior performance in terms of trajectory accuracy, control effort, and robustness, particularly in nonlinear and uncertain environments.

The overarching objectives of this research are to design a robust and intelligent control framework for quadcopter navigation, enhance trajectory tracking performance under nonlinear and disturbed conditions, and reduce energy consumption during flight. While this study is simulation-based, the findings are intended to form the basis for future experimental validation. Potential directions for continued research include hardware implementation of our work, integration with Model Predictive Control (MPC) techniques for real-time tuning, and the extension of the hybrid framework to swarm robotics applications for coordinated multi-UAV missions. In fulfilling these objectives, the thesis addresses the themes encapsulated in its title: *Intelligent Control, Trajectory Planning, and Optimization of a Miniature Drone*. The outcomes of this work are intended to contribute meaningfully to both theoretical and practical advancements in autonomous aerial robotics.

CHAPTER I

STATE OF THE ART OF DRONES

I.1 Introduction

Drones, formally known as UAV, have rapidly evolved into a groundbreaking technology with far-reaching impacts across military, commercial, and recreational sectors. Their capacity to operate either autonomously or through remote control has redefined operational standards in fields such as agriculture, surveillance, logistics, environmental monitoring, and cinematography .

The swift progression of drone capabilities is largely fueled by continuous innovations in control systems, lightweight and durable materials, and the integration of artificial intelligence. These advancements have not only enhanced performance and efficiency but have also expanded the scope of drone applications in both urban and remote environments.

This chapter offers a detailed exploration of drone technology. It begins by defining UAVs and tracing their historical development, providing context for how they have transitioned from military prototypes to mainstream tools. The chapter then presents a clear classification of drone types, highlighting differences in design, functionality, and application areas.

A core focus is placed on the evolution of drone control systems, tracing the journey from basic manual controls to advanced autonomous navigation and trajectory tracking. Special attention is given to current state-of-the-art techniques, including AI-driven flight control, adaptive learning algorithms, and predictive modeling [1].

The chapter concludes by summarizing key insights and identifying emerging trends and future research directions in drone technology. As UAVs continue to evolve, they are poised to play an increasingly integral role in shaping modern industries and redefining what's possible in aerial operations.

I.2 Definition of Drone

A drone, also referred to as an Unmanned Aerial Vehicle, is an aircraft that flies without a human pilot onboard. Instead, it is operated either remotely by a human operator or autonomously through pre-programmed flight paths or advanced onboard automation systems. These versatile machines have become an essential tool in various domains, from aerial photography to military surveillance [2].

The term "drone" encompasses a broad range of aerial vehicles, from compact, lightweight quadcopters widely used in consumer photography and commercial applications, to large fixed-wing UAVs engineered for long-range reconnaissance, search and rescue, or defense operations [2].

Figure I.1: Drone, UAV, Quadcopter, X4.

Despite their varying sizes & purposes, they share a fundamental set of core components:

- **Airframe**: The structural body of the drone that holds all other systems together. Its design can vary widely depending on the drone's function, from compact, foldable builds to large, aerodynamic forms [3].
- **Propulsion System**: Typically consisting of electric motors paired with propellers, though larger drones may use internal combustion engines or hybrid systems for extended endurance [4].
- **Flight Controller**: The "brain" of the drone, this onboard computer collects data from sensors and issues real-time control commands to maintain stability, execute maneuvers, and follow navigation paths [5].
- **Sensors**: Critical for navigation and situational awareness, sensors include accelerometers, gyroscopes, GPS modules, barometers, and various visual sensors such as cameras, LiDAR, or infrared systems [5].
- **Communication Systems**: These enable the exchange of data and control signals between the drone and a ground control station, often using radio frequencies, Wi-Fi, or even satellite links for long-range missions [5].

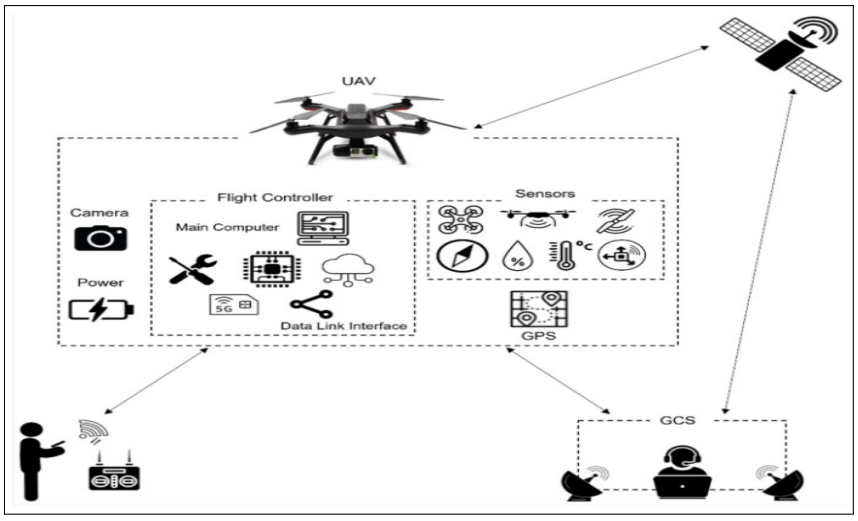


Figure I.2: Shima's Graphic of Drone Components.

A defining characteristic of drones (Figure I.2) is their level of autonomy, which can vary significantly. Some drones are fully manual, requiring constant input from a remote operator, while others are semi-autonomous, capable of executing pre-defined tasks with limited human oversight. At the cutting edge, fully autonomous drones can adapt in real-time to their environment, make complex decisions, and complete missions with little to no human intervention.
As drone technology continues to evolve, the distinction between manned and unmanned aviation grows increasingly blurred, marking a shift toward smarter, more independent aerial systems.

I.3 Historical Evolution of Drones

The concept of UAVs, now commonly known as drones, has evolved dramatically over the past century. From early experimental prototypes to intelligent flying machines capable of autonomous missions, the history of drones is a testament to human ingenuity and technological advancement. This timeline outlines key milestones and the shifting roles of UAVs across different eras.

I.3.1 Early Developments (1910s–1940s)

The origins of drone technology can be traced to the early 20th century, driven primarily by military experimentation and the need to reduce human risk in aerial warfare [6].

- **1916 – The Hewitt-Sperry Automatic Airplane**: Often considered the first true UAV prototype, this pilotless biplane was designed during World War I to deliver explosives without risking a pilot's life. Although it was never used in combat, it laid the groundwork for future autonomous flight systems [**7.blanchard20191**].
- **1930s – Radioplane OQ-2**: Developed by actor-turned-inventor Reginald Denny, the OQ-2 was one of the first mass-produced drones in the United States. Used as a target for anti-aircraft gunnery training, it marked a shift toward practical military drone applications [7].
- **1944 – German V-1 Flying Bomb ("Buzz Bomb")**: Deployed during World War II, the V-1 was a pulse-jet-powered UAV used by Nazi Germany to carry out bombing missions over London and other Allied targets. It was one of the first UAVs used in offensive combat, capable of flying preset trajectories without a pilot [7].

I.3.2 Cold War Era (1950s–1980s)

The Cold War catalyzed rapid development in drone technology, as the U.S. and the Soviet Union raced to dominate the skies with increasingly sophisticated reconnaissance and target drones as shown in Figure I.3.

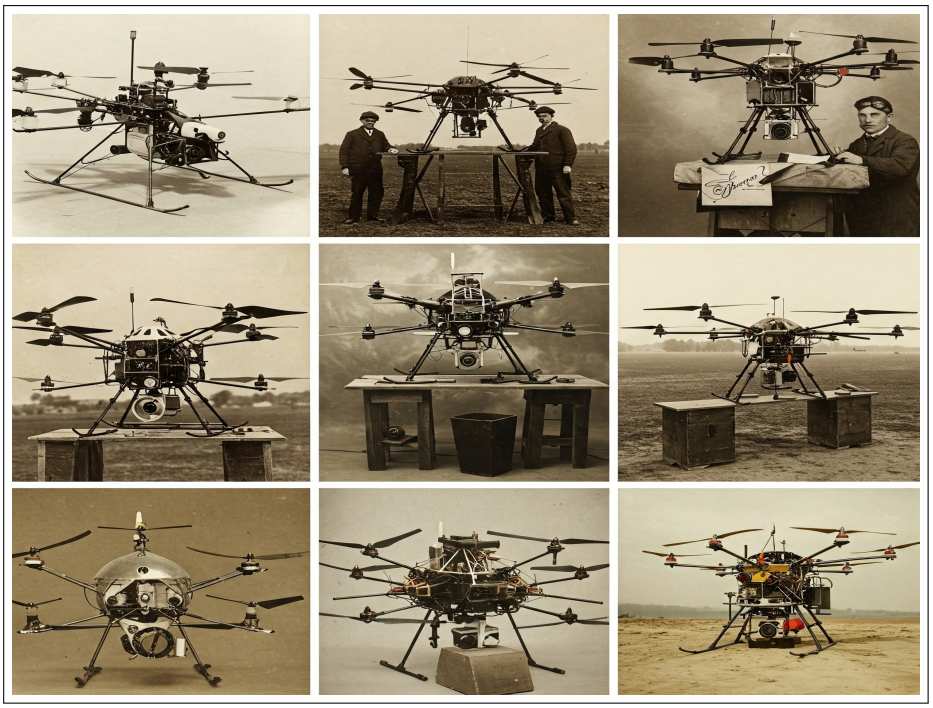


Figure I.3: Historical Overview of Drone Development in the Cold War Era.

- **1950s–1960s – Emergence of Tactical Target Drones**: Drones like the **Ryan Firebee**, launched by the U.S. Air Force, were widely used for gunnery practice and later modified for reconnaissance roles. Firebees could be launched from aircraft or ground stations and recovered by parachute, proving their versatility in training and intelligence gathering [8].
- **1960 – Use of UAVs in the Vietnam War**: The U.S. deployed modified Firebee drones for photographic reconnaissance over hostile territory. These missions demonstrated the strategic advantage of UAVs in collecting data without risking human pilots [8].
- **1970s–1980s – Lockheed D-21**: The **Lockheed D-21** was a high-speed, high-altitude UAV designed to be launched from a modified A-12 (precursor to the SR-71 Blackbird). Though short-lived due to technical issues, it was a bold step in autonomous long-range drone design [9].

I.3.3 Modern Era (1990s–Present)

The modern era has seen drones evolve from military assets into indispensable tools across numerous industries. The combination of miniaturized sensors, powerful processors, and improved communications technology has expanded their capabilities as seen in Figure I.4.

- **1990s – MQ-1 Predator and Military Dominance**: Developed by **General Atomics**, the **MQ-1 Predator** transformed drone warfare. Equipped with long-range surveillance cameras and, later, air-to-ground missiles, it enabled remote strikes and intelligence operations in real time. Its effectiveness during operations in the Balkans and Middle East redefined modern warfare tactics [10].

- **2000s – Rise of the Consumer Drone Market**: Technological advancements in GPS, lithium-polymer batteries, and brushless motors led to the development of affordable and compact drones. Companies like **DJI** revolutionized the market with drones like the Phantom series, empowering photographers, filmmakers, and hobbyists with professional-grade aerial tools [10].
- **2010s – Smart Drones and Civilian Applications**: The integration of **AI**, machine learning, and **computer vision** ushered in a new generation of smart drones. These UAVs could follow subjects autonomously, avoid obstacles, and map terrain with incredible precision. It's began to be widely used in precision agriculture, infrastructure inspection, search and rescue missions, disaster relief, and parcel delivery (e.g., Amazon Prime Air and Zipline for medical supply delivery in Africa) [11].
- **2020s to Present–Toward Full Autonomy and Swarm Intelligence**: The latest frontier includes autonomous drone swarms, beyond visual line of sight (BVLOS) operations, and urban air mobility (UAM) concepts. Companies and research institutions are developing drones that can make independent decisions in complex environments, while regulators worldwide grapple with safety, airspace integration, and ethical concerns [12].

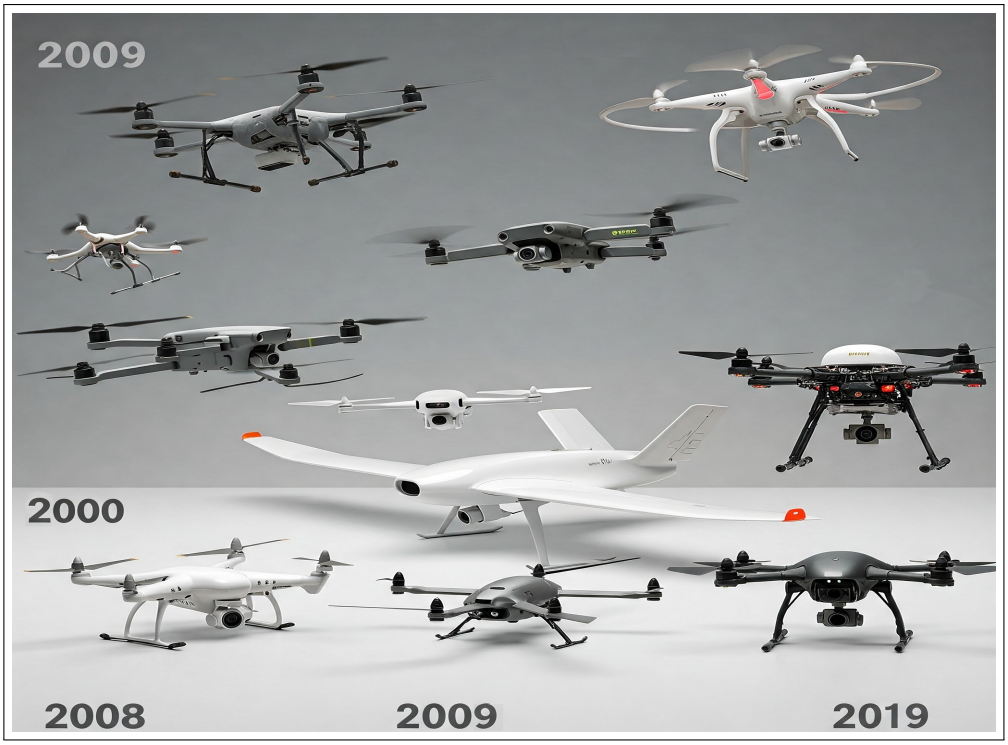


Figure I.4: Modern evaluation of Drones.

I.3.4 Technological Drivers Behind Drone Evolution

The remarkable evolution of drones over the last few decades has been fueled by rapid advancements across multiple areas of science and engineering. These technological breakthroughs have collectively transformed UAVs from simple remote-controlled aircraft into intelligent, adaptive, and highly capable aerial systems [1][5][13]. Below are the key drivers that have shaped this transformation:

- **Miniaturization of Electronics**: Advances in microelectronics have enabled critical drone components, including processors, sensors, and communication modules, to become smaller, lighter, and more power-efficient. This has enabled even compact drones to carry sophisticated payloads including high-resolution cameras, GPS modules, inertial sensors, and real-time communication systems [13]. Consumer drones like the DJI Mini-series now offer 4K video and advanced stabilization in palm-sized frames, while micro-drones can be deployed indoors or in tight urban environments for surveillance [13].
- **Improvements in Battery Efficiency and Power Systems**: Battery technology, particularly lithium-polymer (Li-Po) and lithium-ion batteries, has significantly improved in energy density, charge/discharge rates, and weight-to-power ratios. These improvements have extended drone flight times, increased payload capacity, and enabled longer mission durations [11]. High-end drones now routinely achieve **30–60 minutes** of flight time, while solar-powered drones and fuel-cell prototypes are pushing toward **multi-hour or even multi-day** endurance capabilities [11].
- **Global Navigation Satellite Systems (GNSS)**: GNSS technologies, including **GPS (U.S.)**, **GLONASS (Russia)**, **Galileo (EU)**, and **BeiDou (China)**, have revolutionized drone navigation and positioning. These systems provide real-time geolocation data with centimeter-level accuracy when paired with **RTK (Real-Time Kinematic)** or **PPK (Post-Processing Kinematic)** corrections [7]. Precision agriculture drones use GNSS to perform centimeter-accurate crop spraying, while mapping drones rely on it for creating detailed topographic surveys [7].
- **Advanced Control Algorithms and Real-Time Feedback Systems**: Modern drones leverage complex control algorithms, such as **PID** (Proportional-Integral-Derivative) controllers, **model predictive control**, and nonlinear adaptive systems to maintain flight stability and respond to environmental changes. These algorithms allow drones to execute precise maneuvers, hover in turbulent conditions, and follow complex flight paths automatically [4]. Cinematography drones can now autonomously track moving subjects and avoid collisions, while delivery drones are capable of landing precisely at designated locations, even in dynamic or windy environments [4].

- **Connectivity and Communication Advancements**:

  From traditional **radio frequency (RF)** systems to modern **Wi-Fi**, cellular **(4G/5G)**, and satellite communication, drones are now more connected than ever. Enhanced connectivity allows for long-range control, real-time video streaming, telemetry sharing, and swarm coordination [14]. BVLOS (Beyond Visual Line of Sight) operations, such as Zipline's medical supply deliveries in remote areas, are now possible through reliable cellular or satellite communication links [14].

- **Artificial Intelligence and Onboard Computing**: Onboard AI, combined with edge computing hardware like **NVIDIA Jetson**, **Raspberry Pi**, or custom neural processors, has turned drones into autonomous agents capable of real-time decision-making. AI enables drones to analyze visual data, detect objects, plan routes, avoid obstacles, and adapt to unexpected situations without human input [11]. AI-powered drones are used in disaster zones to identify survivors, in agriculture to detect crop disease, and in industrial settings for predictive maintenance and anomaly detection [11].

- **Sensor Fusion and Environmental Awareness**: Combining inputs from multiple sensors, such as LiDAR, ultrasonic sensors, thermal cameras, and vision-based SLAM (Simultaneous Localization and Mapping), allows drones to develop a detailed understanding of their surroundings [15]. Sensor fusion enhances flight safety, obstacle avoidance, and autonomous navigation in GPS-denied environments. Drones can now navigate indoors, inside mines, under bridges, or through forests, where traditional GPS-based navigation would fail [15].

Together, these technological drivers have not only expanded the functional capabilities of drones but also broadened their accessibility and affordability across industries. As these technologies continue to advance, the next generation of UAVs will be even smarter, more efficient, and capable of tasks once thought impossible for unmanned systems.
From their early days as experimental flying bombs to today's sophisticated autonomous systems, drones have transformed from military curiosities into versatile tools that are reshaping industries and redefining our interaction with the skies.

I.4 Types of Drones

Drones come in a wide array of shapes, sizes, and functionalities, tailored to meet the diverse demands of industries, governments, and individuals [11]. They can be classified across several axes, primarily by their design (aerodynamic structure), propulsion system, and intended application. Understanding these categories provides valuable insight into how drones are engineered and where they are most effectively deployed.

I.4.1 Classification by Design

The aerodynamic design of a drone directly influences its flight capabilities, including endurance, maneuverability, speed, and payload capacity.

Fixed-Wing Drones

Fixed-wing drones (Figure I.5) resemble traditional airplanes, featuring a rigid wing structure that generates lift as the drone moves forward. Unlike rotary-wing drones, they cannot hover but are highly efficient in forward flight, making them ideal for covering long distances or large areas [16].

- **Use Cases**: These drones are often used in military reconnaissance, border patrol, and large-scale agricultural monitoring.
- **Advantages**: Long endurance (often exceeding several hours), high cruising speeds, and ability to operate at higher altitudes.
- **Limitations**: Require runways or catapult systems for launch and landing, and are less suitable for confined or urban environments.

Figure I.5: Show of Fixed-Wing Drones

Rotary-Wing Drones

Rotary-wing drones, the most common being quadcopters, utilize multiple rotors to generate lift and maintain stable hovering Figure I.6). This category also includes hexacopters (6 rotors) and octocopters (8 rotors), which offer redundancy and increased payload capacity [16].

- **Use Cases**: Widely used in aerial photography, surveillance, search and rescue, and infrastructure inspection.
- **Advantages**: Vertical takeoff and landing (VTOL), excellent stability, and precise maneuverability in tight spaces.
- **Limitations**: Shorter flight times due to higher energy consumption and limited aerodynamic efficiency.

Figure I.6: Exemple of Rotary-Wing Drone.

Hybrid Drones (VTOL Drones)

Hybrid drones combine the vertical lift capability of rotary systems with the efficient forward flight of fixed wings as shown in Figure I.7. Typically designed with tiltable rotors or separate lift and cruise systems, these drones offer the best of both worlds [6].

- **Use Cases**: Used in mapping, surveying, and emergency logistics where space is constrained but range is essential.
- **Advantages**: Can take off and land vertically like a helicopter while cruising like an airplane, increasing operational flexibility [6].
- **Limitations**: Often more mechanically complex and expensive than purely fixed- or rotary-wing systems.

Figure I.7: VTOL Hybrid Drone.

I.4.2 Classification by Propulsion System

The propulsion system defines how a drone is powered and, by extension, influences flight duration, altitude capabilities, and noise profile.

Electric Drones

Powered by rechargeable batteries, electric drones (Figure I.8) dominate the consumer and light commercial markets due to their simplicity, low maintenance, quiet operation [10].

- **Use Cases**: Ideal for **photography**, **mapping**, **agriculture**, and **delivery services** in urban areas.
- **Advantages**: Low noise, environmentally friendly, and easy to operate with minimal infrastructure.
- **Limitations**: Limited by battery life, typically offering **20–40 minutes** of flight per charge depending on the drone size and payload [10].

Figure I.8: Type of Electrical Drone.

Combustion Engine Drones

These drones use gasoline, diesel, or heavy fuels and are typically deployed in military or industrial scenarios where long-endurance missions are required [17](Figure I.9).

- **Use Cases**: Suited for long-range surveillance, aerial spraying, and maritime operations.
- **Advantages**: Extended flight time (several hours), greater payload capacity, and suitability for large-scale missions.
- **Limitations**: Noisier, heavier, more maintenance-intensive, and less eco-friendly compared to electric drones.

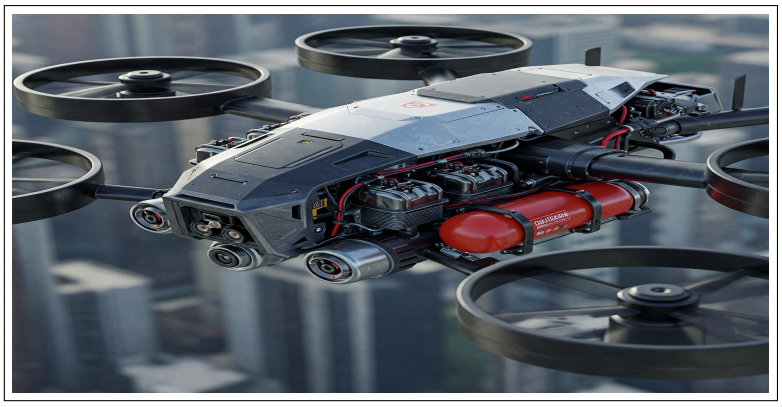


Figure I.9: Type of Combustion Engine Drone.

I.4.3 Classification by Application

The practical uses of drones are diverse, spanning defense, commercial industries, public safety, and personal entertainment. Each category of drone is optimized for the specific needs of its intended environment.

Military Drones

Designed for strategic missions, military drones are capable of operating in hostile or GPS-denied environments, often carrying advanced surveillance equipment or precision weaponry [17].

- **Examples**: The **MQ-9 Reaper**, a high-altitude drone used by the U.S. Air Force for both intelligence gathering and armed strikes.
- **Capabilities**: Long endurance (up to 27 hours), high-altitude operation, real-time data transmission, stealth features, and advanced targeting systems [17].

Commercial Drones

Used across industries, commercial drones streamline workflows, reduce costs, and increase safety in environments where human access is limited or inefficient.

- **Use Cases**:
  - ✓ **Agriculture**: Crop health monitoring, irrigation planning, and pesticide spraying [11].
  - ✓ **Logistics**: Last-mile delivery services, including time-sensitive shipments like medicines and lab samples [11].
  - ✓ **Construction & Infrastructure**: Structural inspections, site mapping, and thermal scanning for preventative maintenance [11].
- **Examples**: DJI's **Matrice 300** for industrial use and Wing (an Alphabet company) for drone delivery (Figure I.10).

Figure I.10: Commercial Drones « *DJI's Matrice 300* ».

Recreational Drones

Targeted at hobbyists and enthusiasts, these drones emphasize ease of use, portability, and entertainment features like video capture, FPV (first-person view), and stunt flying [3].

- **Use Cases**: Aerial photography, cinematography, freestyle flying, and racing.
- **Examples**: The **DJI Mini 4 Pro** (Figure I.11) for casual photography or **FPV racing drones** used in competitive drone sports.
- **Features**: Often include GPS stabilization, follow-me modes, automated flight paths, and social media integration [3].

Figure I.11: Recreational Drones « *DJI Mini 4 Pro* ».

The sheer variety of drone types highlights the technology's remarkable adaptability. Whether flying over battlefields, scanning farmland, delivering vaccines, or capturing breathtaking landscapes, drones are designed with purpose-driven engineering tailored to each mission. As drone technology advances, hybrid models and modular platforms are expected to further blur the lines between these classifications, opening the door to even more specialized and versatile applications in the near future.

I.5 State of the Art in Drone Control Systems

Drone control systems have seen a remarkable evolution, transitioning from simple stabilization mechanisms to highly sophisticated frameworks that enable full autonomy, intelligent decision-making, and even cooperative behavior among swarms. This section explores the spectrum of control strategies developed over the years, grouped into five major categories: classical, adaptive and robust, optimal, learning-based, and hybrid control. We also examine cutting-edge advances in trajectory tracking, a critical challenge in dynamic real-world environments.

I.5.1 Classical Control Methods

Classical control forms the backbone of early UAV development and is still widely used today due to its simplicity, reliability, and ease of implementation.

PID Control (Proportional-Integral-Derivative)

PID controllers are ubiquitous in drone applications. They continuously adjust motor inputs based on error signals, such as deviation from desired altitude, orientation, or velocity. The proportional component corrects the present error, the integral accumulates past errors to eliminate steady-state drift, and the derivative anticipates future trends based on the rate of error change (Figure I.12) [18].

Figure I.12: Quadrotor flight control drone test rig.

- **Strengths**:
  - ✓ Easy to implement and tune, widely supported by hardware.
  - ✓ Lightweight computation suitable for real-time control on embedded systems.

- **Weaknesses**:
  - ✓ Poor performance under nonlinear dynamics or external disturbances.
  - ✓ Requires extensive manual tuning for different flight conditions.
- **Use Case**:
  - ✓ Stabilization and basic altitude/attitude control in hobbyist and entry-level drones [18].

Backstepping Control

This Lyapunov-based method (Figure I.13) handles nonlinear systems by breaking down their dynamics into nested subsystems. By stabilizing each subsystem recursively, the overall drone behavior is tightly controlled [18].

- **Strengths**:
  - ✓ Handles nonlinear dynamics effectively through recursive stability design.
  - ✓ Offers better tracking performance than PID for complex maneuvers.
- **Weaknesses**:
  - ✓ Computationally more intensive than PID.
  - ✓ Requires precise modeling of system dynamics [18].
- **Use Case**:
  - ✓ High-precision trajectory tracking, aggressive maneuvering (e.g., acrobatic drones).

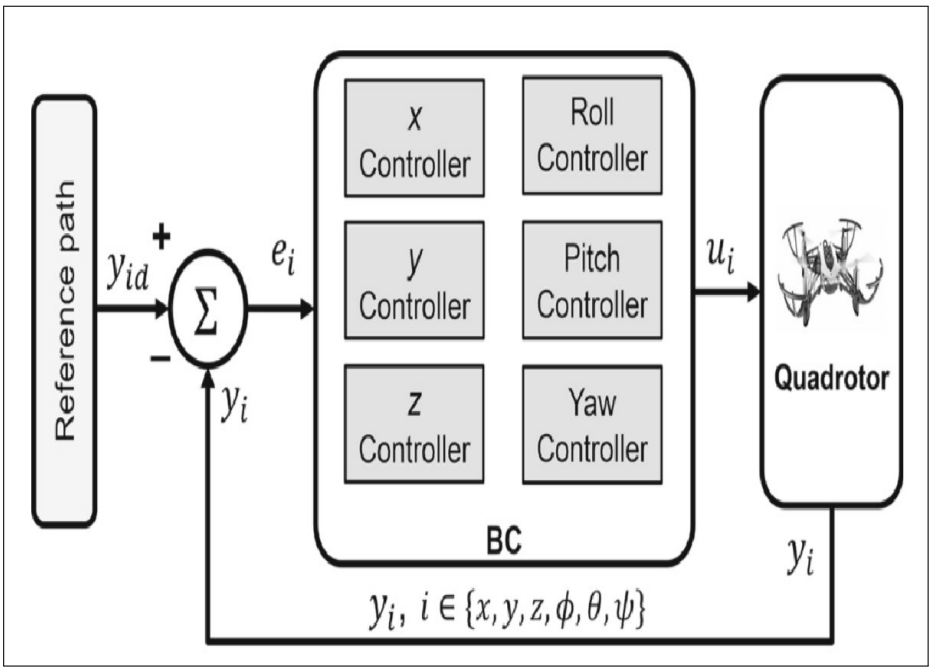


Figure I.13: Drones Controled by Backstepping Controller.

Sliding Mode Control (SMC)

SMC introduces a switching control strategy to drive system states toward a desired trajectory and maintain them there. It is known for robustness to disturbances and model inaccuracies [19].

- **Strengths**:
  - ✓ Robust to external disturbances and model uncertainties.
  - ✓ Strong convergence guarantees under varying conditions.
- **Weaknesses**:

  c

  - ✓ Chattering effect causes high-frequency oscillations in actuators.
  - ✓ May degrade mechanical components over time if not mitigated.
- **Use Case**:
  - ✓ Outdoor drones facing wind gusts or sensor noise.

I.5.2 Adaptive and Robust Control

As drones move into unpredictable and dynamic environments, adaptive and robust control strategies are vital for maintaining performance in the face of uncertainty.

Model Reference Adaptive Control (MRAC)

MRAC modifies control parameters on-the-fly to match the behavior of a predefined reference model. For example, as the drone's weight changes due to payload or battery discharge, the controller adapts accordingly [19].

- **Strengths**:
  - ✓ Online adaptability to changing drone parameters (e.g., mass, inertia).
  - ✓ Good for handling model mismatches and unknown disturbances [19].
- **Weaknesses**:
  - ✓ May suffer from stability issues if adaptation is too aggressive.
  - ✓ Sensitive to noise and requires careful gain design.
- **Use Case**:
  - ✓ Delivery drones adjusting for payload changes or battery drain during flight.

H-infinity (H∞) Control

This technique optimizes the system's worst-case performance, offering resilience against disturbances and parameter variations [20].

- **Strengths**:
  - ✓ Guarantees performance in worst-case disturbances or modeling errors [20].
  - ✓ Well-suited for multi-objective trade-offs between robustness and performance [20].
- **Weaknesses**:
  - ✓ High complexity in controller design and tuning [20].
  - ✓ Requires accurate linearized models around operating points [20].
- **Use Case**:
  - ✓ Wind-resistant control for long-range surveillance drones.

Gain Scheduling

Instead of using one controller for all conditions, gain scheduling switches between pre-tuned controllers based on measurable parameters (e.g., airspeed, altitude) [17].

- **Strengths**:
  - ✓ Flexible control across varying flight regimes.
  - ✓ Effective in handling nonlinearities through multiple operating points [17].
- **Weaknesses**:
  - ✓ May result in performance gaps during transition between gains [17].
  - ✓ Requires thorough pre-tuning and flight envelope knowledge.
- **Use Case**:
  - ✓ VTOL drones transitioning between hover and forward flight.

I.5.3 Optimal Control Strategies

Optimal control frameworks aim to achieve the best possible performance according to a mathematical cost function, balancing factors like control effort, trajectory deviation, and fuel consumption.

Linear Quadratic Regulator (LQR)

LQR minimizes a quadratic cost function by regulating system states and minimizing energy usage (Figure I.14). It works best with linearized models of drone dynamics [21].

- **Strengths**:
  - ✓ Minimizes control effort and state error systematically.
  - ✓ Simple to implement with known linear models.
- **Weaknesses**:
  - ✓ Requires linearization, which limits accuracy for highly nonlinear dynamics [21].
  - ✓ Sensitive to modeling inaccuracies.

- **Use Case**:
  - ✓ Hover stabilization and smooth path following in GPS-guided drones.

Figure I.14: Flight Drone using LQR.

Linear Quadratic Gaussian (LQG)

LQG extends LQR by integrating **Kalman filtering**, a technique for estimating unmeasured states and handling noisy sensor data [22].

- **Strengths**:
  - ✓ Combines LQR with Kalman filtering for noise-resilient control [22].
  - ✓ Suitable for systems with partial observability or sensor noise [22].
- **Weaknesses**:
  - ✓ Increased computational cost compared to LQR.
  - ✓ Can be sensitive to model inaccuracies and unmodeled disturbances.
- **Use Case**:
  - ✓ Precision indoor navigation with noisy sensors (e.g., IMUs, barometers).

Computed Torque Control (CTC)

CTC compensates for the nonlinear dynamics by using inverse dynamics models. Once linearized, classical controllers like PID or LQR can be used effectively [23].

- **Strengths**:
  - ✓ Effectively linearizes nonlinear systems for high-accuracy control [23].
  - ✓ Suitable for systems requiring precise trajectory tracking.
- **Weaknesses**:
  - ✓ Requires detailed dynamic modeling.
  - ✓ Prone to instability if model mismatches occur [23].

- **Use Case**:
  - ✓ Industrial inspection drones performing fine manipulations (e.g., power line monitoring) [23].

I.5.4 Learning-Based Control

With the rise of AI, learning-based methods have enabled drones to operate in uncertain, complex, and previously unmodeled environments.

Fuzzy Logic Control (FLC)

FLC uses human-like reasoning through heuristic rules such as "IF the drone is tilting left AND wind is strong, THEN increase right rotor thrust." It's highly flexible and requires no accurate mathematical model [16].

- **Strengths**:
  - ✓ Handles vague, imprecise input without needing accurate models [16].
  - ✓ Easy to encode expert knowledge through rules.
- **Weaknesses**:
  - ✓ Difficult to scale with system complexity [16].
  - ✓ Rule tuning becomes unmanageable for high-DOF systems [16].
- **Use Case**:
  - ✓ Mid-level control in unpredictable environments like forests or indoors.

Artificial Neural Networks (ANNs)

ANNs can approximate complex nonlinear dynamics and are often embedded within model predictive frameworks [21].

- **Feedforward ANNs**: Used to learn and generalize system behavior from data [24].
- **Recurrent ANNs (RNNs)**: Excellent for time-series prediction, handling wind gusts or sudden changes in payload [24].
- **Strengths**:
  - ✓ Excellent at approximating unknown nonlinear dynamics [24].
  - ✓ Adaptability to new data makes them flexible for changing conditions [24].
- **Weaknesses**:
  - ✓ Requires large training datasets and computational resources [24].
  - ✓ Interpretability and stability are often difficult to guarantee [24].
- **Use Case**:
  - ✓ Predictive control in urban navigation or anomaly detection during flight.

Reinforcement Learning (RL)

RL enables drones to learn through interaction with their environment. Over time, the agent optimizes its control policy by receiving rewards or penalties [12].

- **Deep RL**: Combines deep neural networks with RL, enabling real-time, end-to-end control [15].
- **Imitation Learning**: Drones mimic expert pilots by learning from demonstration data, useful for replicating human-level agility [16].

- **Strengths**:
  - ✓ Learns optimal policies directly from interactions [12].
  - ✓ Suitable for environments with high uncertainty and dynamic obstacles [12].
- **Weaknesses**:
  - ✓ High sample complexity; training is resource-intensive [12].
  - ✓ Transfer from simulation to real-world flight is non-trivial [12].
- **Use Case**:
  - ✓ Drone racing, obstacle avoidance, and autonomous exploration.

I.5.5 Hybrid and Advanced Nonlinear Control

Hybrid control strategies leverage the strengths of multiple control paradigms, combining the robustness of traditional methods with the adaptability of learning-based systems.

Lyapunov-Based Nonlinear Control

By defining energy-like functions (Lyapunov functions), controllers ensure global or local stability even in nonlinear, time-varying systems [25].

- **Integration**: Often combined with backstepping or sliding mode for aggressive, high-performance flight [25].

MPC + Neural Networks

Model Predictive Control (MPC) (Figure I.15) solves an optimization problem over a finite future horizon [25]. When paired with **Neural Networks** that predict disturbances or unknown dynamics, it becomes a powerful tool for adaptive trajectory planning.

Hierarchical Control Architectures

- **High-level Control**: Handles mission planning, task scheduling, and environmental perception using AI and global optimization [25].
- **Low-level Control**: Executes fine-tuned motor commands using PID, LQR, or SMC [25].

- **Advantage**: Decouples abstract decision-making from real-time control loops for greater scalability and safety [25].

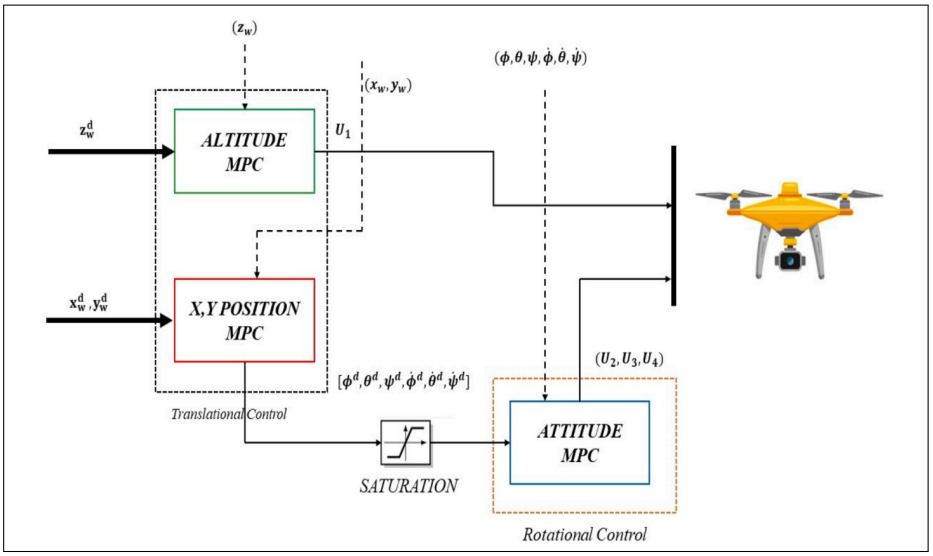


Figure I.15: Diagram of the Drone's MPC Controller.

I.6 Recent Advances in Trajectory Tracking

Trajectory tracking remains a vital challenge in drone navigation—especially in unpredictable, cluttered, or dynamic environments. Recent advances aim to improve agility, resilience, and autonomy.

I.6.1 Deep Reinforcement Learning for Dynamic Planning

Traditional path planners struggle with real-time obstacle avoidance, especially in unknown or rapidly changing terrains. Deep RL enables drones to learn navigation policies directly from simulated or real-world experiences [12].

- **Examples:**
  - ✓ Drones trained with **Proximal Policy Optimization (PPO)** can race through obstacle courses [12].
  - ✓ **Soft Actor-Critic (SAC)** allows low-latency responses to moving objects in real-time.
  - ✓ Transfer Learning: Simulation-trained policies are transferred to real-world drones using domain adaptation techniques [12].

I.6.2 Adaptive Optimal Control for Uncertain Dynamics

To handle real-time variability in drone dynamics (e.g., payload shifts, sensor drift), modern systems blend optimal control with online learning.

- **H∞ + Adaptive Tuning**: Combines robustness with real-time adaptability, ideal for autonomous delivery or rescue missions in adverse weather [20].

- **LQR with Online Learning**: Continuously updates its cost matrices and state estimators using new flight data [21].

I.6.3 Swarm Intelligence and Multi-Agent Systems

In swarm robotics, multiple drones cooperate using decentralized control laws, inspired by nature (e.g., bird flocks, fish schools).

- **Decentralized Algorithms**: Each drone operates autonomously but responds to nearby agents using rules like cohesion, separation, and alignment [12].
- **Consensus Control**: Ensures all drones in a fleet agree on shared states such as velocity, heading, or formation pattern [12].
- **Applications**:
  - ✓ Coordinated surveillance over large areas.
  - ✓ Disaster relief missions with distributed sensor networks.
  - ✓ Light shows with synchronized aerial maneuvers.

The field of drone control systems has evolved into a rich ecosystem of both theoretical and practical tools. From the foundational simplicity of PID to advanced applications like deep reinforcement learning and swarm coordination, each approach offers its own unique strengths. As drone missions become more complex, demanding greater adaptability, intelligence, and resilience, the trend is shifting toward hybrid, modular systems that integrate control theory with artificial intelligence. The future belongs to drones that not only fly but also think, learn, and collaborate.

I.7 Contemporary Trends and Specialized Applications in Drone Control

The evolution of drone control has been marked by a steady progression from basic techniques to highly advanced, intelligent systems. At the heart of early developments, **Proportional-Integral-Derivative (PID) controllers** were widely implemented for trajectory tracking, due to their simplicity and effectiveness. Foundational work by Pedro Castillo Garcia et al. (2005) addressed modeling and control strategies [26], while Jinhyun Kim et al. (2009) concentrated on achieving robust hovering capabilities [27].

The integration of expert system knowledge into control mechanisms led to the adoption of Fuzzy Logic Control (FLC), which provides a framework for managing nonlinearities with linguistic rules and heuristics. Santos et al. (2010) laid early groundwork with fuzzy controllers, while later developments included fuzzy gain-scheduling PID controllers and adaptive fuzzy techniques to improve performance across varying flight conditions [28].

Another milestone in this journey has been the introduction of Model Predictive Control (MPC) as an advanced framework for trajectory optimization. Alexis et al. (2012) made a significant contribution to the early application of MPC in drone control [29].
As the field progressed, Nonlinear MPC (NMPC) gained popularity due to its ability to handle system constraints and reduce control effort, a feature that is especially important for drones carrying suspended payloads. In addition, hybrid approaches that combine vector field methods with MPC have been proposed for resource-constrained multi-agent systems, enabling efficient trajectory tracking during coordinated missions.

Despite their initial success, PID controllers were found to have limitations in handling nonlinearities and uncertainties, prompting researchers to explore more advanced alternatives. One such direction involved optimal linear control strategies, particularly the Linear Quadratic Regulator (LQR). Argentim et al. (2013) compared LQR to PID, demonstrating performance improvements [30]. Furthermore, enhancements like the LQR with integral action (LQRI) have been proposed, enabling real-time trajectory tracking while achieving better energy efficiency than traditional PID control [30].

To bolster robustness under uncertain conditions, Sliding Mode Control (SMC) methods gained popularity. Adaptive SMC techniques developed by Mofd and Mobayen (2018), and model-free fuzzy SMC strategies introduced by Abro G. E. M. et al. (2021), showcased the potential for maintaining stability despite disturbances. Additionally, Backstepping control emerged as another method designed to ensure reliable tracking performance under uncertain dynamics [31] [32]. Advanced SMC variants, such as Terminal Sliding Mode Control (TSMC) and Nonsingular Fast Terminal Sliding Mode Control (NFTSMC), were proposed to enhance precision and dynamic response. However, certain formulations still face issues such as singularity near equilibrium points.

While Computed Torque Control (CTC) has long been established in robotic manipulator systems, its application to drone platforms has garnered increasing attention. Meddahi Y. et al. (2020) conducted a study focused on the nonlinear computed torque control of a quadrotor helicopter [23]. They introduced a CTC methodology that enables the quadrotor to execute trajectory-following missions in simulation, particularly for prospecting environments. Additional documents have explored its viability for drone-based medical logistics. For instance, Eid et al. (2020) designed and simulated a CTC-based drone system aimed at providing first aid in hard-to-reach regions [4]. Their drone was equipped with medical sensors to assess patients' vitals and used SolidWorks and CFD for aerodynamic and structural validation.

In recent years, the field has experienced a profound shift toward Neural Networks (NN) and Machine Learning (ML)-based control systems. Razmi (2018) and Wu (2020) contributed early examples of NN controllers capable of modeling and predicting drone

behavior [23] [33]. As computational capabilities expanded, Deep Reinforcement Learning (DRL) methods, such as Deep Q-learning (DQN), were applied to ensure continuous service availability in cellular-connected drones. Reviews by Xinran Wei et al. (2023) emphasized the growing role of AI in quadrotor systems [18], while Mohammed Zaid et al. (2024) identified ML and Reinforcement Learning (RL) as pivotal in the evolution of drone mobility and control strategies [14].

To adapt dynamically to changing system behavior, researchers turned to Model Reference Adaptive Control (MRAC) strategies. Villa et al. (2024) introduced a novel approach combining MRAC with an adaptive Robust Integral of the Sign of the Error (RISE) controller for cooperative load transportation [19]. Their experimental results demonstrated improved agility and responsiveness in multi-drone collaborative tasks.

Within the context of drone communication and network integration, ML-based methods are also being explored for optimal handover (HO) decisions in cellular networks [14]. Algorithms such as Double Deep Q-Network (D3QN) and hybrid TOPSIS-Q-learning aim to reduce the frequency of handovers and enhance energy efficiency [14]. Parallel research has investigated the use of Polynomial Recurrent Neural Networks (PRNNs) for adaptive PID tuning, expanding the toolbox of intelligent control methods [34]. AI is also being employed to optimize energy usage through payload management, and by improving takeoff and landing procedures to extend operational endurance [14].

Beyond control strategies, drones are increasingly employed in mission-specific applications, each presenting unique challenges and requirements. Examples include medical logistics, wildfire boundary surveillance, data acquisition in Wireless Sensor Networks (WSNs) using optimized flight paths, and search-and-rescue operations powered by LiDAR and hyperspectral imaging guided by predictive modeling. These application-driven studies underscore the importance of integrating advanced control, AI, and sensing technologies for real-world operational success.

I.8 Conclusion

The swift progress of drone technology has transformed UAVs from rudimentary military prototypes into highly sophisticated, multi-functional tools that now serve a wide range of applications across military, commercial, and recreational sectors. This chapter provided a comprehensive overview of drone technology, starting with a clear definition of UAVs and an exploration of their core components, including airframes, propulsion systems, flight controllers, sensors, and communication systems. The historical evolution of drones highlighted key milestones, tracing their development from early 20th-century experiments to modern AI-driven systems.

This progression demonstrates how drones have transitioned from being military curiosities to indispensable assets in industries such as agriculture, logistics, surveillance, and disaster relief.

A detailed classification of drones by design, propulsion, and application illustrated their versatility and adaptability to diverse tasks. Fixed-wing, rotary-wing, and hybrid drones each present unique advantages tailored to specific operational needs.

Additionally, the use of electric and combustion propulsion systems addresses varying requirements for endurance and payload capacity. The chapter also delved into advanced control systems, ranging from classical Proportional-Integral-Derivative (PID) and adaptive methods to cutting-edge AI-based approaches, such as reinforcement learning and neural networks. These technological advancements enable drones to autonomously perform complex tasks, even in dynamic and unpredictable environments.

Emerging trends, including swarm intelligence, beyond visual line of sight (BVLOS) operations, and urban air mobility (UAM), underscore the growing potential of drones to redefine modern industries. The integration of AI, sensor fusion, and advanced communication technologies is continuously expanding the capabilities of UAVs, paving the way for smarter, more efficient, and collaborative aerial systems.

Looking ahead, future research will focus on enhancing autonomy, robustness, and scalability, while addressing ongoing challenges related to regulation and ethics. The insights presented in this chapter lay the groundwork for understanding the transformative role of drones in shaping the future of aerial operations and their increasingly significant impact on society.

CHAPTER II

DRONE MATHEMATICAL MODELING : QUADCOPTER

II.1 Introduction

In recent years, the UAVs, particularly quadrotors, have gained significant attention, driven by their expanding capabilities and versatility. These flying machines are now employed in a wide range of applications, from aerial photography to complex inspection tasks, largely due to their relatively low cost, ease of use, and ability to perform demanding operations [35]. As vertical take-off and landing (VTOL) aircraft [35], quadcopters offer distinct advantages such as compact size, lightweight design, mechanical simplicity, and the potential for autonomous flight. These features have made them especially attractive to researchers and engineers [35].

Despite their growing popularity and impressive capabilities, quadrotors present considerable challenges when it comes to their internal dynamics. They are highly dynamic systems, influenced by a complex network of interacting forces and behaviors. Their motion is characterized by nonlinear, coupled, and under-actuated dynamics, making the task of controlling them both difficult and intricate.

Developing a control system that maintains a quadrotor's stability and responsiveness begins with an accurate mathematical model of its dynamics. Most control strategies for quadrotors heavily depend on the availability of such a model [36]. The primary objective is to construct a mathematical representation that enables engineers to predict the vehicle's behavior accurately, guiding the design of systems that keep the drone balanced and responsive. These models are also critical for simulations and for testing control strategies before they are deployed in real-world scenarios [36].

This chapter is dedicated to building that foundational model. The quadrotor is treated as a rigid body with six degrees of freedom (6 DOF), allowing a detailed description of both its position and orientation in space. The vehicle's motion encompasses both translational movement (through space) and rotational movement (changes in orientation).

The modeling process begins by establishing coordinate systems: one fixed to the Earth (the inertial frame) and one attached to the quadrotor itself (the body frame) [37]. Using principles from classical mechanics, the forces and torques acting on the quadrotor are then described. The Newton-Euler method is commonly employed to derive the translational and rotational equations of motion, although the Euler-Lagrange approach is sometimes used as an alternative.

The model incorporates various forces acting on the drone, such as gravitational force, which pulls the vehicle downward, and the thrust generated by the propellers, which enables flight and maneuverability [37]. By adjusting the rotational speeds of individual rotors, the quadrotor can achieve different movements and orientations. Additional factors, including aerodynamic drag and motor-propeller behavior, are also considered to create a more realistic model [37].

II.2 Quadcopter Overview:

Unmanned Aerial Vehicles, and quadcopters in particular, have seen a significant rise in popularity in recent years. This surge is largely driven by their versatility, relatively low cost, and ease of operation. These flying machines, often referred to as quadrotors or **QUAVs**, belong to a class of multirotor aircraft equipped with four individual rotors [13]. Their ability to take off and land vertically places them in the category of Vertical Take-Off and Landing **(VTOL)** vehicles [35].

Quadcopters have become essential platforms in fields such as aerial robotics, research, surveillance, delivery systems, and entertainment. Their appeal stems from several key features, including compact size, lightweight construction, mechanical simplicity, and a high level of autonomy. In practical applications, they are capable of steady hovering, precise maneuvering in confined spaces, and can be operated using relatively straightforward hardware and software. These advantages make them especially attractive to engineers and researchers alike.

Structurally, a quadcopter consists of a rigid frame with four rotors positioned at the ends of a cross-shaped configuration [38]. The central section of the frame typically houses critical electronics, such as the flight controller, which acts as the brain of the system. Frames are commonly constructed from materials like carbon fiber or aluminum, providing a strong yet lightweight design.

Each rotor comprises a propeller driven by an individual motor, most commonly a brushless DC motor due to its efficiency and durability. The rotors are arranged in two pairs: one pair spins **clockwise (CW)**, and the other spins **counter-clockwise (CCW)** [39]. This arrangement is crucial for balancing the torque generated by the spinning propellers. Without such balance, the drone would experience uncontrollable rotation [39]. For instance, if rotors 1 and 3 spin clockwise, then rotors 2 and 4 must spin counter-clockwise to maintain stability.

Other key components in a quadcopter system include:

- **Electronic Speed Controllers (ESCs)**, which regulate the speed of each motor.
- **Lithium Polymer (LiPo) batteries** that supply power.
- **Sensors and flight control boards** that manage movement and orientation.

The movement of a quadcopter is controlled by adjusting the **rotational speeds of the four rotors**. Here's how different types of motion are achieved:

- **Lift or Vertical Movement** is created by increasing or decreasing the speed of all four rotors at the same time [40].
- **Roll Movement**, which causes the drone to tilt side to side (along the X-axis), is achieved by changing the speed difference between the left and right rotors [40].

- **Pitch Movement**, which tilts the drone forward or backward (along the Y-axis), comes from varying the speeds between the front and back rotors [40].
- **Yaw Movement**, which rotates the drone left or right (around the Z-axis), results from adjusting the balance between the CW and CCW spinning rotors [40].

Even though quadcopters are fairly straightforward to construct and fly, their dynamics and control behavior are quite intricate. The mathematical models that describe them are nonlinear, coupled, and underactuated, indicating that they possess more degrees of freedom than the number of control inputs available. As a result, achieving stable and precise control becomes a demanding task.

II.2.1 Structure and Components

The **structural design** of a quadcopter is relatively simple but highly effective. The key components include:

- **Frame**: The frame, usually constructed from lightweight and durable materials like carbon fiber, ensures structural integrity without adding unnecessary weight. It serves as the foundation for mounting essential components including the motors, rotors, and sensors [38].

  Total frame weight contributes to the system's mass:

$$\mathbf{m_{total}} = \mathbf{m_{frame}} + \mathbf{m_{motors}} + \mathbf{m_{battery}} + \cdots \tag{II.1}$$

  This total mass affects the gravitational force:

$$\mathbf{F_g} = \mathbf{m_{total}} \cdot g \tag{II.2}$$

- **Rotors and Motors**: A quadcopter features four rotors, with two generally spinning clockwise (CW) and the other two counterclockwise (CCW) [38]. This opposite rotation pattern counteracts the torque generated by each rotor, which plays a key role in maintaining balance and overall stability.

  To cancel net yaw torque:

$$\tau_{\mathbf{CW}} = \tau_{\mathbf{CCW}} \Rightarrow \sum_{\mathbf{i}\in\mathbf{CW}} (\mathbf{k}_\tau \omega_\mathbf{i}^2) = \sum_{\mathbf{i}\in\mathbf{CCW}} (\mathbf{k}_\tau \omega_\mathbf{i}^2) \tag{II.3}$$

  where ω_i is the angular velocity of rotor i, and the torque coefficient is k_τ.

- **Motors**: Brushless DC motors are widely used in quadcopters because they offer high efficiency, dependable performance, and the ability to operate at high speeds [38].

$$\mathbf{F_i} = \mathbf{k_f} \cdot \omega_\mathbf{i}^2 \tag{II.4}$$

  where $\mathbf{F_i}$ is the thrust of rotor $\mathbf{i}$, $\mathbf{k_f}$ is a constant depending on the propeller and motor, and $\omega_\mathbf{i}$ is the angular velocity of rotor $\mathbf{i}$.

- **Electronic Speed Controllers (ESCs)**: These components regulate the speed of each motor according to instructions from the flight controller, allowing the quadcopter to generate the necessary thrust and execute movements such as pitch, roll, or yaw [38].

  ESCs control ω_i through PWM (pulse width modulation) signals [41]. The motor speed is a function of the input signal:

$$\omega_{\mathbf{i}} = f(\mathbf{PWM_i}) \tag{II.5}$$

  where:
  - ✓ ω_i: Rotor angular speed
  - ✓ PWM_i: Pulse Width Modulation signal sent to motor i
  - ✓ $f(\cdot)$: Represents the mapping between PWM input and actual speed (often nonlinear)

  ESCs receive PWM signals and convert them into appropriate motor voltages [41]. The function f is typically obtained through calibration, and it directly controls how fast each rotor spins.

- **Propellers**: Propellers create lift, or thrust, depending on how fast they spin [**4**]. The thrust generated by each rotor is directly proportional to the square of its rotational speed, which can be expressed mathematically as:

$$\mathbf{T} = \mathbf{k_f}\omega^2 \tag{II.6}$$
$$\tau = \mathbf{k}_\tau\omega^2 \tag{II.7}$$

  Where:
  - ✓ T is thrust and τ is the reactive torque.
  - ✓ ω is the rotor angular velocity.
  - ✓ k_f, k_τ are empirically derived coefficients.

- **Flight Controller**: Acting as the quadcopter's central control unit, the flight controller oversees all onboard sensors—including the IMU, GPS, and barometer—and processes incoming flight data to adjust motor speeds and keep the system stable [**5**]. It usually implements control algorithms like the PID (Proportional-Integral-Derivative) controller for managing motion along each axis:

$$\mathbf{u(t)} = \mathbf{K_p e(t)} + \mathbf{K_i}\int \mathbf{e(t)}\,dt + \mathbf{K_d}\frac{d\mathbf{e(t)}}{dt} \tag{II.8}$$

  where:
  - ✓ $u(t)$: Control input (e.g., desired motor speed adjustment)
  - ✓ $e(t)$: Error between desired and actual state (e.g., angle, altitude)
  - ✓ K_p, K_i, K_d: Proportional, Integral, and Derivative gains

This equation allows the quadcopter to respond effectively to deviations in position or orientation by applying precise motor adjustments, which enhances both stability and responsiveness during flight [42].

- **Battery**: Lithium polymer (LiPo) batteries are commonly used in quadcopters because of their high energy density [42], allowing for extended flight durations with minimal need for frequent recharging.

  The energy stored in a battery is given by:

$$\mathbf{E} = \mathbf{V} \cdot \mathbf{Q} \tag{II.9}$$

  And the flight time can be estimated as:

$$\mathbf{T_{flight}} = \frac{\mathbf{E}}{\mathbf{P_{avg}}} = \frac{\mathbf{V} \cdot \mathbf{Q}}{\mathbf{P_{avg}}} \tag{II.10}$$

  where:

  - ✓ E: Total energy stored in the battery (in joules or watt-hours)
  - ✓ V: Battery voltage
  - ✓ Q: Battery capacity (in ampere-hours or coulombs)
  - ✓ P_{avg}: Average power consumption during flight
  - ✓ t_{flight}: Estimated flight time

  These equations illustrate the direct relationship between a battery's voltage and capacity and the quadcopter's flight time [42]. To maximize how long it can stay airborne, efficient energy management and a lightweight build are essential.

II.2.2 Degrees of Freedom (DOF)

A quadcopter has 6 Degrees of Freedom (6DOF), This means the quadcopter can move or rotate freely in three-dimensional space, both in terms of its position and its orientation [43]. These degrees of freedom (DOFs) are essential for accurate navigation and effective control. The six DOFs include three translational and three rotational motions. Here's a closer look at each:

A. Translational Movements (Position)

These are movements along the three axes in space and refer to the change in the position of the quadcopter's center of mass (COM) [43].

- **X-axis (Surge)**: Movement forward or backward (along the longitudinal axis of the quadcopter).
- **Y-axis (Sway)**: Movement left or right (along the lateral axis of the quadcopter).
- **Z-axis (Heave)**: Movement upward or downward (along the vertical axis of the quadcopter).

To move in any of these directions, a quadcopter needs to vary the **thrust** generated by its rotors. The equations for translational motion capture the forces acting along each axis, accounting for both the thrust produced and the pull of gravity [43].
Equations of translational motion:

- In the **X-axis** (Surge):

$$\mathbf{F_x} = \mathbf{m} \cdot \ddot{\mathbf{x}} \tag{II.11}$$

- In the **Y-axis** (Sway):

$$\mathbf{F_y} = \mathbf{m} \cdot \ddot{\mathbf{y}} \tag{II.12}$$

- In the **Z-axis** (Heave):

$$\mathbf{F_z} = \mathbf{m} \cdot \ddot{\mathbf{z}} = T - mg \tag{II.13}$$

where:

✓ m is the mass of the quadcopter,
✓ $\ddot{x}$, $\ddot{y}$, $\ddot{z}$ are the accelerations along the X, Y, and Z axes,
✓ T is the total thrust,
✓ g is the acceleration due to gravity.

These movements are controlled by varying the angular velocities of the quadcopter's four rotors, changing the thrust at each rotor [36].

B. Rotational Movements (Attitude)

Rotational movements describe how a quadcopter's orientation shifts around its center of mass. These motions are key to managing the quadcopter's attitude, meaning its angle or tilt relative to the surrounding environment [36].

- **Roll (ϕ)**: Rotation around the X-axis causes the quadcopter to tilt sideways, either to the left or right, which is referred to as **roll**. The roll axis usually runs along the quadcopter's front-to-back direction [36].

- **Pitch (θ)**: Rotation around the Y-axis results in the quadcopter tilting either forward or backward, a motion known as **pitch**. The pitch axis generally runs along the quadcopter's left-to-right direction [36].

- **Yaw (ψ)**: Rotation around the Z-axis causes the quadcopter to spin about its vertical axis, producing a motion known as **yaw** [36].

To manage these rotations, a quadcopter varies the rotational speeds of its individual rotors, generating torques around the X, Y, and Z axes. These torques arise from the differences in how fast the rotors spin and the directions in which they rotate [36].

Equations of rotational motion:

- In the **Roll** (ϕ) direction:

$$\tau_{\mathbf{x}} = l \cdot \mathbf{k_f}(\omega_{\mathbf{2}}^2 - \omega_{\mathbf{4}}^2) \tag{II.14}$$

- In the **Pitch** (θ) direction:

$$\tau_{\mathbf{y}} = l \cdot \mathbf{k_f}(\omega_{\mathbf{3}}^2 - \omega_{\mathbf{1}}^2) \tag{II.15}$$

- In the **Yaw** (ψ) direction:

$$\tau_{\mathbf{z}} = \mathbf{k_m}(\omega_{\mathbf{1}}^2 - \omega_{\mathbf{2}}^2 + \omega_{\mathbf{3}}^2 - \omega_{\mathbf{4}}^2) \tag{II.16}$$

where:

- ✓ l is the distance from the center of the quadcopter to each rotor (typically the length of the arms),
- ✓ K_f is the thrust coefficient,
- ✓ k_m is the torque coefficient,
- ✓ ω_1, ω_2, ω_3, ω_4 are the angular velocities of rotors 1, 2, 3, and 4.

These torque-producing effects are necessary to control the attitude (orientation) of the quadcopter, enabling it to perform actions like rolling, pitching, or yawing.

II.2.3 Coordinate Systems

In quadcopter dynamics, coordinate systems help define the position, velocity, and orientation of the quadcopter within two main frames: the inertial world frame (fixed to the Earth) and the body frame (fixed to the quadcopter itself) [44]. These systems are crucial for accurately modeling and controlling the quadcopter's motion.

A. Inertial (Earth) Frame

The **inertial frame**, also known as the **Earth frame**, is a stationary coordinate system that remains fixed relative to the quadcopter's motion [13]. It is typically aligned with the Earth's surface and serves as a global reference for tracking position and velocity. In this frame:

- The **X-axis** points forward along the longitudinal axis of the quadcopter.
- The **Y-axis** points to the right along the lateral axis.
- The **Z-axis** points upward in the vertical direction (opposite to gravity).

This coordinate system is mainly used to measure the quadcopter's position, velocity, and the forces acting on it from external influences like gravity, wind, and other environmental conditions [13].

B. Body (Quadcopter) Frame

The **body frame** refers to a moving coordinate system that is attached directly to the quadcopter and remains fixed relative to its center of mass (COM) [44]. This frame rotates along with the quadcopter as it changes orientation in space. In this frame:

- The **X-axis** is directed forward, parallel to the front of the quadcopter.
- The **Y-axis** is directed to the right, perpendicular to the X-axis.
- The **Z-axis** is directed downward, opposite the direction of thrust produced by the propellers.

The body frame is used to express the rotational movements of the quadcopter, such as roll, pitch, and yaw, and the forces and torques generated by the rotors.

C. Coordinate Transformation

As the quadcopter rotates through space, the relationship between the inertial frame and the body frame continuously changes [45]. To accurately represent this, a coordinate transformation is used to convert quantities from one frame to the other.

Rotation Matrix

A **rotation matrix** R is employed to convert position and velocity vectors from the body frame to the inertial frame [45]. This matrix is usually constructed using the **Euler angles**—roll ϕ, pitch θ, and yaw ψ—which define the quadcopter's orientation with respect to the inertial frame.

The rotation matrix R in Active Rotation is defined as:

$$\mathbf{R_{xyz}} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi - \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{II.17}$$

where:

- ✓ ϕ is the **roll** angle (rotation about the X-axis),
- ✓ θ is the **pitch** angle (rotation about the Y-axis),
- ✓ ψ is the **yaw** angle (rotation about the Z-axis),
- ✓ C and S represent **cosine** and **sine** functions, respectively.

This matrix allows you to rotate a vector from the body frame to the inertial frame.

Transformation of Position

If $\mathbf{r}_B = [x_B, y_B, z_B]^T$ is a position vector in the body frame, the corresponding position $\mathbf{r}_E = [x_E, y_E, z_E]^T$ in the inertial frame is given by:

$$\mathbf{r}_E = R \cdot \mathbf{r}_B \tag{II.18}$$

where R is the rotation matrix that converts from the body frame to the inertial frame.

Transformation of Velocity

The velocity in the body frame $\mathbf{v}_B = [v_x, v_y, v_z]^T$ can be related to the velocity in the inertial frame $\mathbf{v}_E = \left[v_x^{'}, v_y^{'}, v_z^{'}\right]^T$ using the same rotation matrix [46]:

$$\mathbf{v}_E = R \cdot \mathbf{v}_B \tag{II.19}$$

Transformation of Angular Velocity

The **angular velocity** vector ω (comprising roll, pitch, and yaw rates) in the body frame can also be transformed into the inertial frame. The angular velocity vector $\omega_B = [p, q, r]^T$ is related to the angular velocity in the inertial frame $\omega_E = \left[p^{'}, q^{'}, r^{'}\right]^T$ by the following equation [42]:

$$\omega_E = R \cdot \omega_B \tag{II.20}$$

where:

- ✓ p is the **roll rate** (rate of change of roll angle),
- ✓ q is the **pitch rate** (rate of change of pitch angle),
- ✓ r is the **yaw rate** (rate of change of yaw angle).

This relationship is crucial for the control of the quadcopter's orientation in space.

II.3 Quadcopter Mathematical Modeling

Mathematical modeling plays a key role in understanding and analyzing the complex behavior of quadcopters [37]. By offering a detailed and accurate representation of the vehicle's motion and the forces acting upon it, modeling allows engineers and researchers to develop robust and efficient control systems [37]. This section explores the fundamental elements of quadcopter modeling, with a focus on Kinematics, Dynamics, and Forces and Torques. Each of these aspects is essential for describing how a quadcopter moves and operates within three-dimensional space.

II.3.1 Kinematics

Kinematics focuses on the geometric aspects of motion, without taking into account the forces that generate it. In the case of a quadcopter, this involves describing its position, orientation, and velocities within both the inertial and body-fixed coordinate frames [46]. The purpose of kinematic modeling is to understand how the quadcopter moves and how its different components are related to each other throughout its motion.

A. Coordinate Frames

- **Inertial Frame (Earth Frame)**: This frame remains fixed relative to the Earth [40] and is commonly used as a global reference. It serves as the foundation for measuring the quadcopter's absolute position and orientation [46].

- **Body Frame**: This moving coordinate system, fixed to the quadcopter, makes it easier to describe its dynamics. The origin is located at the quadcopter's center of mass, and the axes are aligned with its structural layout [46].

B. Position and Orientation

The quadcopter's state in 3D space is typically expressed through:

- A position vector $\mathbf{p} = [x\ y\ z]^T$ indicating the quadcopter's location in the inertial frame.
- Three orientation angles - roll (ϕ), pitch (θ), and yaw (ψ) - which describe its rotation around the x, y, and z axes, respectively. These angles collectively define the attitude of the quadcopter.

C. Rotation Matrix ($\mathbf{R_{xyz}}$)

To convert between coordinate systems, particularly from the body frame to the inertial frame, a rotation matrix R_{xyz}, as defined in equation (II.17), is used. This matrix is derived from a sequence of three rotations that correspond to yaw, pitch, and roll.
This transformation is essential for both control and simulation, as it links the quadcopter's internal control signals (expressed in the body frame) to its actual motion in the external environment (inertial frame) [42].

D. Velocity Relationships

In addition to position and orientation, it's necessary to track how quickly these variables change:

- Linear velocity ($\dot{\mathbf{p}}$) tells us how the position changes over time.
- Angular velocity (ω_B) describes how the quadcopter's orientation changes.

$$\omega_{\mathbf{B}} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{II.21}$$

This matrix [42] helps relate angular velocities in the body frame to the rate of change of Euler angles, allowing for precise control of orientation.

II.3.2 Dynamics

Dynamics focuses on the forces and torques that drive the quadcopter's motion. By modeling these dynamics, we can predict how the vehicle will react to control commands, external disturbances, and environmental factors such as gravity and air resistance [6].

A. Newton-Euler Formalism

This approach uses Newton's second law for translational motion and Euler's equations for rotational motion [45].

Translational Dynamics

$$\mathbf{m}\ddot{\mathbf{p}} = \mathbf{F_g} + \mathbf{R_{xyz}}\mathbf{F_T} + \mathbf{F_d} \tag{II.22}$$

This equation shows how the quadcopter's mass and acceleration are affected by gravitational force, thrust from the rotors (transformed to the inertial frame), and aerodynamic drag. It explains vertical lift and general motion through space [47].

Rotational Dynamics

$$\mathbf{I}\dot{\omega} + \omega \times (\mathbf{I}\omega) = \tau \tag{II.23}$$

This equation explains how the quadcopter rotates in response to applied torques and its moment of inertia [47]. Each rotational axis —roll, pitch, and yaw— has a corresponding equation that governs its motion:

$$\tau_\phi = \mathbf{I_x}\dot{\mathbf{p}} + (\mathbf{I_z} - \mathbf{I_y})qr \tag{II.24}$$

$$\tau_\theta = \mathbf{I_y}\dot{\mathbf{q}} + (\mathbf{I_x} - \mathbf{I_z})pr \tag{II.25}$$

$$\tau_\psi = \mathbf{I_z}\dot{\mathbf{r}} + (\mathbf{I_y} - \mathbf{I_x})pq \tag{II.26}$$

These equations capture gyroscopic coupling between axes and highlight the nonlinear nature of the system.

B. Euler-Lagrange Formalism

The Euler-Lagrange method, based on energy principles, offers a powerful way to symbolically derive the dynamics of complex systems [41]. It serves as an alternative to the Newton-Euler method for obtaining the same equations of motion, using the relation $L = T - V$ [48].

In this formulation, the kinetic energy T includes both linear and rotational motion, while the potential energy V is mainly attributed to gravity. The generalized coordinates q_i are used to represent the system's position and orientation states [48].

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \tag{II.27}$$

This formalism is advantageous when analyzing multi-body systems or when extending models to include more complex interactions.

II.3.3 Forces and Torques

Controlling a quadcopter involves adjusting the speed of each rotor, which in turn alters the thrust and torques they produce [46]. By modifying these forces, the quadcopter can move in various directions and rotate around its different axes.

A. Thrust Force

$$\mathbf{F_i} = \mathbf{k_f}\omega_\mathbf{i}^2 \tag{II.28}$$

Each rotor's thrust is a function of its angular velocity squared [41]. The total thrust acting on the quadcopter is the sum of the thrusts from all four rotors:

$$\mathbf{T} = \sum_{\mathbf{i}=1}^{4} \mathbf{F_i} = \mathbf{k_f}(\omega_\mathbf{1}^2 + \omega_\mathbf{2}^2 + \omega_\mathbf{3}^2 + \omega_\mathbf{4}^2) \tag{II.29}$$

This thrust opposes gravity and allows vertical motion or hovering.

B. Torques from Rotors

Different combinations of rotor speeds produce torque about the x, y, and z axes, leading to roll, pitch, and yaw movements respectively [41].

$$\tau_\phi = l \cdot \mathbf{k_f}(\omega_\mathbf{4}^2 - \omega_\mathbf{2}^2) \tag{II.30}$$

$$\tau_\theta = l \cdot \mathbf{k_f}(\omega_\mathbf{3}^2 - \omega_\mathbf{1}^2) \tag{II.31}$$

$$\tau_\psi = \mathbf{k_m}(\omega_\mathbf{1}^2 - \omega_\mathbf{2}^2 + \omega_\mathbf{3}^2 - \omega_\mathbf{4}^2) \tag{II.32}$$

Where:

- ✓ l: Arm length from the center to each rotor.
- ✓ k_f: Thrust coefficient.
- ✓ k_m: Moment (drag) coefficient.

These equations demonstrate how fine-tuned control of motor speeds results in targeted changes in attitude and translational movement [46]. A detailed and accurate representation of these interactions is essential for developing control systems that ensure the quadcopter remains both stable and responsive.

All of these components combine to create a complete and nonlinear mathematical model of the quadcopter. This model forms the basis for simulation environments, controller development, and performance evaluation in practical applications.

II.4 Aerodynamic Modeling

Accurate aerodynamic modeling is essential for understanding how a quadcopter interacts with the surrounding air during flight [49].

Beyond the immediate effects of rotor thrust, aerodynamic forces and moments significantly influence the vehicle's stability, control, and overall dynamics, particularly during high-speed maneuvers or when exposed to external disturbances such as wind gusts. While simpler models may assume the quadcopter operates in a vacuum, incorporating aerodynamic effects offers a more realistic view of its behavior[46][49], which is critical for designing reliable controllers and optimizing performance. This section explores the key aerodynamic phenomena that affect quadcopter operation.

II.4.1 Drag Forces

The main aerodynamic force acting on a quadcopter during translational motion is aerodynamic drag. This force resists the relative movement between the quadcopter and the surrounding air, increasing with velocity and significantly affecting both flight dynamics and energy usage. Mathematically, the translational drag force $\mathbf{F_d}$ is given by:

$$\mathbf{F_d} = -\mathbf{k_d}\dot{\mathbf{p}} \tag{II.33}$$

Where k_d is a positive-definite matrix that contains the drag coefficients for each principal axis (x, y, and z), while $\dot{p}$ represents the linear velocity vector of the quadcopter's center of mass in the inertial frame [49]. The negative sign shows that drag acts in the opposite direction of motion.

For the range of flight speeds typically encountered in quadcopter operations, it's often acceptable to model drag as being directly proportional to velocity [49]. However, as flight speed increases, a quadratic model becomes more accurate, capturing the nonlinear behavior of air resistance [46]. Drag forces introduce damping into the system, which can help by reducing oscillations and contributing to stability, or complicate control by introducing disturbances, especially during rapid maneuvers [50]. Being able to understand and estimate drag forces accurately is crucial for following precise flight paths and optimizing energy use [41].

II.4.2 Rotor Aerodynamics

Beyond translational drag, the spinning rotors interact with the surrounding air in complex ways that affect the overall forces and torques produced by the quadcopter [47]. These rotor-related aerodynamic effects can be broken down into several key components:

- **Induced Drag**: When the rotors generate lift, they also create a downward airflow known as downwash, which leads to induced drag [51]. This effect, explained by momentum theory [52], reflects an inherent energy cost that comes from thrust.
- **Profile Drag**: Profile drag results from friction and the pressure differences around the rotor blades as they spin through the air [51]. This type of drag increases with the square of the blade's rotational velocity and becomes more pronounced at higher angular speeds.

- **Blade Flapping**: At high forward speeds, asymmetries in airflow across the rotor disc can cause the blades to flap upward and downward [51]. This leads to variations in lift distribution, affecting the stability and controllability of the vehicle, especially during rapid translational motion.

For many control and estimation tasks, simplified aerodynamic models often group rotor-related effects into empirical drag coefficients.

Still, when it comes to high-fidelity simulations or the development of advanced autonomous systems, more detailed modeling techniques are used [49]. Approaches like Blade Element Momentum Theory (BEMT) or full Computational Fluid Dynamics (CFD) simulations offer a deeper look into rotor airflow behavior [53]. These methods help capture complex phenomena such as vortex ring states or rotor-ground interactions, which can significantly influence both safety and flight performance.

As summary, the rotor aerodynamics introduces key nonlinearities and interdependencies into the system's behavior, making it essential to account for them carefully during both the modeling process and control system design.

II.5 Control Inputs and State Variables

When controlling quadcopters, it's important to clearly define how the actions taken by the actuators (motors and propellers) influence the system's behavior [54] To approach this in a structured way, two key components must be identified: the control inputs, which are the variables the controller adjusts [54], and the state variables, which describe the system's current condition at any moment. A solid grasp of both is essential for analyzing the system, designing effective control strategies, planning trajectories, and implementing feedback mechanisms [55].

II.5.1 Control Inputs

As a dynamic system, the quadcopter operates within a six degrees-of-freedom (6-DOF) space but relies on only four primary control inputs. These inputs consist of the total thrust and three separate moments (torques) applied around the main body axes.

The control input vector is expressed as:

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{T} \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{II.34}$$

where:

- ✓ T is the total vertical thrust from the four rotors,
- ✓ τ_ϕ represents the **roll moment** (rotation around the body's x-axis),
- ✓ τ_θ represents the **pitch moment** (rotation around the body's y-axis),
- ✓ τ_ψ represents the **yaw moment** (rotation around the body's z-axis).

These control inputs are functions of the individual rotor angular speeds ω_i [53]. The mapping from motor speeds to control inputs is often described using a control allocation or mixing matrix, which accounts for the quadcopter's geometry and motor configuration [53].

In particular, the thrust T is proportional to the sum of the squares of the rotor speeds, while the torques $\tau_\phi, \tau_\theta, \tau_\psi$ arise from differences in those speeds [54]. A precise mapping between these control inputs and the rotor angular velocities ensures that high-level flight commands are accurately translated into motor-level actions. Each input depends nonlinearly on the individual rotor speeds $\omega_1, \omega_2, \omega_3, \omega_4$ [56], and this relationship is commonly expressed as:

$$\begin{bmatrix} \mathbf{T} \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \mathbf{k_f} & \mathbf{k_f} & \mathbf{k_f} & \mathbf{k_f} \\ 0 & l\mathbf{k_f} & 0 & -l\mathbf{k_f} \\ -l\mathbf{k_f} & 0 & l\mathbf{k_f} & 0 \\ \mathbf{k_m} & \mathbf{k_m} & \mathbf{k_m} & \mathbf{k_m} \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{II.35}$$

where:

- ✓ k_f is the thrust coefficient,
- ✓ k_m is the moment (torque) coefficient,
- ✓ l is the distance from the quadcopter's center to each rotor.

This matrix relationship, commonly known as the [57], plays a key role in control allocation. It enables the translation of desired forces and torques into specific commands for each motor, ensuring the quadcopter responds accurately to control inputs.

II.5.2 State Variables

The full dynamical state of the quadcopter at any given moment is represented by a **state vector**. This vector includes all the key translational and rotational kinematic variables needed to describe how the system evolves with time [58].
The state vector $\mathbf{x}$ is formulated as [58]:

$$\mathbf{x} = [\mathbf{x}\ \mathbf{y}\ \mathbf{z}\ \phi\ \theta\ \psi\ \dot{\mathbf{x}}\ \dot{\mathbf{y}}\ \dot{\mathbf{z}}\ \mathbf{p}\ \mathbf{q}\ \mathbf{r}]^T \tag{II.36}$$

with each component explained as follows:

- $(\mathbf{x}, \mathbf{y}, \mathbf{z})$: Position coordinates of the quadcopter's center of mass in the inertial (world-fixed) frame,
- (ϕ, θ, ψ): Roll, pitch, and yaw angles describing the orientation of the body frame relative to the inertial frame,
- $(\dot{\mathbf{x}}, \dot{\mathbf{y}}, \dot{\mathbf{z}})$: Linear velocity components in the inertial frame,
- $(\mathbf{p}, \mathbf{q}, \mathbf{r})$: Angular velocity components about the body-fixed x, y, and z axes, respectively.

Euler angles provide a compact and intuitive means of representing orientation, making them a practical choice in many quadcopter applications [59]. However, this form of parametrization introduces the possibility of singularities, commonly referred to as gimbal lock, particularly when the pitch angle approaches ±90 degrees [59]. Despite this limitation, Euler angles remain effective for moderate flight maneuvers typically encountered during standard quadcopter operation [59]. Their straightforward nature supports efficient control and estimation. In more complex scenarios, especially those involving aggressive maneuvers or fully three-dimensional trajectory tracking, advanced control frameworks may adopt alternative orientation representations such as quaternions or rotation matrices [59]. These alternatives avoid singularities and allow for smoother and more reliable rotational control. The complete 12-dimensional state space of the quadcopter offers a robust foundation for implementing advanced control strategies [59]. These may include feedback linearization, nonlinear control methods, model predictive control (MPC), or robust optimal control approaches, each leveraging the system's full dynamics to improve performance and maintain stability.

II.6 State-Space Representation

An essential and foundational step in designing modern control systems for aerial vehicles such as quadcopters involves reformulating their inherently nonlinear dynamics into a state-space model [58]. This formalism serves several critical purposes. First, it provides a rigorous mathematical description of the system's behavior by capturing the evolution of internal states and the generation of outputs through first-order differential equations [58]. Second, the state-space representation facilitates the systematic implementation of a broad range of control strategies [56]. Classical linear methods, including proportional - integral - derivative (PID) control, pole placement, and linear quadratic regulators (LQR), depend heavily on linear state-space formulations [56]. Moreover, this modeling approach establishes the foundation for advanced control techniques such as nonlinear control, optimal control, robust control, and model predictive control (MPC) [56]. These methodologies are particularly important when the control objective involves ensuring stability, robustness, and high performance in the presence of external disturbances, system uncertainties, or operational constraints [49]. Therefore, constructing an accurate and reliable state-space model is not merely a theoretical requirement but a practical necessity for achieving dependable, high-performance quadcopter operation across a wide range of mission scenarios [49].

II.6.1 General Nonlinear State-Space Formulation

At its core, the motion of a quadcopter is governed by a system of coupled, nonlinear differential equations. These equations are derived by applying the principles of rigid-body dynamics using the Newton-Euler framework, which simultaneously accounts for both translational and rotational motion [56].

To systematically organize and analyze these complex dynamics, a nonlinear state-space representation is typically adopted [54]. This form offers a structured mathematical framework through which the behavior of the quadcopter can be rigorously modeled and controlled.
The general nonlinear state-space model is written as:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} &= \mathbf{g}(\mathbf{x}, \mathbf{u}) \end{aligned} \tag{II.37}$$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the **state vector**, aggregating all quantities necessary to fully describe the system's current condition at any time t, we find $\mathbf{x}$ in equation (II.36).
- $\mathbf{u} \in \mathbb{R}^m$ is the **control input vector**, containing all externally applied inputs, typically the collective thrust and body torques.
- $\mathbf{y} \in \mathbb{R}^p$ is the **output vector**, representing the measurable quantities of interest (e.g., position, attitude angles).
- $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the **state transition function**, capturing how the system's states evolve over time.
- $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ is the **output function**, relating the internal state and inputs to the outputs.

A. Structure of the Nonlinear Functions $\mathbf{f}(\mathbf{x}, \mathbf{u})$ and $\mathbf{g}(\mathbf{x}, \mathbf{u})$

The function $f(x, u)$ typically involves:

- **Translational Dynamics**: Describing the acceleration of the center of mass based on applied thrust forces, gravitational effects, and aerodynamic drag [54].
- **Rotational Dynamics**: Describing the evolution of the orientation based on body torques, rotational inertia, and gyroscopic effects [54].
  These equations are highly nonlinear due to [54]:
  - ✓ Trigonometric relations in rotation matrices,
  - ✓ Coupling between translational and rotational motions,
  - ✓ Dependence of aerodynamic forces on velocity and orientation.

  The function $g(x, u)$ depends on the type of sensors available. For instance:
  - ✓ If GPS and IMU data are used, outputs could include (x, y, z) and (ϕ, θ, ψ).
  - ✓ If only inertial measurements are available, outputs could be limited to acceleration and angular rates.

B. Advantages of the Nonlinear State-Space Representation

- **Complete Description**: It models the full system dynamics without simplifying assumptions about small angles or low velocities [56].
- **Foundational for Nonlinear Control**: Techniques such as feedback linearization, sliding mode control, and nonlinear observers directly operate on these full-order models [56].
- **Essential for High-Fidelity Simulation**: Simulating the real-world behavior of quadcopters under aggressive maneuvers, environmental disturbances, and actuator saturation requires the original nonlinear form [56].

However, while powerful, nonlinear models are computationally demanding and analytically complex, which motivates the need for local linearization when designing controllers for specific flight conditions, like hovering.

II.6.2 Linearized State-Space Model (Hover Condition)

In practical applications, directly handling the nonlinear dynamics of a quadcopter often proves to be computationally intensive and analytically intractable for control design [57]. To address this challenge, a widely adopted engineering approach involves **linearizing** the nonlinear equations of motion around a specific operating point [57]. This is typically the **hover condition**, in which the quadcopter maintains a stationary and level orientation in space. Linearization around this equilibrium enables the use of well-established linear control techniques while preserving sufficient fidelity for many control and estimation tasks [57].
The **hover condition** is defined by:

- ✓ Small attitude angles: $\phi, \theta, \psi \approx 0$ (measured in radians),
- ✓ Negligible linear velocities: $\dot{x}, \dot{y}, \dot{z} \approx 0$,
- ✓ Zero angular rates: $p, q, r \approx 0$,
- ✓ Constant thrust balancing gravitational force: $T \approx mg$.

Under these simplifying assumptions, the **nonlinearities** introduced by trigonometric functions such as $\sin(\theta)$ and $\cos(\phi)$, along with the coupling effects between translational and rotational dynamics, can be approximated using first-order Taylor series expansions [60]. This process results in a **linear time-invariant (LTI)** model, which significantly streamlines both the analytical treatment of the system and the design of control strategies [49].

The resulting **linearized** state-space model is represented as follows:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

A. Structure and Physical Interpretation of A, B, C, D Matrices

- **A Matrix (System Matrix)**:

  The matrix A captures how the current states of the system influence their own rates of change [54]. For a quadcopter:

  ✓ The position states (x, y, z) are directly influenced by their corresponding velocities $(\dot{x}, \dot{y}, \dot{z})$.
  ✓ The rotational angles (ϕ, θ, ψ) are influenced by angular rates (p, q, r).
  ✓ Gravity introduces couplings between attitude angles and translational accelerations. For instance, a small roll angle ϕ creates a lateral force proportional to $g\phi$, moving the quadcopter sideways.

- **B Matrix (Input Matrix)**:

  The B matrix describes how the control inputs - collective thrust and body torques - affect the state [54], derivatives:

  ✓ The collective thrust input directly affects the vertical acceleration $\dot{z}$.
  ✓ Roll, pitch, and yaw torques affect the angular accelerations p, q, and r based on the quadcopter's moments of inertia I_x, I_y, and I_z.

- **C Matrix (Output Matrix)**:

  The matrix C maps the internal states to the measurable outputs. In most basic designs, it is assumed that all the states are directly measurable (full-state feedback), thus C becomes an identity matrix [54].

- **D Matrix (Feedthrough Matrix)**:

  Typically, D is a zero matrix because there is no direct, instantaneous effect of the control input on the output without passing through system dynamics [54].

B. State Space Matrices for an X-Configuration Quadcopter

Let us assume the quadcopter is symmetric, with mass m, gravitational acceleration g, and inertia matrix:

$$\mathbf{I} = \text{diag}(\mathbf{I_x}, \mathbf{I_y}, \mathbf{I_z}) \tag{II.38}$$

The linearized dynamics around hover lead to approximate matrices of the form:

- **A Matrix**:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{G} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{II.39}$$

where **G** accounts for gravitational coupling effects between roll and pitch angles and linear accelerations.

$$G = \begin{bmatrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{II.40}$$

- **B Matrix**:

$$\text{which } \mathbf{B_m} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{\mathbf{1}}{\mathbf{m}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \text{ and } \mathbf{B_f} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\mathbf{1}}{\mathbf{I_x}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\mathbf{1}}{\mathbf{I_y}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\mathbf{1}}{\mathbf{I_z}} \end{bmatrix} \tag{II.41}$$

Here, $\mathbf{B_f}$ relates the thrust and torque inputs to the linear and angular accelerations, including the inverse mass of inertia in $\mathbf{B_m}$, with $B = \begin{bmatrix} B_m & B_f \end{bmatrix}^T$ [54].

- **C Matrix**:

$$\mathbf{C} = \mathbf{I}_{12} = \text{eye}(12) \tag{II.42}$$

- **D Matrix**:

$$\mathbf{D} = \mathbf{0}_{12\times 4} \tag{II.43}$$

C. Mathematical Derivation Approach

To obtain these matrices rigorously, the following steps are performed:

- Define the full nonlinear dynamics (from Newton-Euler equations).
- Select the equilibrium point — usually, a stationary hover at a constant altitude.
- Compute Jacobians:
  - ✓ $A = \frac{\partial f}{\partial x}\Big|_{x_0,u_0}$
  - ✓ $B = \frac{\partial f}{\partial u}\Big|_{x_0,u_0}$
  - ✓ $C = \frac{\partial g}{\partial x}\Big|_{x_0,u_0}$
  - ✓ $D = \frac{\partial g}{\partial u}\Big|_{x_0,u_0}$

This Jacobian linearization preserves the local behavior near the hover point and is an essential bridge toward model-based controller design.

D. Importance of Linearization for Controller Design

Linearization plays a central role in quadcopter control design, especially in advanced academic and engineering contexts where analytical precision and system reliability are crucial [49]. By approximating the nonlinear dynamics near an operating point such as hover, a linear time-invariant (LTI) model can be derived, enabling the application of classical control methods [49].

- **Simplification of Complex Dynamics**:

  The original nonlinear dynamics of a quadcopter are intrinsically coupled and exhibit multi-input multi-output (MIMO) behavior, influenced by nonlinear forces including gravity, aerodynamic drag, and gyroscopic effects [54]. Designing controllers directly for these nonlinear equations poses significant mathematical and computational challenges. To address this, the system is often linearized around a specific operating point, such as the hover condition [55]. This process simplifies the dynamic model while preserving the key characteristics required for effective control system design.

- **Enabling Classical and Modern Control Techniques**:

  Linear systems theory provides a well-established framework with powerful analytical and design tools, all of which depend on the availability of a linear model [58]. Once the system has been linearized, a range of control techniques can be applied, including:

  ✓ **Pole Placement**: Assigning closed-loop eigenvalues to ensure desired system stability and dynamic response [56].
  ✓ **Linear Quadratic Regulator (LQR)**: Designing an optimal controller by balancing control performance with actuation energy [56].
  ✓ **Model Predictive Control (MPC)**: Utilizing the linear dynamics to predict future behavior and optimize control sequences over a horizon [49].
  ✓ **Observer Design**: Estimating unmeasured states, such as velocities or disturbances, using Kalman filters or Luenberger observers [55].

  Without linearization, these control methods would be inapplicable in their standard form or would necessitate advanced nonlinear extensions, which are typically more complex to implement, analyze, and fine-tune.

- **Facilitating Stability Analysis**:

  Stability is a fundamental requirement for quadcopter operation. The linearized system allows for the use of eigenvalue and eigenvector analysis [59]:

  ✓ The location of poles in the complex plane directly reveals whether the system is stable, marginally stable, or unstable.
  ✓ Dominant poles indicate the speed and damping of the system's response.

  This analytic capability is crucial for both theoretical validation and practical tuning of the controller.

- **Control-Oriented Modeling**:

  In practical engineering, controller design typically follows an iterative cycle of modeling, simulation, and experimental verification. Linearized models act as efficient initial approximations, offering simplicity and clarity for early-stage analysis and controller prototyping [61].

- **Designing for Robustness and Performance**:

  Linearization facilitates the assessment of robustness margins, including gain margin, phase margin, and sensitivity functions. These metrics are vital for verifying that the quadcopter remains stable and responsive in the presence of model inaccuracies, external disturbances such as wind, and sensor noise, all while continuing to meet its mission requirements [61].

- **Extending to Advanced Control Frameworks**:

  Once a linear model is established, it becomes straightforward to extend toward more sophisticated approaches such as [59]:

  ✓ **Gain scheduling**: Designing multiple linear controllers around different operating points and switching between them as the operating conditions change [59].

  ✓ **Nonlinear control via linearization**: Using linear techniques locally, while recognizing the need for global nonlinear controllers over larger maneuvering envelopes [59].

  Thus, linearization serves not only as a practical control design tool but also as a bridge between basic control theory and advanced aerospace control methodologies.

II.6.3 Interpretation and Utility

Recasting quadcopter dynamics into the **state-space formulation**, especially in its linearized form, is a fundamental step in both analyzing and controlling aerial vehicles. This representation serves as a vital link between the complex, nonlinear behavior of quadrotor systems and the structured frameworks of modern control theory. A clear grasp of the state-space approach is essential for applying robust and optimal control techniques effectively [62].

A. Interpretation of the State-Space Matrices

Each matrix in the state-space model carries a clear physical and mathematical significance:

- **System Matrix A**: The state matrix A captures the system's internal dynamics, showing how states change over time without any input. It reflects inherent couplings, like how small roll angles cause lateral movement due to gravity, illustrating how tilting results in horizontal motion [63].

- **Input Matrix B**: Captures how external control actions (thrust and body torques) influence the rate of change of the states. It shows, for instance, how an increase in thrust leads to vertical acceleration, or how a torque around the roll axis induces angular acceleration [63].

- **Output Matrix C**: Identifies the portions of the internal state that can be measured through available sensors. In full-state feedback configurations, such as those utilizing high-precision IMU and GPS systems, C is often set as the identity matrix. In other cases, it may represent a mapping that captures only selected states, such as positions and yaw angles, corresponding to the outputs being monitored [63].
- **Feedthrough Matrix D**: Represents how inputs influence outputs instantly, without involving system dynamics. In typical quadcopters, D is usually a zero matrix, as motor speed changes affect states over time [63].

B. Utility in Control System Design

The benefits of using the state-space model in quadcopter control design are multi-faceted:

- **Structured Mathematical Framework**:

  State-space models enable a structured treatment of quadcopter dynamics through linear algebraic techniques. This framework is essential for developing state-feedback controllers, optimal estimators, or system observers [63].

- **Facilitation of Advanced Controllers**:

  Methods like **Linear Quadratic Regulator (LQR)**, **Backstepping Control**, and Model Predictive Control **(MPC)**, fundamentally depend on the state-space representation of the system [64]. These approaches are designed to ensure stability, maintain desired performance, and provide robustness against both model uncertainties and external disturbances.

- **Design of Observers**:

  In practical scenarios, not all states can be directly measured. For instance, angular rates are typically sensed, while velocities often require estimation. State observers such as Luenberger observers or Kalman filters use the A and C matrices to reconstruct the full state vector from available measurements [62].

- **Stability and Performance Analysis**:

  The eigenvalues of the A matrix, which represent the poles of the system, provide direct insight into its stability properties [63]. When all eigenvalues have negative real parts, the linearized model around the hover condition is considered stable [63].

C. Limitations and Practical Considerations

Despite its advantages, the linear state-space model has inherent limitations:

- **Valid Only Locally**: The linearized model provides an accurate description of the system dynamics only in the vicinity of the chosen operating point, typically hover. When the system experiences large deviations from this point, the linear approximation becomes less valid, leading to substantial modeling errors [63].
- **Neglect of Actuator Dynamics**: Not all states can be directly measured in practice. For example, angular rates can be sensed, but velocities may need to be estimated. By adding state observers such as Luenberger observers or Kalman filters, the A and C matrices are used to reconstruct the full state vector from measured outputs [63].
- **Environmental Effects**: Factors like wind disturbances, ground effects, and aerodynamic couplings are typically excluded from basic linear models. To manage these influences, separate mechanisms such as disturbance observers or robust control strategies are often incorporated [63]. Nevertheles, the state-space framework continues to serve as the cornerstone for nearly all quadcopter control architectures, ranging from basic PID implementations in undergraduate projects to advanced nonlinear and adaptive strategies explored in academic research [63].

A deep understanding of the state-space model, including its structure, interpretation, and practical limitations, equips engineers and researchers with a powerful toolkit. This understanding provides the mathematical basis for designing controllers that guarantee the stability, responsiveness, and robustness of quadcopter platforms.

II.7 Actuator Modeling for Quadcopter System

A comprehensive quadcopter model must account not only for the rigid-body dynamics of the vehicle but also for the behavior of its actuators [65]. Motors and propellers are essential components that directly determine the quadcopter's ability to produce thrust and torques [65]. As a result, accurate actuator modeling is vital for achieving precise control, maintaining stability, and ensuring reliable trajectory tracking, especially in aggressive or high-precision flight. Although the vehicle's overall behavior is shaped by its inertial properties and aerodynamics, it is the actuators that ultimately generate the necessary forces and moments. Ignoring actuator dynamics can compromise controller performance, reduce stability, or even result in a loss of control [65].

II.7.1 Motor and Propeller Dynamics

The dynamics of the actuators -motors and propellers- represent a critical connection between the control commands generated by the flight controller and the resulting forces and moments exerted on the quadcopter [57]. For high-performance scenarios, including aggressive maneuvers, accurate trajectory tracking, or effective disturbance rejection, accurately understanding and modeling these dynamics is essential [53].

A. Motor Dynamics

Each quadcopter rotor is driven by a Brushless DC motor**(BLDC)**, chosen for its high efficiency, low mass, and superior power-to-weight ratio [59]. The motor dynamics characterize how rapidly the motor's rotational speed responds to variations in control input (e.g., voltage or PWM command) [62].

A widely accepted simplified model for a BLDC motor is a first-order differential equation:

$$\tau_{\mathbf{m}} \frac{d\omega_{\mathbf{m}}}{dt} + \omega \mathbf{m} = \mathbf{k_m u_m} \tag{II.44}$$

where:

- ✓ $\omega_{m(t)}$ [rad/s] is the instantaneous motor angular velocity,
- ✓ $u_{m(t)}$ is the control input signal (normalized PWM, voltage, or current command),
- ✓ τ_m [s] is the motor's time constant, reflecting the electrical and mechanical inertias,
- ✓ k_m is a static motor gain relating steady-state input to speed.

B. Physical Interpretation

- **Time Constant** τ_m: Indicates how fast the motor reacts to a command change. A small τ_m means fast response; a large τ_m implies sluggish behavior [62].
- **Motor Lag**: Real motors cannot instantaneously achieve the desired speed due to their inductance, inertia, and friction [62].
- **Steady-State Behavior**: At steady state ($\dot{\omega}_m = 0$), the motor speed satisfies:

$$\omega \mathbf{m} = \mathbf{k_m u_m} \tag{II.45}$$

meaning the speed is proportional to the input, scaled by $\mathbf{k_m}$.

C. Higher-Order Effects (if needed)

In more precise models (used in high-end research or industrial UAVs), additional elements could be incorporated:

- **Motor inductance and back-EMF** leading to a second-order system,
- **Torque saturation** at high speeds,
- **Temperature effects** reducing motor efficiency.

However, for most controller design tasks, the first-order model offers a very practical trade-off between simplicity and accuracy [62].

D. Propeller Thrust and Torque Characteristics

The propeller, mechanically coupled to the motor, converts rotational motion into thrust and reactive torque through aerodynamic forces. The relationship between motor speed and these forces is inherently nonlinear, but can be effectively approximated as [53]:

$$\begin{aligned} \mathbf{T_i} &= \mathbf{k_T}\omega^2_{\mathbf{m,i}} \\ \mathbf{Q_i} &= \mathbf{k_Q}\omega^2_{\mathbf{m,i}} \end{aligned} \tag{II.46}$$

where:

- ✓ T_i [N] is the thrust produced by propeller i,
- ✓ Q_i [Nm] is the aerodynamic torque (drag) opposing motor rotation,
- ✓ $k_T \left[\frac{N \cdot s^2}{rad^2}\right]$ is the thrust coefficient,
- ✓ $k_Q \left[N \cdot m \cdot \frac{s^2}{rad^2}\right]$ is the torque coefficient,
- ✓ $\omega_{m,i}$ is the motor speed for rotor i.

E. Combined Actuator Behavior

When motor dynamics and propeller aerodynamics are considered together:

- **Motor Lag → Delayed Force Production**:

  A command to increase thrust will not produce an immediate increase in force; there will be a transient phase governed by τ_m.

- **Nonlinear Mapping**:

  The relation between control input u_m and output forces/moments is not linear but quadratic in ω_m, and subject to a **first-order dynamic delay**.

- **Importance for Control Design**:

  - ✓ Ignoring motor lag can lead to overshoots, poor tracking, or even instability.
  - ✓ Assuming linear force production could mislead the controller about actuator authority.
  - ✓ Advanced controllers may explicitly compensate for actuator dynamics (e.g., by inverse dynamics, feedforward terms, or observer-based estimation).

The actuator modeling through motor and propeller dynamics is vital for guaranteeing that the designed controllers operate reliably and safely. It captures real-world constraints and nonlinearities that remain hidden in idealized rigid-body models [63].

II.7.2 Actuator Constraints and Saturation

In practical quadcopter operations, actuators are **not ideal**. They encounter multiple **physical, electrical, and mechanical limitations** that must be thoroughly addressed in both modeling and control design. Recognizing actuator constraints is **critical** for designing controllers that are both **realistic** and **robust** [59].

A. Types of Actuator Constraints

- **Maximum and Minimum Motor Speeds**

  Each motor has an operational range: $\omega_{\min} \leq \omega_{\mathbf{m}} \leq \omega_{\max}$.

  ✓ $\omega_{\min}$ the minimum speed needed to produce enough thrust to keep the quadcopter airborne (may not be exactly zero due to friction and dead zone effects).

  ✓ $\omega_{\max}$ the maximum safe speed before mechanical failure, excessive heating, or power system limits are reached.

  Implication for Control:

  ✓ Commands resulting in $\omega_m > \omega_{\max}$ must be **saturated** (clamped) to $\omega_{\max}$.

  ✓ Operating close to $\omega_{\max}$ reduces actuator authority and can cause **control performance loss** (known as "control surface saturation").

- **Input Saturation (PWM Signal Limits)**

  The control input u_m sent to Electronic Speed Controllers (ESCs) typically ranges between standard PWM values [66]: 1000 $\mu s \leq u_m \leq$ 2000 μs or, equivalently, in normalized form: $0 \leq u_m \leq 1$

  ✓ **1000 µs** corresponds to **minimum throttle** (motor idle or cutoff),

  ✓ **2000 µs** corresponds to **maximum throttle**.

  Implication for Control:

  ✓ Commands outside this range are **clipped**.

  ✓ Controllers must be designed to operate well within actuator limits to prevent "clipping-induced" nonlinearities.

- **Rate Limits (Motor Acceleration Limits)** Motors cannot adjust their rotational speed instantaneously because of their rotational inertia and electrical dynamics [62]. The motor time constant τ_m constrains how rapidly the motor speed can follow the desired input. This results in rate constraints: $|\dot{\omega}\mathbf{m}| \leq \dot{\omega}$max, where $\dot{\omega}_{\max}$ is defined by the motor's torque capacity and system voltage [62].

- **Asymmetry and Dead Zones**

  ✓ Some motors may have slight manufacturing asymmetries: different motors respond slightly differently to the same control input [53].

  ✓ **Dead zones** exist where small PWM changes around the minimum input do not produce any motor movement [53].

  Implication for Control:

  ✓ Calibration procedures are required.

  ✓ Dead-zone compensators or adaptive control techniques may be necessary.

B. Consequences of Ignoring Actuator Constraints

If actuator constraints are ignored in controller design:

- **Saturation Instability**: When a control signal saturates, the effective closed-loop dynamics change — potentially leading to loss of stability [63].
- **Poor Tracking**: Desired accelerations or trajectories may not be achievable, leading to significant tracking errors [63].
- **Windup Effects**: In PID-based controllers, large integrator terms can accumulate if actuators saturate, causing severe overshoots when saturation ceases [63]. (This is known as **integrator windup**.)
- **Reduced Robustness**: Control systems may become highly sensitive to disturbances near actuator limits [63].

B. Methods to Handle Actuator Constraints

To safely handle actuator limitations, several techniques are employed:

- **Command Saturation**: Explicitly limit control signals before sending them to actuators.
- **Anti-Windup Mechanisms**: Modify integrator behavior in PID controllers when saturation occurs to prevent instability.
- **Model Predictive Control (MPC)**: MPC can explicitly incorporate actuator constraints during online optimization.
- **Gain Scheduling**: Adjust controller gains dynamically as the available actuator authority changes.
- **Saturation-Aware Design**: Design the baseline controller considering maximum available thrust/torque.

II.7.3 Integration into the Overall Model

In realistic quadcopter modeling, especially for precise or aggressive flight control, it is insufficient to assume that the motor thrusts and torques are immediately available upon command. Instead, motors and propellers introduce **additional dynamics** and limitations that must be incorporated into the global system model [62]. This section explains how actuator modeling modifies the full quadcopter dynamics and what implications this has for control system design. Properly accounting for these effects ensures that the control strategies remain feasible and effective under real-world conditions [62].

A. From Ideal to Realistic Control Inputs

- Idealized assumption (basic rigid body model):
  - ✓ Inputs u are assumed to be *immediate* and *direct* thrust and torque values.
- Realistic assumption (including actuator dynamics):
  - ✓ Inputs u_m are PWM signals (electrical control signals to motors).
  - ✓ These inputs pass through the **motor dynamics** (modeled as first-order systems) and then through **propeller models** (producing thrust and torque quadratically from motor speed).

In essence, the control input is no longer directly the physical force or torque but is mediated by actuator dynamics.

B. Extending the State-Space Model

- To account for this, the state vector must be expanded.
- Basic state vector (rigid body only) is mentioned on equation (II.36)
- Extended state vector (including actuators):

$$x_{extended} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r, \omega_1, \omega_2, \omega_3, \omega_4]^T \tag{II.47}$$

where:

✓ ω_i are the rotational speeds of the motors (rad/s),

✓ The additional states introduce **motor dynamics** into the system.

Important note:

This higher-dimensional system has a **direct impact** on controller and observer design, raising both the complexity of the system and the associated computational burden [67].

C. Modified Dynamics Equations

Each motor dynamic is described by a first-order linear differential equation:

$$\tau_{\mathbf{m}} \cdot \omega_{\mathbf{i}} + \omega_{\mathbf{i}} = \mathbf{k_m} \cdot \mathbf{u_{m,i}} \text{ for } \mathbf{i} = 1, 2, 3, 4 \tag{II.48}$$

✓ τ_m = motor time constant (accounts for mechanical and electrical inertia).

✓ k_m = motor gain (relates input command to steady-state motor speed).

Then, thrust and torque generated by each propeller are functions of motor speed:

$$\begin{aligned} \mathbf{T_i} &= \mathbf{k_T} \cdot \omega_\mathbf{i}^2 \\ \mathbf{Q_i} &= \mathbf{k_Q} \omega_\mathbf{i}^2 \end{aligned} \tag{II.49}$$

Thus, the **total forces** and **torques** applied to the quadcopter body are nonlinear functions of motor speeds, which themselves are *dynamically evolving* based on the inputs [67].
Impact on system matrices:

- The **A matrix** (system dynamics) becomes larger and includes motor dynamics.
- The **B matrix** (input influence) no longer directly maps the control inputs to forces and torques but instead to motor accelerations.

D. Control Design Considerations

Including actuator dynamics forces the control designer to **rethink** controller structure:

- **Lag Compensation**: Controllers must anticipate and correct for motor response lag, particularly in aggressive maneuvers [63].
- **Feedforward Terms**: Adding feedforward control based on predicted actuator behavior can improve performance [63].
- **Observer Design**: If motor speeds are not directly measured, a state observer (e.g., Kalman Filter) must estimate them, making the observer more complex [63].
- **Gain Scheduling**: Control gains may need adjustment depending on motor operating points (especially when motors saturate or behave nonlinearly at high speeds) [63].
- **Stability Margins**: Actuator dynamics introduce additional phase delay, reducing gain and phase margins, which must be considered in control stability analysis [63].
- **Selecting an Appropriate Control Strategy**: With many control techniques available, choosing the right one is crucial. Some methods offer greater freedom and robustness in managing quadcopter dynamics—particularly those grounded in a comprehensive mathematical model that accounts for all system equations. For example, the Backstepping method, in combination with Lyapunov-based stability analysis, systematically computes tracking errors and iteratively traces them back through the system's dynamics. Such model-based approaches allow for custom controller development tailored to the unique behavior of the quadcopter, especially under nonlinear and time-varying conditions [54].

II.8 Conclusion

This chapter laid a rigorous and structured foundation for modeling and analyzing the dynamics of a quadcopter drone, setting the stage for the development and evaluation of advanced control systems in the chapters to follow. It began by outlining the quadcopter's mechanical architecture and operational principles, highlighting the challenges that arise from its nonlinear, under-actuated, and tightly coupled dynamics.

The quadcopter was modeled as a six-degree-of-freedom (6DOF) rigid body, and its kinematic and dynamic equations were carefully derived using both Newton-Euler and Euler-Lagrange methods. These mathematical formulations capture the essential physical influences on the system, including translational and rotational dynamics, gravitational forces, rotor-induced thrust, and aerodynamic effects such as drag and rotor interactions.

A detailed discussion of coordinate transformations between inertial and body-fixed frames was also provided to accurately represent the drone's position and orientation in three-dimensional space. By presenting both the full nonlinear state-space model and its linearized version around the hover condition, the analysis offers two valuable perspectives: one that fully reflects the complexity of the system and another that simplifies the design of practical controllers while retaining key behaviors.

Furthermore, actuator dynamics were examined with attention to real-world constraints such as motor lag, thrust generation delays, saturation limits, and rate constraints. These practical factors are often neglected in idealized models but are vital for achieving reliable performance in actual flight conditions. Addressing these limitations highlights the importance of adopting robust, adaptive, and often nonlinear control approaches.

In summary, the models and insights developed in this chapter provide a solid theoretical backbone for the control strategies that will follow. They not only deepen our understanding of quadcopter behavior but also prepare us to tackle the challenges of stable and responsive autonomous flight. Building on this foundation, the next chapter will introduce advanced nonlinear control techniques, with a particular focus on the Backstepping method. This approach, grounded in Lyapunov-based stability analysis, allows complex control tasks to be broken down into smaller, manageable steps. Its structured use of the mathematical model makes it especially well-suited for handling the quadcopter's nonlinearities, ultimately leading to more intuitive and effective control design.

CHAPTER III

ADVANCED OPTIMIZATION STRATEGIES FOR QUADCOPTER TRAJECTORY TRACKING UNDER WIND DISTURBANCES

III.1 Introduction

The rapid advancement of the UAV technology has driven its widespread adoption across a wide range of applications, including surveillance, delivery, agriculture, and search-and-rescue missions [68][69]. A critical factor in UAV performance is their ability to maintain precise trajectory tracking under dynamic environmental conditions, especially when confronted with wind disturbances [70]. This chapter provides a comprehensive analysis of advanced control strategies aimed at improving drone trajectory tracking performance under challenging windy conditions. The focus is specifically on quadcopter systems with an X4 configuration (four rotors arranged in an X formation), which have become the most common platform for both research and commercial applications due to their mechanical simplicity and excellent maneuverability.

Traditional control methods, such as Proportional-Integral-Derivative (PID) controllers, have revealed limitations in handling the nonlinear dynamics and external disturbances inherent to UAV operations [71][72]. To address these challenges, researchers have developed more sophisticated control strategies [73], including backstepping control, which offers systematic design procedures along with Lyapunov-based stability guarantees [74]. However, even backstepping controllers can encounter difficulties when dealing with uncertainties, model inaccuracies, and time-varying parameters that frequently arise in real-world scenarios [74][75].

This chapter explores the integration of optimization algorithms with backstepping control to further enhance performance. Specifically, we investigate four approaches:

- Standard Backstepping Control (used as the baseline).
- Grey Wolf Optimizer (GWO)-Enhanced Backstepping Control [76].
- Garra-Rufa Optimization (GRO)-Enhanced Backstepping Control
- Pelican Optimization Algorithm (POA)-Enhanced Backstepping Control [77].

Through detailed mathematical modeling, simulation results, and comparative analysis, we demonstrate how these optimization-enhanced control strategies achieve significant improvements in trajectory tracking accuracy and robustness under wind disturbances.

III.2 Quadrotor Dynamics and Modeling

The accurate mathematical modeling of quadrotor dynamics is fundamental to designing effective control strategies, particularly under environmental disturbances such as wind [78][79]. This section provides a detailed derivation of the six degrees of freedom (6-DOF) quadrotor model, including both translational and rotational dynamics, and presents the state-space representation that will serve as the foundation for the control design.

III.2.1 Mathematical Model of 6-DOF

A quadrotor represents an underactuated system characterized by four independent control inputs, namely the thrust forces produced by its rotors, while possessing six dynamic degrees of freedom: three translational (x, y, z) and three rotational, specifically roll (ϕ), pitch (θ), and yaw (ψ) [80].
For this study, we adopt the commonly used X-configuration, where rotors M1 and M3 rotate clockwise, while M2 and M4 rotate counterclockwise, creating counteracting torques for balanced control [81].
The system is analyzed in two coordinate frames:

- **Earth-Fixed (Inertial) Frame (EF):** A global reference frame denoted by (X, Y, Z), typically aligned with the North-East-Down (NED) convention [82].
- **Body-Fixed Frame (BF):** A local frame attached to the quadrotor's center of gravity, denoted by (x_b, y_b, z_b) where z_b points downward through the fuselage [82].

The orientation of the quadrotor relative to the earth frame is described using Euler angles:

- Roll (ϕ): rotation about the body-fixed x_b-axis
- Pitch (θ): rotation about the body-fixed y_b-axis
- Yaw (ψ): rotation about the body-fixed z_b-axis

To transform between the two frames, the rotation matrix $R(\phi, \theta, \psi)$ is used [80]:

$$\mathbf{R} = \begin{bmatrix} \mathbf{Cos}\psi\mathbf{Cos}\theta & \mathbf{Cos}\psi\mathbf{Sin}\phi\mathbf{Sin}\theta - \mathbf{Cos}\phi\mathbf{Sin}\psi & \mathbf{Sin}\phi\mathbf{Sin}\psi + \mathbf{Cos}\phi\mathbf{Cos}\psi\mathbf{Sin}\theta \\ \mathbf{Cos}\theta\mathbf{Sin}\psi & \mathbf{Sin}\phi\mathbf{Sin}\psi\mathbf{Sin}\theta + \mathbf{Cos}\phi\mathbf{Cos}\psi & \mathbf{Cos}\phi\mathbf{Sin}\psi\mathbf{Sin}\theta - \mathbf{Cos}\psi\mathbf{Sin}\phi \\ -\mathbf{Sin}\theta & \mathbf{Cos}\theta\mathbf{Sin}\phi & \mathbf{Cos}\phi\mathbf{Cos}\theta \end{bmatrix} \quad \text{(III.1)}$$

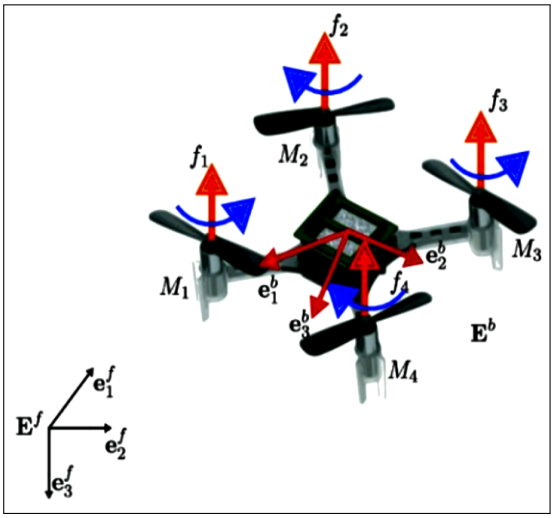


Figure III.1: Euler-Newton approach on Quadcopter [83].

The full nonlinear dynamics are derived using the Euler-Newton approach (Figure III.1), resulting in the following set of coupled equations [80][84]:

$$\begin{cases} \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{y}} \\ \dot{\mathbf{z}} \end{bmatrix} = [\mathbf{Rotxyz}] \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix} \\ \begin{bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{w}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{r} & -\mathbf{q} \\ -\mathbf{r} & \mathbf{0} & \mathbf{p} \\ \mathbf{q} & -\mathbf{p} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix} + g \begin{bmatrix} -\mathbf{sin}\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{bmatrix} + \frac{1}{\mathbf{m}} \begin{bmatrix} \mathbf{f_{wx}} \\ \mathbf{f_{wy}} \\ \mathbf{f_{wz}} - \mathbf{f_t} \end{bmatrix} \\ \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \sin\phi\tan\theta & \cos\phi\tan\theta \\ \mathbf{0} & \cos\phi & -\sin\phi \\ \mathbf{0} & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{r} \end{bmatrix} \\ \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{I_y}-\mathbf{I_z}}{\mathbf{I_x}} \\ \frac{\mathbf{I_z}-\mathbf{I_x}}{\mathbf{I_y}} \\ \frac{\mathbf{I_x}-\mathbf{I_y}}{\mathbf{I_z}} \end{bmatrix} \begin{bmatrix} \mathbf{rq} \\ \mathbf{pr} \\ \mathbf{pq} \end{bmatrix} + \begin{bmatrix} \frac{1}{\mathbf{I_x}}\mathbf{u_2} \\ \frac{1}{\mathbf{I_y}}\mathbf{u_3} \\ \frac{1}{\mathbf{I_z}}\mathbf{u_4} \end{bmatrix} \end{cases} \tag{III.2}$$

III.2.2 Mathematical Dynamic Representation

For systematic control design, especially when applying backstepping methods, it is advantageous to express the dynamics of the quadrotor in state-space form. We define the 12-dimensional state vector [85][86]:

$$\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_{12}}]^{\mathbf{T}} = \left[\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}, z, \dot{z}, y, \dot{y}, x, \dot{x}\right]^{\mathbf{T}} \tag{III.3}$$

The corresponding state-space equations are the following [80][87]:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1 x_4 x_6 + b_1 u_2 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = a_2 x_2 x_6 + b_2 u_3 \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = a_3 x_2 x_4 + b_3 u_4 \\ \dot{x}_7 = x_8 \\ \dot{x}_8 = \frac{\cos(x_1)\cos(x_2)}{m} U_1 - g \\ \dot{x}_9 = x_{10} \\ \dot{x}_{10} = \frac{U_y}{m} U_1 \\ \dot{x}_{11} = x_{12} \\ \dot{x}_{12} = \frac{U_x}{m} U_1 \end{cases} \tag{III.4}$$

where:

- ✓ $a_1 = \frac{I_y - I_z}{I_x}, a_2 = \frac{I_z - I_x}{I_y}, a_3 = \frac{I_x - I_y}{I_z}$
- ✓ $b_1 = \frac{d}{I_x}, b_2 = \frac{d}{I_y}, b_3 = \frac{d}{I_z}$ (With d representing the control moment arm)
- ✓ $U_x = \cos(x_1)\cos(x_3)\cos(x_5) + \sin(x_1)\sin(x_5)$
- ✓ $U_y = \cos(x_1)\sin(x_3)\sin(x_5) - \sin(x_1)\cos(x_5)$

This formulation explicitly separates the rotational and translational subsystems, providing a clear structure for the subsequent design of hierarchical control laws.

III.3 Backstepping Control Design

Backstepping control is a powerful nonlinear control technique that has gained prominence in UAV applications due to its ability to handle complex, cascaded dynamics inherent in quadcopters [88]. Unlike linear control methods (e.g., PID), which often rely on local linearization, backstepping preserves the nonlinear structure of the system, enabling global stability guarantees [89]. This section introduces the methodology and its application to quadcopter control, with a focus on:

- Theoretical foundations (principles, stability guarantees)
- Step-by-step design for attitude/position subsystems
- Practical considerations for implementation

We chose the backstepping because the Quadcopters are underactuated (6 DOF controlled by 4 rotors) and highly nonlinear due to:

- Coupling between rotational and translational dynamics
- Actuator saturation (limited rotor thrust)
- External disturbances (e.g., wind gusts)

Backstepping addresses these challenges by:

- Decomposing the system into manageable subsystems (attitude → position)
- Recursively stabilizing each subsystem using Lyapunov functions
- Preserving nonlinearities instead of approximating them

III.3.1 Principles of Backstepping Control

The backstepping control method offers a structured approach for managing nonlinear systems, particularly those with high-order dynamics such as a quadcopter. At its core, backstepping divides the overall system into a sequence of smaller, interconnected virtual subsystems. For each of these, intermediate control laws, known as virtual inputs, are designed to guide the system's behavior. By systematically organizing and stabilizing these subsystems, the designer can ensure that stability is maintained across the entire control framework, resulting in reliable system performance [90].

- **Virtual Control Variables:** These variables serve as intermediate control signals or reference states introduced within the control architecture. For example, a roll angle error is converted into a desired roll rate, which then acts as the reference for calculating the necessary torque input. This layered approach simplifies the overall control task by breaking down complex relationships into a sequence of manageable steps [89].

- **Lyapunov Stability:** A Lyapunov function V is constructed for each subsystem to formally demonstrate that the subsystem's error dynamics remain stable, which requires satisfying the condition $\dot{V} \leq 0$. By ensuring the stability of each inner control loop, the design progressively builds stability outward toward the higher-level loops, following a principle sometimes called back-propagated stability [90].

- **Recursive Design:** The design proceeds step by step, starting from the innermost dynamic loop [90]:

  ✓ Step 1: Stabilize the most immediate subsystem, such as the roll dynamics.

  ✓ Step 2: Use the virtual control law derived from Step 1 as a reference input for the next subsystem, for example, the roll rate dynamics.

  ✓ Continue this recursive process until the final actual control input, such as the rotor torques or thrust, is computed.

Mathematical Formulation

Consider a general n-order nonlinear system described by [90]:

$$\begin{cases} \dot{\mathbf{x}}_\mathbf{1} = \mathbf{f_1}\left(\mathbf{x_1}\right) + \mathbf{g_1}\left(\mathbf{x_1}\right)\mathbf{x_2} \\ \dot{\mathbf{x}}_\mathbf{2} = \mathbf{f_2}\left(\mathbf{x_1}, \mathbf{x_2}\right) + \mathbf{g_2}\left(\mathbf{x_1}, \mathbf{x_2}\right)\mathbf{u} \end{cases} \tag{III.5}$$

The backstepping procedure follows these key steps:

1. Treat $\mathbf{x_2}$ as a virtual input for the $\dot{\mathbf{x}}_\mathbf{1}$ subsystem.
2. Design $\mathbf{x_2} = \alpha(\mathbf{x_1})$ to stabilize x_1, for instance by selecting $\alpha\left(\mathbf{x_1}\right) = -\mathbf{k_1}\mathbf{x_1}$, where k_1 is a positive design gain.
3. Define the error $z = x_2 - \alpha(x_1)$, and derive a control law that stabilizes $\dot{z}$.
4. Compute the actual control input u such that the combined Lyapunov derivative satisfies $\dot{V} = \dot{V}_1 + \dot{V}_2 \leq 0$, guaranteeing global system stability.

Transition to Quadcopter Application

When applying these principles to the quadcopter's six degrees of freedom (6-DOF) dynamics, as outlined in Section 3.2, the control design is arranged in a hierarchical manner.

- **Inner Loop (Attitude Control):** The first stage is dedicated to stabilizing the roll, pitch, and yaw angles. This is achieved by designing virtual control inputs, such as desired angular rates, that effectively manage the quadcopter's rotational behavior [91].

- **Outer Loop (Position Control):** Once the attitude is stabilized, the outer loop takes over to regulate the translational motion. This ensures that the quadcopter accurately follows the desired $(x,\ y,\ z)$ trajectories in space [91].

This structured control architecture proves especially valuable because the inner loop can quickly counteract disturbances [91], including wind effects or modeling inaccuracies, before these disturbances affect the outer position loop. As a result, the entire system maintains robust and reliable performance, even under environmental uncertainties or limitations in the actuators.

III.3.2 Backstepping Control Formulation for Quadcopters

To apply backstepping control effectively to a quadcopter, we first decompose its complex dynamics into two major control layers [92]:

- **Attitude Control**, which governs Roll, Pitch, and Yaw.
- **Position Control**, which manages the translational motion along the X, Y, and Z axes.

Each layer is stabilized using a recursive approach, ensuring that tracking errors at each stage progressively shrink to zero [92]. This layered strategy not only simplifies the control design (Figure III.2) but also strengthens the system's resilience to disturbances and model uncertainties [92].

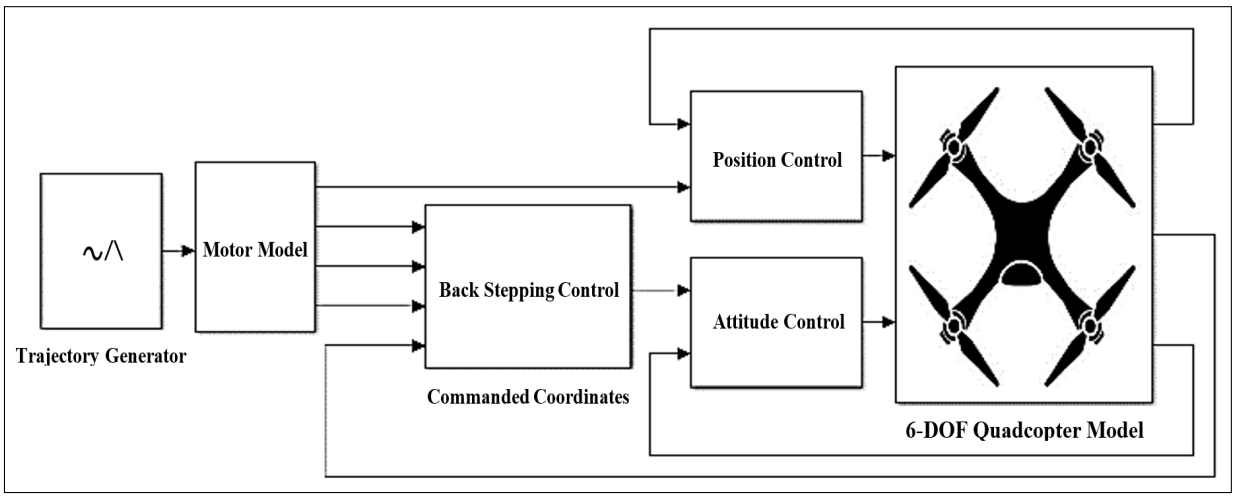


Figure III.2: 6-DOF Quadcopter Schema, including the backstepping controller.

Step 1: Attitude Control (Focusing on Roll Angle Example)

Let's start with the roll dynamics, which serve as a representative case for the general method. The roll behavior can be modeled by the following system of equations [92]:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1 x_4 x_6 + b_1 u_2 \end{cases} \tag{III.6}$$

where:

- ✓ $x_1 = \phi$ represents the roll angle.
- ✓ $x_2 = \dot{\phi}$ represents the roll rate.
- ✓ u_2 is the control input acting on Roll.

The terms a_1 and b_1 are constants derived from the system's physical parameters, while x_4 and x_6 correspond to other states such as pitch and yaw rates, reflecting the coupling between the different rotational axes [92].

Design Steps:

1. **Define Tracking Error:** We first quantify how far the system is from the desired roll behavior:
$$\varepsilon_1 = x_{1d} - x_1 \tag{III.7}$$
where x_{1d} represents the target roll angle. This error becomes the starting point for designing a stabilizing controller.

2. **First Lyapunov Function:** To formally ensure stability, we define a Lyapunov function:
$$V_1 = \frac{1}{2}\varepsilon_1^2 \tag{III.8}$$
This function measures the "energy" or deviation in the system due to the roll angle error. Taking its derivative gives:
$$\dot{V}_1 = \varepsilon_1\dot{\varepsilon}_1 = \varepsilon_1(\dot{x}_{1d} - x_2) \tag{III.9}$$
To stabilize this subsystem, we need to design a control law that keeps $\dot{V}_1$ non-positive, meaning the error does not grow over time.

3. **Virtual Control Law for x_2:** We introduce a virtual control input to guide the roll rate:
$$x_{2d} = \dot{x}_{1d} + k_1\varepsilon_1 \tag{III.10}$$
where $k_1 > 0$ is a gain chosen to correct the roll angle error. This virtual input effectively tells the system how the roll rate should behave to bring the roll angle error back to zero.

4. **Define New Error for Roll Rate:** Next, we define the difference between the actual roll rate and the desired (virtual) roll rate:
$$\varepsilon_2 = x_{2d} - x_2 \tag{III.11}$$
This second error allows us to construct a more refined control law that handles both the angle and rate dynamics.

5. **Augmented Lyapunov Function:** We extend our Lyapunov function to capture both tracking errors:
$$V_2 = V_1 + \frac{1}{2}\varepsilon_2^2 \tag{III.12}$$

The derivative becomes [90]:

$$\begin{aligned}\dot{V}_2 = \dot{V}_1 + \varepsilon_2\dot{\varepsilon}_2 = \varepsilon_1\dot{\varepsilon}_1 + \varepsilon_2\dot{\varepsilon}_2 = \varepsilon_1\left(\dot{x}_{1d} - x_2\right) + \varepsilon_2\left(\dot{x}_{2d} - \dot{x}_2\right)\\ = \varepsilon_1\left(\dot{x}_{1d} - (\dot{x}_2 - \varepsilon_2)\right) + \varepsilon_2(\dot{x}_{2d} - (a_1x_4x_6 + b_1u_2))\\ = \varepsilon_1\left(\dot{x}_1 - \dot{x}_{2d}\right) + \varepsilon_1\varepsilon_2 + \varepsilon_2(\dot{x}_{2d} - a_1x_4x_6 - b_1u_2)\\ = \varepsilon_1(-k_1\varepsilon_1) + \varepsilon_2\left(\varepsilon_1 + (\dot{x}_{2d} - a_1x_4x_6 - b_1u_2)\right)\end{aligned}$$

So:

$$\dot{V}_2 = -k_1\varepsilon_1^2 + \varepsilon_2(\varepsilon_1 + \dot{x}_{2d} - a_1x_4x_6 - b_1u_2) \tag{III.13}$$

Here, we see that part of the system's dynamics is now explicitly linked to the actual control input u_2.

6. **Derive the Final Control Law for Roll:** To make sure the full system stabilizes, we solve for u_2 in a way that ensures $\dot{V}_2 \leq 0$:

$$\mathbf{u_2} = \frac{1}{b1}\left[\varepsilon_1 - k_1x_2 - a_1x_4x_6 + k_2\varepsilon_2\right] \tag{III.14}$$

with $k_2 > 0$ as an additional gain tuning the response to the roll rate error. This final expression gives us the actual torque command that should be applied to the roll axis to correct both angle and rate deviations [90].

By following this stepwise backstepping procedure, we ensure that the errors in both roll angle and roll rate converge to zero:

$$\dot{V}_2 = -k_1\varepsilon_1^2 - k_2\varepsilon_2 \leq 0 \tag{III.15}$$

This guarantees that the system's energy decreases over time, leading to stable behavior. The same recursive design is then applied to the pitch and yaw subsystems, as well as to the outer position control loops (x, y, z) [90].
The strength of backstepping lies in its ability to handle the nonlinear and coupled nature of the quadcopter's dynamics by layering virtual and real controls systematically. This layered stabilization makes the overall system more robust, even in the presence of disturbances or modeling uncertainties [90].

Step 2: Pitch and Yaw Control

Building on the earlier backstepping approach, we extend the control design to cover the remaining rotational angles (pitch θ and yaw ψ) as well as the translational positions along the X, Y, and Z axes [90]. By using the system equations provided in equation (3) and applying the same recursive stabilization principles, we derive the following control laws for each subsystem. These expressions ensure that tracking errors across all rotational and positional channels converge properly [90].

- **Pitch Control (u_3):**

$$\mathbf{u_3} = \frac{1}{b_2}\left[\varepsilon_3 - k_3 x_4 - a_2 x_2 x_6 + k_4 \varepsilon_4\right] \tag{III.16}$$

- **Yaw Control (u_4):**

$$\mathbf{u_4} = \frac{1}{b_3}\left[\varepsilon_5 - k_5 x_6 - a_3 x_2 x_4 + k_6 \varepsilon_6\right] \tag{III.17}$$

Step 3: Position Control (Altitude "Z" and "X","Y" Motion)

At this stage of the backstepping design, we shift our attention from stabilizing the quadcopter's attitude (its rotational orientation) to managing its position in three-dimensional space. This involves two main aspects: Altitude control Z and horizontal motion control along the X and Y directions [90].

- **Altitude Control (u_1):**

$$\mathbf{u_1} = \frac{m}{\cos(x_1)\cos(x_2)}\left[-\varepsilon_7 + g - k_7 x_8 - k_8 \varepsilon_8\right] \tag{III.18}$$

- **X-Position Control (U_x):**

$$U_x = \frac{m}{u_1}\left[-\varepsilon_{11} - k_{11} x_{12} - k_{12} \varepsilon_{12}\right] \tag{III.19}$$

- **Y-Position Control (U_y):**

$$U_y = \frac{m}{u_1}\left[-\varepsilon_9 - k_9 x_{10} - k_{10} \varepsilon_{10}\right] \tag{III.20}$$

So, in the result we obtain the complete set of backstepping control laws:

$$\begin{cases} Roll\ control\ Angle\ \phi : \mathbf{u_2} = \frac{1}{b1}\left[\varepsilon_1 - k_1 x_2 - a_1 x_4 x_6 + k_2 \varepsilon_2\right] \\ Pitch\ control\ Angle\ \theta : \mathbf{u_3} = \frac{1}{b2}\left[\varepsilon_3 - k_3 x_4 - a_2 x_2 x_6 + k_4 \varepsilon_4\right] \\ Yaw\ control\ Angle\ \psi : \mathbf{u_4} = \frac{1}{b3}\left[\varepsilon_5 - k_5 x_6 - a_3 x_2 x_4 + k_6 \varepsilon_6\right] \\ Altitude\ "Z"\ Control : \mathbf{u_1} = \frac{m}{\cos(x_1)\cos(x_2)}\left[-\varepsilon_7 + g - k_7 x_8 - k_8 \varepsilon_8\right] \\ X - Position\ Control : \mathbf{U_x} = \frac{m}{U_1}\left[-\varepsilon_{11} - k_{11} x_{12} - k_{12} \varepsilon_{12}\right] \\ Y - Position\ Control : \mathbf{U_y} = \frac{m}{U_1}\left[-\varepsilon_9 - k_9 x_{10} - k_{10} \varepsilon_{10}\right] \end{cases} \tag{III.21}$$

The control gains K_1–K_{12} need to be carefully selected to ensure system stability and satisfactory performance. This is where optimization algorithms can significantly enhance the controller's performance.

III.4 Grey Wolf Optimizer (GWO)-Enhanced Backstepping Control

Combining backstepping control with the **Grey Wolf Optimizer (GWO)** offers a practical and adaptive solution for improving the performance of nonlinear systems such as quadcopters [80]. While backstepping provides a structured method for stabilizing subsystems, its reliance on precise gain tuning can limit effectiveness. GWO addresses this by automatically optimizing control gains, leading to improved tracking performance, stronger disturbance rejection, and greater robustness under dynamic conditions [80].

III.4.1 Grey Wolf Optimization Algorithm

The Grey Wolf Optimizer (GWO) is a well-known metaheuristic algorithm that draws inspiration from the social organization and hunting behavior of grey wolves in nature (Figure III.3) [93]. This optimization technique has gained significant attention due to its simplicity, flexibility, and ability to handle complex, nonlinear search spaces effectively [94]. In the GWO algorithm, the population of candidate solutions is structured to reflect the hierarchical nature of a wolf pack. Specifically, individuals in the population are ranked into four distinct roles [93]:

- **Alpha (α):** Represents the best solution found so far and leads the decision-making process.
- **Beta (β):** The second-best candidate, supporting the alpha and providing additional guidance.
- **Delta (δ):** The third-ranked solution, assisting both alpha and beta in influencing the pack.
- **Omega (ω):** The remaining individuals, which follow the lead of the top three and explore the search space.

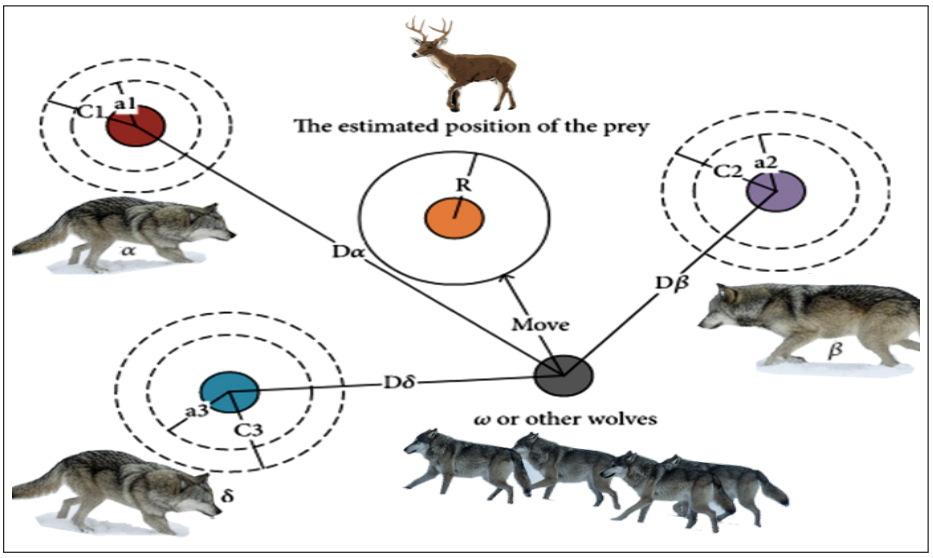


Figure III.3: Grey Wolf Behaviour to hunting a prey

The optimization process mimics the wolves' hunting mechanism, which unfolds in three key stages:

1. **Encircling the prey:**

   In this phase, the wolves estimate the location of the prey (representing the optimal solution) and adjust their positions accordingly. The mathematical model captures this dynamic through the following relations [93]:

$$\overrightarrow{D} = \left|\overrightarrow{C}\overrightarrow{X_p}(t) - \overrightarrow{X}(t)\right| \tag{III.22}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X_p}(t) - \overrightarrow{A}.\overrightarrow{D} \tag{III.23}$$

   Here, $\overrightarrow{A}$ and $\overrightarrow{C}$ are adaptive coefficient vectors, defined by [92]:

$$\overrightarrow{A} = 2.\overrightarrow{a}.\overrightarrow{r_1} - \overrightarrow{a} \tag{III.24}$$

$$\overrightarrow{C} = 2.\overrightarrow{r_2} \tag{III.25}$$

   where $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vectors in the range [0,1], and $\overrightarrow{a}$ decreases linearly over time, shifting the balance from exploration to exploitation [93].

2. **Hunting:**

   Once the prey's estimated location is roughly identified, the three top-ranking wolves (alpha, beta, and delta) take the lead in guiding the hunting process [93]. The remaining wolves adjust their positions by following the influence and directional cues provided by these leaders [93]:

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}.\left|\overrightarrow{C_1}.\overrightarrow{X_\alpha} - \overrightarrow{X}\right| \tag{III.26}$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.\left|\overrightarrow{C_2}.\overrightarrow{X_\beta} - \overrightarrow{X}\right| \tag{III.27}$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.\left|\overrightarrow{C_3}.\overrightarrow{X_\delta} - \overrightarrow{X}\right| \tag{III.28}$$

   The average of these guided positions forms the updated candidate [92]:

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{III.29}$$

3. **Searching for prey (exploration phase):**

   To prevent the search process from getting stuck in local optima, the wolves must strike a careful balance between following the lead wolves and independently exploring new areas. When the magnitude of $\mathbf{A}$ is greater than 1 ($\|\mathbf{A}\| > 1$), the wolves are encouraged to move away from the prey or from each other, effectively broadening their search [93]. This exploration phase increases diversity within the population and allows the algorithm to scan the wider solution space for better answers [94].

4. **Attacking the prey (exploitation):**

   As the wolves begin to close in on the prey, the level of exploration gradually decreases, and the wolves shift their focus toward fine-tuning their positions [94]. This occurs when the magnitude of $\mathbf{A}$ drops below 1 ($\|\mathbf{A}\| < 1$), signaling a transition to exploitation [93]. During this phase, the search concentrates on refining the best-known solutions and ensuring convergence toward the global optimum [94].

As a summary of the Grey Wolf Optimizer (GWO) algorithm, we present its key steps in the form of the pseudo-code provided below [80]:

Initialize the grey wolf population $\overrightarrow{X_i}, \quad i = 1, 2, \ldots, n$
Initialize parameters α, $\mathbf{A}$, and $\mathbf{C}$
Calculate the fitness of each search agent
$X_\alpha \leftarrow$ best search agent
$X_\beta \leftarrow$ second best search agent
$X_\delta \leftarrow$ third best search agent
while $t <$ Max number of iterations **do**
  for each search agent **do**
    Update the position of the current search agent using X_α, X_β, and X_δ
  end for
  Update parameters $\mathbf{A}$ and $\mathbf{C}$
  Recalculate fitness of all agents
  Update X_α, X_β, and X_δ
  $t \leftarrow t + 1$
end while
return X_α

III.4.2 GWO for Backstepping Gain Optimization

The Grey Wolf Optimizer (GWO) is applied to fine-tune the backstepping control gains k_1 through k_{12}, with the goal of achieving the best possible controller performance [95]. This optimization process unfolds through a series of well-structured steps that combine systematic search with adaptive learning as follow:

Initialization

At the outset, the search space for each gain is carefully defined, specifying the upper and lower bounds within which the algorithm can explore [95]. A population of candidate solutions, representing different combinations of gain values, is then initialized randomly across this search space. The number of iterations and the size of the population are set as key parameters to control the optimization run [95].

Fitness Function Design

The fitness of each candidate gain set is evaluated using a performance-driven objective function [96]. This function measures how effectively the controller balances multiple goals, such as minimizing the tracking error, reducing control effort, and achieving fast settling times. The fitness function is typically expressed as [96]:

$$J = w_1 \int e^2(t)dt + w_2 \int u^2(t)dt + w_3 t_s \tag{III.30}$$

where:

- ✓ $e(t)$ represents the tracking error, reflecting how closely the system follows the desired trajectory;
- ✓ $u(t)$ is the control effort, indicating how much input energy the controller uses;
- ✓ t_s denotes the settling time, measuring how quickly the system reaches a steady state;
- ✓ w_1, w_2, w_3 are weighting factors that balance the importance of each term depending on the design priorities.

Optimization Process

Throughout the optimization process, the algorithm continuously identifies the top-performing solutions, known as the alpha, beta, and delta wolves, which guide the search direction for the rest of the population [80]. By iteratively updating each wolf's position in the search space, the population gradually converges toward a set of gains that optimally balance performance trade-offs [80].

Implementation

Once the optimization is complete, the identified gains are implemented within the backstepping controller, significantly enhancing its ability to handle complex dynamic conditions [94]. If desired, the system can also include an online adaptation mechanism, allowing the controller to adjust gains in real time when facing varying operating environments [80].

III.4.3 Advantages of GWO-Backstepping

The integration of the Grey Wolf Optimizer (GWO) with the backstepping control framework brings a range of important advantages that strengthen the system's performance and usability, offering several advantages:

1. **Improved tracking accuracy:** Achieved because the optimized gains systematically reduce the tracking errors between the system's actual response and the desired reference signals. This leads to more precise trajectory following and better overall system behavior [80].

2. **Enhanced robustness:** The approach provides enhanced robustness, giving the controller stronger disturbance rejection capabilities. Thanks to the optimization process, the controller is better equipped to handle uncertainties in the system model and external disturbances such as wind or payload changes [80].
3. **Automatic tuning:** Another key benefit is automatic tuning, which removes the need for time-consuming and often tedious manual gain selection. The GWO algorithm explores the solution space efficiently, identifying the best gain settings without relying on trial-and-error approaches from the designer [93].
4. **Adaptability:** In addition, the method offers adaptability, as it can be extended to include online adaptation mechanisms. This means the controller can adjust its parameters in real time, ensuring consistent performance even as the system's dynamics or environment evolve [93].
5. **Computational efficiency:** The approach is known for its computational efficiency. The GWO algorithm typically converges relatively quickly compared to other metaheuristic methods, making it suitable for practical engineering applications where computational resources or time are limited [96].

Together, these advantages position the GWO-enhanced backstepping controller as a powerful and versatile solution for complex control problems, offering both strong theoretical foundations and practical effectiveness.

III.5 Garra Rufa Optimization (GRO)-Enhanced Backstepping Control

III.5.1 Garra Rufa Optimization Algorithm

The Garra Rufa Optimization (GRO) algorithm draws inspiration from the feeding behavior of doctor fish (Figure III.4), small aquatic creatures known for their cooperative yet competitive dynamics [97]. This optimization method introduces several unique features that set it apart from other metaheuristic approaches [97]:

- **Population initialization:** The algorithm begins by distributing the fish (candidate solutions) randomly throughout the search space, ensuring a diverse set of starting points [98].
- **Movement strategy:** Each fish updates its position by combining influences from both the best-known solution and randomly selected neighbors. Mathematically, this is expressed as [98]:

$$\mathbf{x_i^{new}} = \mathbf{x_i} + \text{rand} \cdot (\mathbf{x_{best}} - \mathbf{x_i}) + \text{rand} \cdot (\mathbf{x_{rand}} - \mathbf{x_i}) \tag{III.31}$$

  where x_{best} represents the leader's position, x_{rand} is a randomly chosen fish, and rand is a random number between 0 and 1.

- **Feeding behavior:** The algorithm intensifies its local search in promising regions, mimicking the way doctor fish cluster around nutrient-rich areas [98].
- **Social hierarchy:** Leader fish play a guiding role, directing the movement and behavior of the rest of the population toward more optimal solutions [98].

Figure III.4: Garra-Rufa hunting Behaviour (feeding behavior of Doctor fiches)

III.5.2 GRO for Backstepping Gain Optimization

The GRO algorithm is employed to fine-tune the gains used in the backstepping controller, aiming to achieve an optimal balance between performance, robustness, and efficiency [92]. The optimization process follows these steps:

- **Parameter setup:** Define the search boundaries for each backstepping gain and set the population size along with the maximum number of iterations allowed [92].
- **Fitness function:** The quality of each candidate solution is evaluated using a multi-objective fitness function:

$$J = w_1 \cdot \text{RMSE} + w_2 \cdot \max|e(t)| + w_3 \cdot \text{energy consumption} \tag{III.32}$$

  where RMSE is the root mean square tracking error, $\max|e(t)|$ represents the maximum error magnitude, and the last term accounts for the total energy used by the control system. The weighting factors w_1, w_2, w_3 balance the importance of each criterion.
- **Optimization steps:**
  1. Initialize the fish population randomly within the defined search space [92].
  2. Evaluate the fitness of each fish using the defined performance criteria [92].
  3. Update each fish's position according to the GRO movement equations, incorporating both leader guidance and peer influence [92].
  4. Apply intensified local search through the feeding behavior mechanism to refine promising solutions [92].
  5. Repeat the process until the stopping conditions are satisfied.

III.5.3 Performance of GRO-Backstepping

The GRO-enhanced backstepping controller offers several notable performance advantages:

1. **Rapid convergence:** It identifies good-quality solutions quickly, improving the controller's overall efficiency.
2. **Effective disturbance rejection:** The controller can effectively manage external disturbances, such as sudden wind gusts, maintaining stable system behavior [92].
3. **Smooth control inputs:** By generating smoother control signals, the controller helps reduce mechanical stress on actuators, extending their operational life [92].
4. **Robust performance:** It consistently maintains system stability even when operating conditions vary, demonstrating strong resilience and adaptability [92].

III.6 Pelican Optimization Algorithm (POA)-Enhanced Backstepping Control

Before diving into the details of the Pelican Optimization Algorithm, it is important to understand why this algorithm is paired with the backstepping control approach. The integration aims to enhance the controller's performance by optimizing its parameters through a nature-inspired search strategy. Similar to how pelicans skillfully locate and capture prey in dynamic environments, the POA offers a balanced mechanism for navigating complex solution landscapes [90].

III.6.1 Pelican Optimization Algorithm

The Pelican Optimization Algorithm (POA) draws inspiration from the distinctive hunting strategies of pelicans [99]. Designed to balance exploration and exploitation, the POA is divided into two main phases that mirror the pelicans' natural search and capture behavior [99].

A. Exploration Phase (Moving Towards Prey)

In this phase, pelicans search for the prey's location and adjust their positions to move closer. Mathematically, this is expressed as [99]:

$$x_{i,j}^{new1} = x_{i,j} + \text{rand} \cdot (p_j - I \cdot x_{i,j}) \tag{III.33}$$

Where:

- ✓ p_j represents the prey's position in dimension j
- ✓ I is a randomly selected integer (either 1 or 2)
- ✓ rand is a random number between 0 and 1

This formulation helps the pelicans (candidate solutions) explore the search space broadly, preventing premature convergence and promoting the discovery of promising areas.

B. Exploitation Phase (Winging on the Water Surface)

Once the pelicans are close to the prey, they switch to fine-tuned movements that resemble winging on the water surface to scoop up fish [99]. This phase is described by:

$$x_{i,j}^{new1} = x_{i,j} + R \cdot \left(1 - \frac{t}{T}\right) \cdot (2 \cdot \text{rand} - 1) \cdot x_{i,j} \tag{III.34}$$

Where:

- R is a constant (commonly set to 0.2)
- t is the current iteration
- T is the maximum number of iterations

Here, the position updates become more localized and refined, focusing on exploiting the most promising solutions identified during the exploration phase [90]. If the new position results in an improved fitness value, it is accepted; otherwise, the previous is retained [90]. By combining these two phases, the POA offers a flexible and efficient search mechanism, making it well-suited for optimization tasks where both global exploration and local exploitation are crucial.

III.6.2 POA for Backstepping Gain Optimization

The Pelican Optimization Algorithm (POA) is applied to systematically fine-tune the backstepping control gains, ensuring optimal controller performance across various system conditions [90].

A. Initialization

The process begins by defining a population of pelicans, where each pelican represents a candidate solution with randomly assigned gain values. The algorithm's key parameters, such as the constant R and the maximum number of iterations, are also set during this stage [90].

B. Fitness Evaluation

For each pelican (i.e., gain configuration), the system is simulated, and its performance is assessed using a carefully designed multi-objective fitness function [100]:

$$J = \alpha_1 \cdot \text{ISE} + \alpha_2 \cdot \text{IAE} + \alpha_3 \cdot \text{ITAE} + \alpha_4 \cdot \text{control effort} \tag{III.35}$$

where:

- ISE is the integral of squared error [100].
- IAE is the integral of absolute error
- ITAE is the integral of time-weighted absolute error [100].
- The control effort term penalizes excessive actuation

The weighting factors $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ allow prioritization of the different objectives depending on the design goals.

C. Optimization Process

The POA optimization unfolds in two key phases:

- The exploration phase guides the pelicans to search broadly across the solution space
- The exploitation phase fine-tunes their positions by focusing on promising regions

After each iteration, the algorithm updates the best solutions based on fitness improvements. This cycle is repeated until the stopping criterion is met, typically defined by convergence or a maximum number of iterations, as the algorithm explains below [90]:

Algorithm: Pelican Optimization Algorithm (POA)

Initialize the pelican population X_i $(i = 1, 2, \ldots, n)$
Initialize parameters for exploration and convergence
Calculate the fitness of each pelican
Identify the top-performing pelicans as leaders
while $t <$ Max number of iterations **do**
    # Step 1: Exploration and Encircling
    for each pelican (agent) **do**
        if exploration phase **then**
            Move the pelican to explore the search space broadly
        else
            Adjust the pelican's position to converge towards the leader positions, simulating group coordination.
        end if
    end for
    # Step 2: Updating Leaders
    Calculate the fitness of each pelican
    Reevaluate and update the leader positions based on the best solutions found in this iteration
    Gradually shift parameters to transition from exploration to convergence
    $t = t + 1$
end while
return *the best solution found*

D. Implementation

Once the optimal gain set is identified, it is implemented directly into the backstepping controller. Additionally, the POA framework can be periodically reapplied in real time to adapt the gains dynamically, ensuring the controller remains effective even under time-varying system conditions.

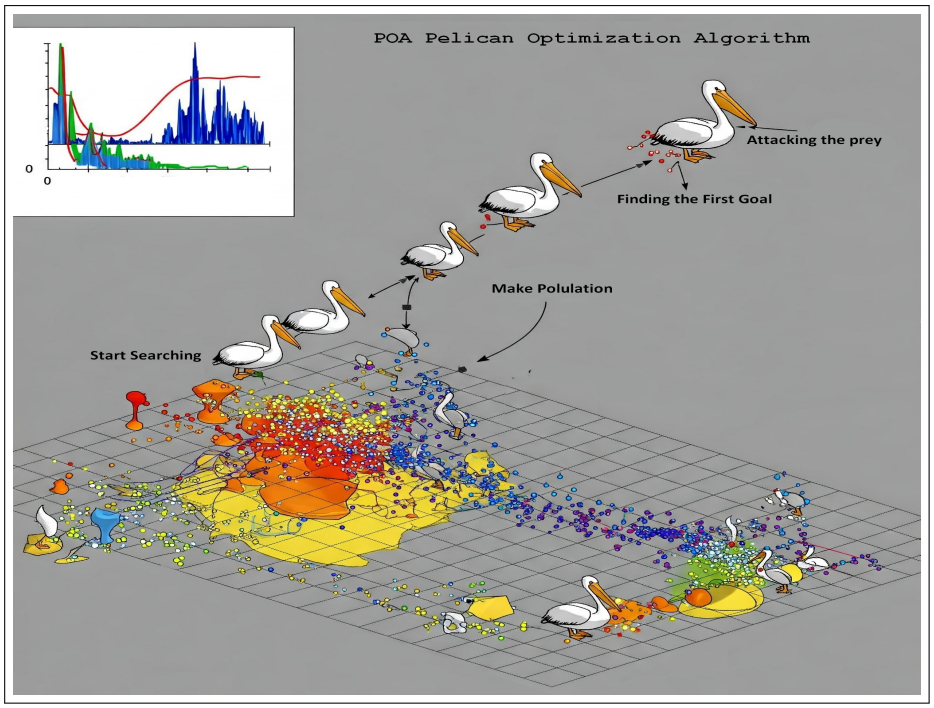


Figure III.5: Pelican Optimization Algorithm from Pelicans' Behaviour

III.6.3 Advantages of POA-Backstepping

The Pelican Optimization Algorithm (POA), when integrated into the backstepping control framework, offers a set of notable advantages over alternative metaheuristic methods such as the Grey Wolf Optimizer (GWO) [90]. By leveraging its distinctive two-phase strategy, the POA is able to navigate the optimization landscape with both breadth and precision. This results in performance improvements that are evident not only in the speed of convergence but also in the accuracy and robustness of the controller [90]. Below, we summarize the key advantages provided by the POA-enhanced backstepping approach:

- **Better exploration-exploitation balance:** The POA's two-phase mechanism ensures a more effective balance between global search and local refinement, improving the overall efficiency of the optimization process [90].
- **Faster convergence:** By efficiently navigating the parameter space, the POA reaches optimal solutions with fewer iterations compared to GWO-based designs [90].
- **Higher precision:** The optimized controller achieves notably lower tracking errors, enhancing the accuracy of system responses [90].
- **Improved disturbance rejection:** The POA-based controller demonstrates strong robustness against environmental uncertainties, particularly in rejecting wind disturbances [90].
- **Adaptability:** The method maintains reliable performance across a wide range of operating conditions, making it well-suited for systems requiring flexible and resilient control.

III.7 Simulation Results and Comparative Analysis

This section critically evaluates the effectiveness of the proposed methods for trajectory tracking in X-4 drone systems. The quadcopter's dynamics are modeled and simulated using MATLAB 2021a. To establish a solid baseline for comparison, a conventional backstepping controller is first implemented to track two distinct trajectory paths [101]. The performance of this basic backstepping controller is then compared against that of the Grey Wolf Optimizer (GWO)-, Pelican Optimization Algorithm (POA)-, and Garra-Rufa Optimization (GRO)-enhanced backstepping controllers. This comparative analysis aims to clearly highlight the impact of integrating these bio-inspired optimization methods such as POA on the overall performance of the control strategy [102]. In the second phase of the evaluation, the same trajectory paths, control methods, and optimization algorithms are used, but a wind disturbance is introduced at the 30-second mark. This experimental setup allows for a comprehensive assessment of the drone's dynamic response and the relative effectiveness of the standard backstepping controller and its enhanced versions in mitigating the effects of external environmental disturbances.

III.7.1 Simulation Setup

The designed controllers were thoroughly evaluated using simulations conducted in MATLAB/Simulink. To comprehensively evaluate controller performance, two scenarios were tested, one with disturbance and one without, each incorporating two challenging trajectory paths [103]:

- **A cylindrical path**, representing a three-dimensional circle motion
- **A helical path**, resembling a spring-shaped trajectory

These trajectories were chosen to test both the precision of the controllers and their ability to handle complex motion patterns [103]. To conduct this test, a simulation model was developed, as illustrated in the corresponding Figure III.6, implementing a backstepping controller integrated with an X-4 quadcopter system.

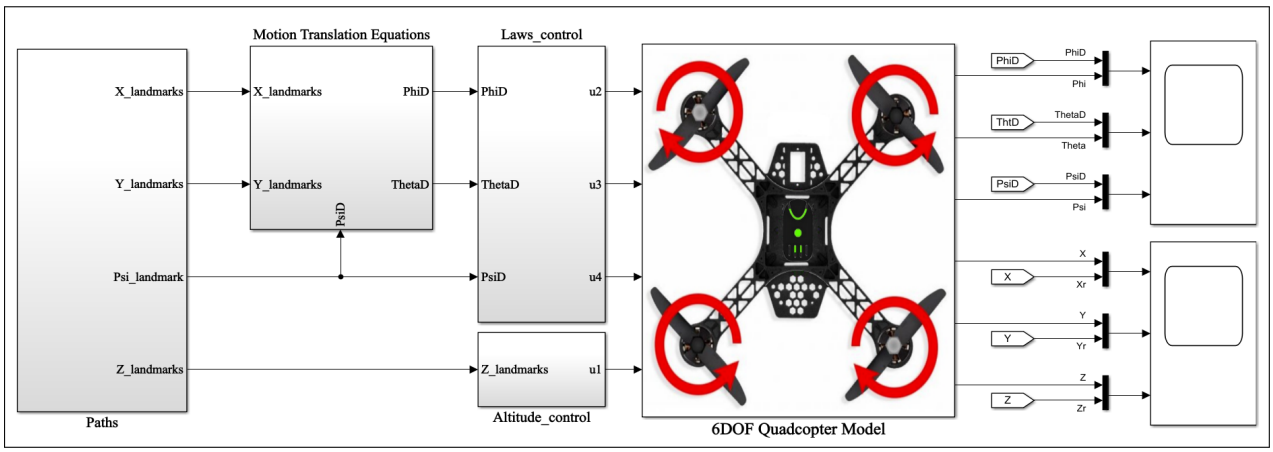


Figure III.6: Simulation Block Diagram of an X-4 Quadcopter Controlled by Backstepping)

The quadrotor system was modeled with the following parameters [80][90][92]:

Table III.1: Quadcopter parameters

Type	**Parameters**	**Units**
X, Y moment of inertia	I_{xx}, I_{yy}	2.2×10^{-2} kg.m^2
Z moment of inertia	I_{zz}	4.39×10^{-2} kg.m^2
Distance to center	l	2.25×10^{-1} m
Acceleration Gravity	g	9.81 m/s^2
Quad Mass	m	1.8 kg
Drag factor	d	1
Grey Wolf Optimization Parameters	Max of wolves	25
	Max Iterations	115
Garra Rufa Optimization parameters	Max of Fishes	100
	groups	2
	Iteration's	100
Pelican Optimization Algorithm Parameters	Max of Pelicans	50
	Iteration's	100
	Upper Bounds	50
	Lower Bounds	1
Sampling time 1st Path	T_s	0.1s
Sampling time 2nd Path	T_s	0.2s
Wind disturbance	W	5 m/s
	t	30s

III.7.2 Performance Metrics

The performance of each controller was evaluated using several key metrics:

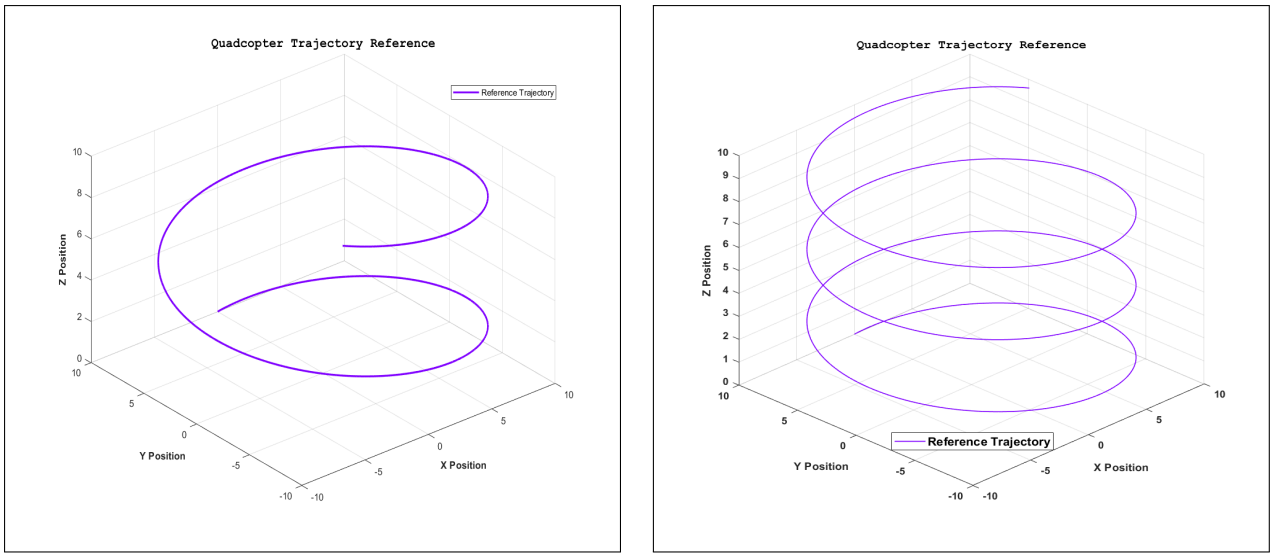
- **Mean Square Error (MSE):** Measures the average deviation between the desired and actual trajectories over time [104].
- **Maximum tracking error:** Captures the largest instantaneous deviation during the trajectory [104].
- **Settling time:** Quantifies the time taken for the system to stabilize within an acceptable error band after disturbances

III.7.3 Analysis of Backstepping Controller Performance with Algorithms

This study compares the performance of the standard backstepping controller with the GWO-, GRO- and POA-optimized backstepping controller across two trajectory paths. In both paths, the quadcopter moves clockwise on the X, Y and Z planes around the origin (0, 0, 0).

A- First Scenario: Trajectory Tracking Assessment

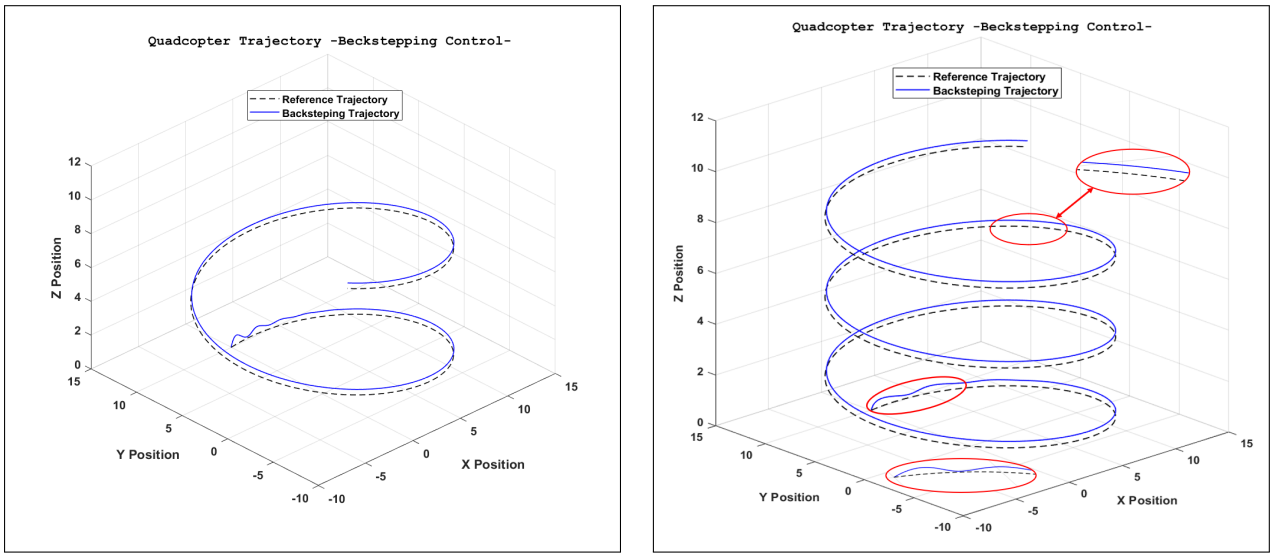
In the first scenario we will try to control the quadcopter tracking trajectory with backstepping controller and the three algorithms without any disturbance, so Figure III.7a & Figure III.7b present the two reference trajectories that the quadcopter, governed by the backstepping controller, is required to follow.



(a) First Trajectory Path (b) Second Trajectory Path

Figure III.7: Two challenging trace trajectory paths

In the first path which follows a cylindrical path across the X, Y, and Z axes, the standard backstepping controller was able to track the desired path, although it exhibited minor errors along the Z-axis and slight deviations on the X and Y axes during the initial few seconds. In the second path, which involves a helical path, similar patterns were observed as shown in Figure III.8 (a & b) for both paths. For additional details, Annex II provides the full position and orientation trajectories.



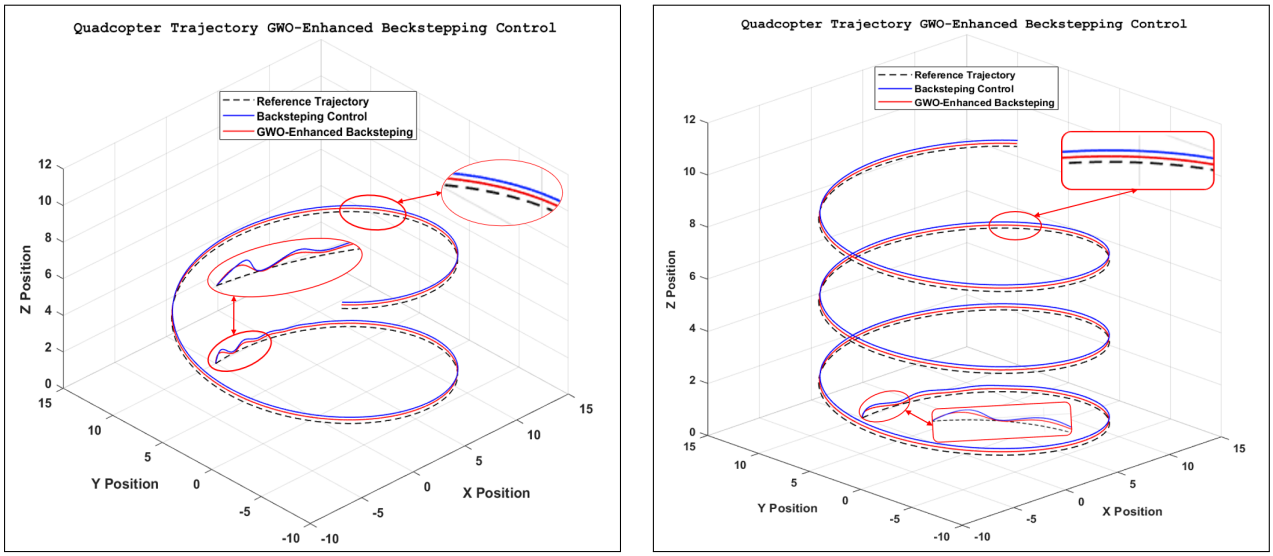
(a) First Trajectory Path (b) Second Trajectory Path

Figure III.8: Quadcopter Tracking Path using Backstepping Controller

A-1 Backstepping with GWO:

As demonstrated in Figure III.8a & Figure III.8b, the backstepping controller is able to follow the track, though it exhibits minor errors along the Z-axis and during the initial few seconds on the X and Y axes.

In contrast, when the GWO-enhanced controller was applied, it showed in Figure III.9 (a & b) that there were significant improvements in performance tracking on both paths. Specifically, it reduced errors along the Z-axis and further minimized deviations on the X and Y axes compared to the backstepping controller. Annex II and Figure III.10 also provides the full position and angle trajectories with explanation zoom as additional details.



(a) First Trajectory Path (b) Second Trajectory Path

Figure III.9: GWO-Enhanced Backstepping Controlling a Quadcopter

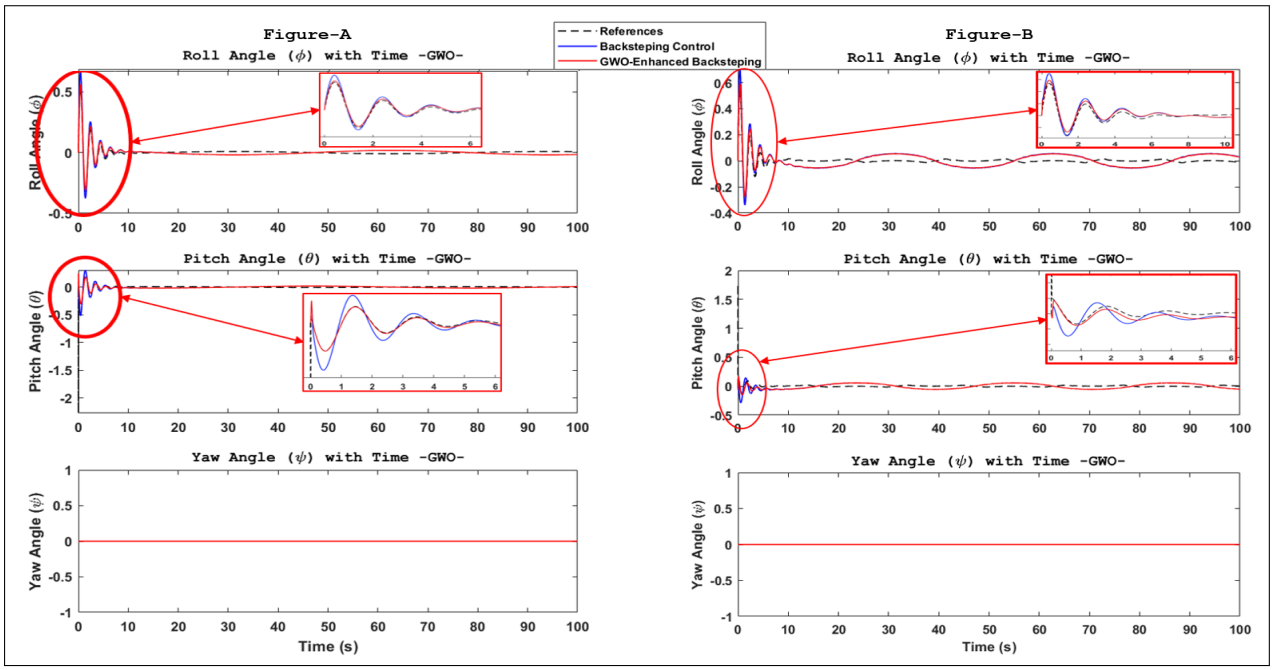
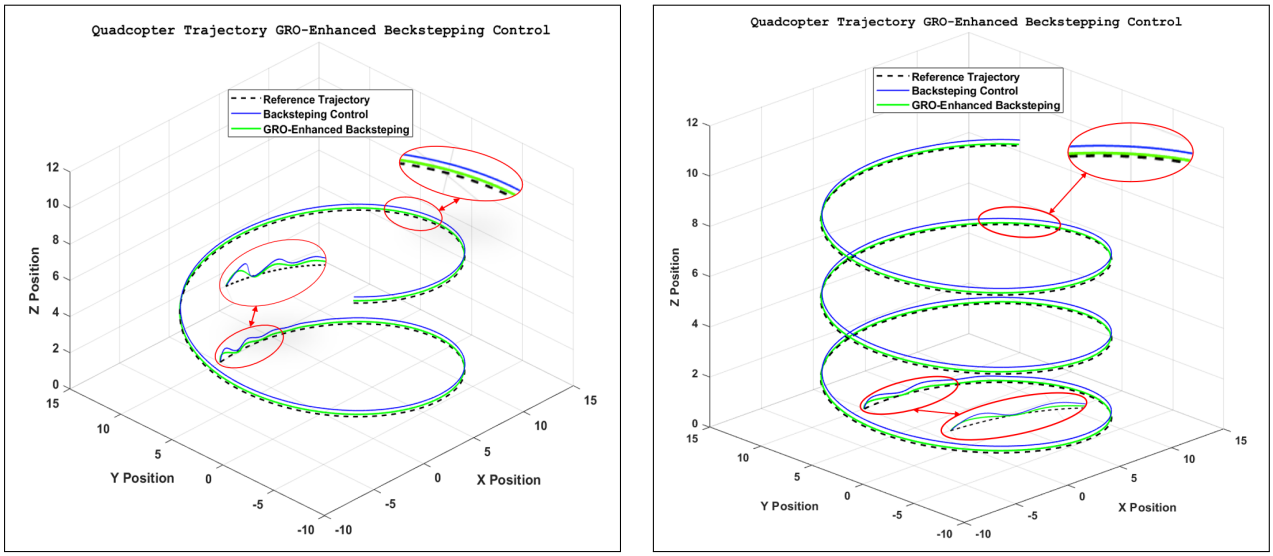


Figure III.10: Orientation angles Phi-Theta-Psi of GWO-Enhanced Backstepping

A-2 Backstepping with GRO:

Figure III.8a & Figure III.8b, shown the success of the backstepping controller in following the tracks but has minor errors along the Z-axis and slight deviations during the initial seconds on the X and Y axes.

The application of the GRO-enhanced controller led to a significant improvement in tracking performance. This controller on both paths minimizes the errors along the Z-axis and further reduces deviations on the X and Y axes compared to the backstepping controller (Figure III.11 (a & b)). Annex II and Figure III.12 includes a detailed explanation of the algorithm and its impact on both positions and orientations.



(a) First Trajectory Path (b) Second Trajectory Path

Figure III.11: GRO-Enhanced Backstepping Controlling a Quadcopter

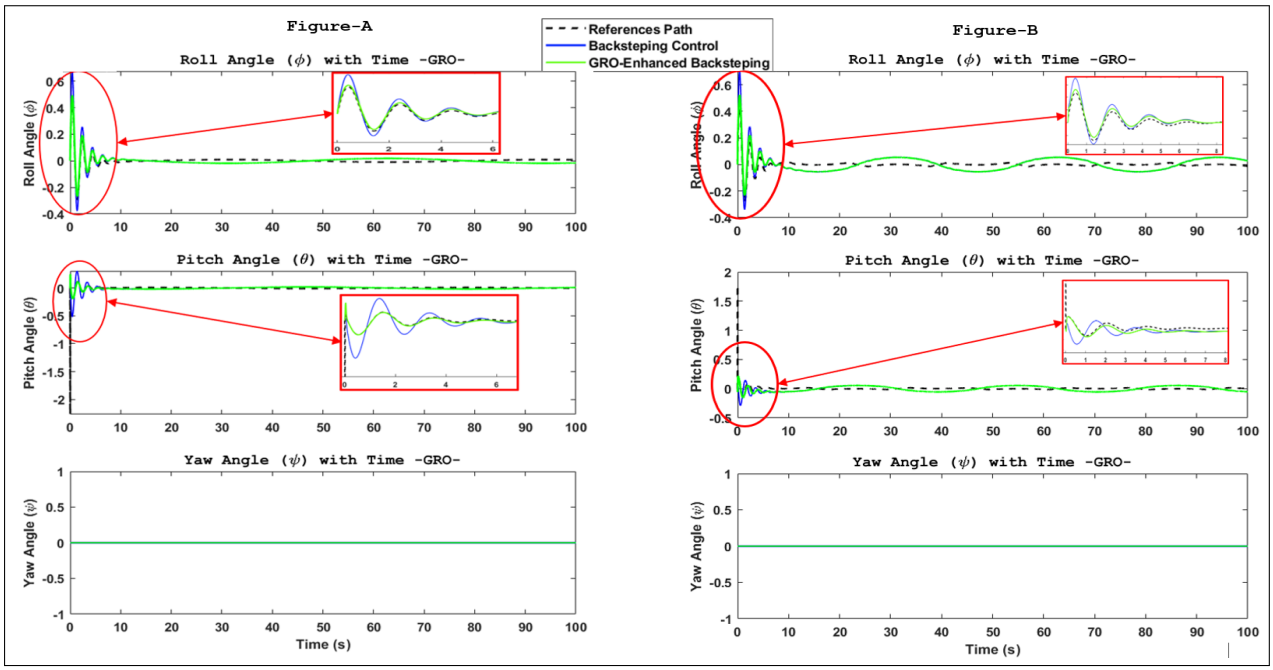
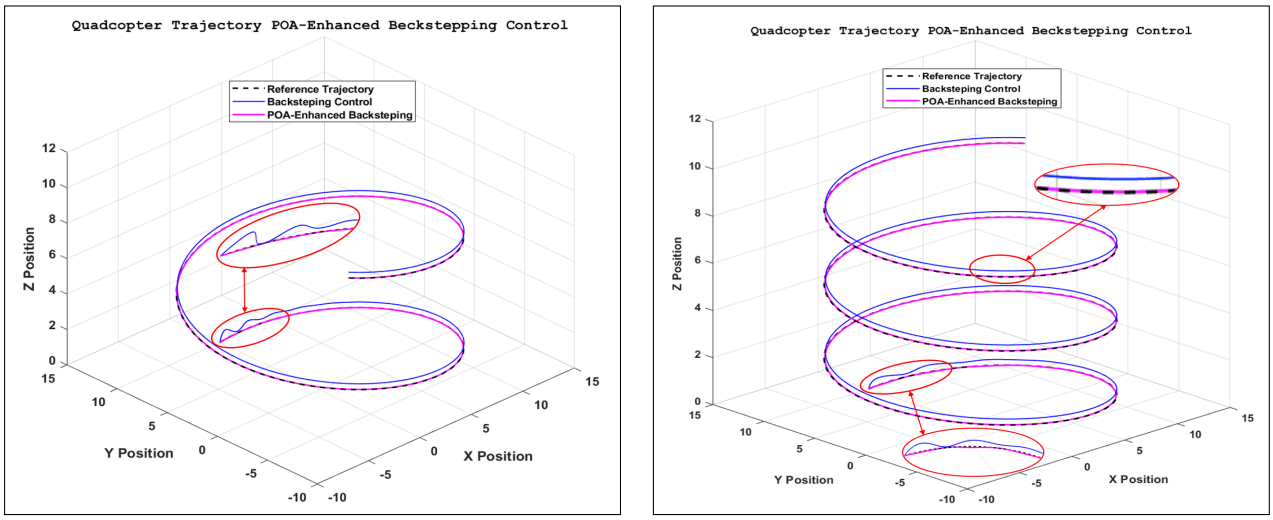


Figure III.12: Orientation angles Phi-Theta-Psi of GRO-Enhanced Backstepping

A-3 Backstepping with POA:

Based on the analysis of the results obtained with the backstepping controller, it can be observed that the system is able to follow the desired track but displays minor errors along the Z-axis and slight deviations during the initial seconds on the X and Y axes.

Otherwise, the use of the POA-enhanced controller results in a substantial improvement in tracking performance on both paths (Figure III.13a & Figure III.13b). It completely eliminates errors along the Z-axis and reduces deviations on the X and Y axes to such a minimal level that these errors become almost imperceptible all this found in Figure III.14 and Annex II.



(a) First Trajectory Path (b) Second Trajectory Path

Figure III.13: POA-Enhanced Backstepping Controlling a Quadcopter

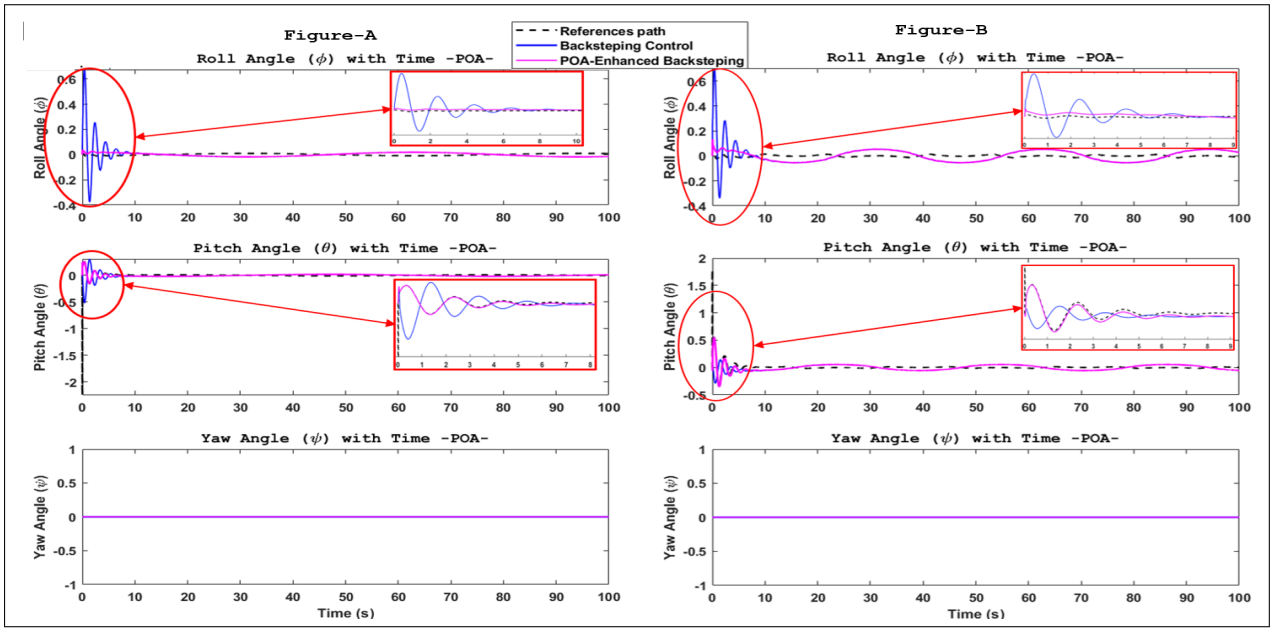
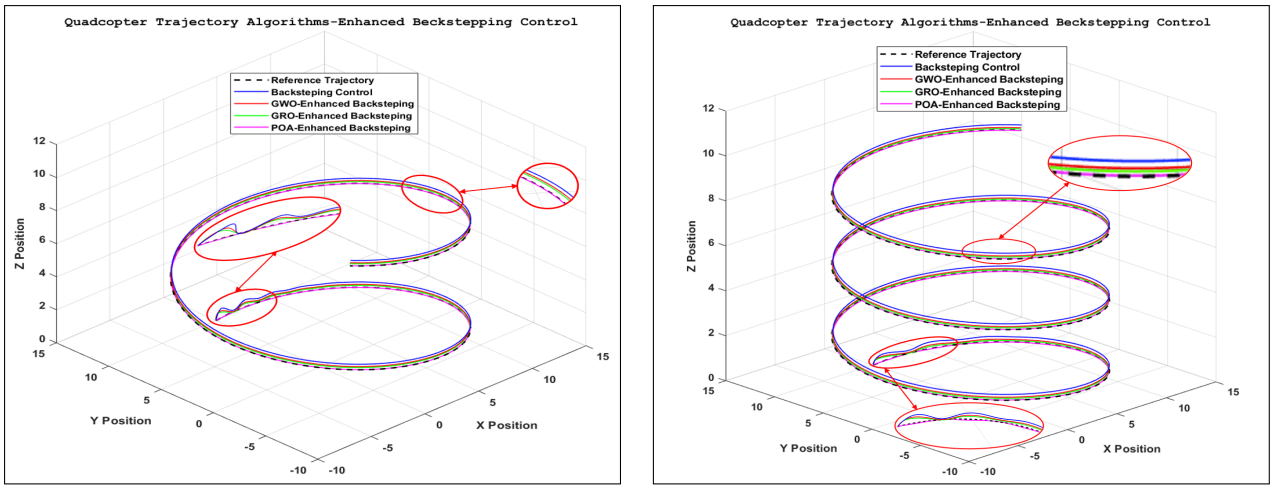


Figure III.14: Orientation angles Phi-Theta-Psi of POA-Enhanced Backstepping

The optimized backstepping controllers—GWO, GRO, and POA—show clear improvements over the basic version (Figure III.15 (a & b)). While the standard controller can follow the path, it shows small Z-axis errors and brief X-Y deviations at the start. GWO reduces these errors noticeably, especially in height tracking. GRO improves further, removing Z-axis errors entirely and cutting X-Y drift. POA delivers the best performance, with near-perfect tracking across all axes. Additional explanatory figures are provided in Annex II.

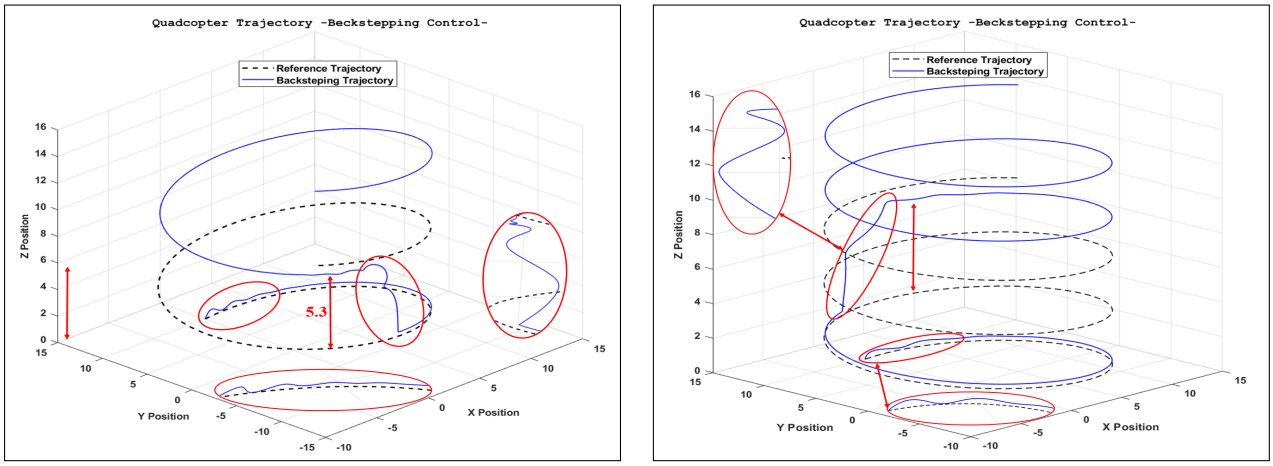


(a) First Trajectory Path (b) Second Trajectory Path

Figure III.15: Comparative of Optimized Backstepping Control Strategies

B- Second Scenario: Trajectory Tracking Assessment, in Windy Environment

In this scenario, we assess the impact of wind disturbances with a step value of 5 m/s on the two trajectory paths analyzed in the first scenario. The wind is introduced at the 30-second mark to evaluate the robustness of each optimizer under challenging external conditions. For the first 30 seconds, the standard backstepping controller initially performs effectively, showing only minor tracking errors, particularly around the wave troughs. Once the wind disturbance is introduced, its performance deteriorates significantly. The controller struggles to maintain alignment with the reference trajectory, leading to noticeable deviations and a clear decline in tracking accuracy in both of the paths under the influence of the external disturbance (Figure III.16 (a & b)).



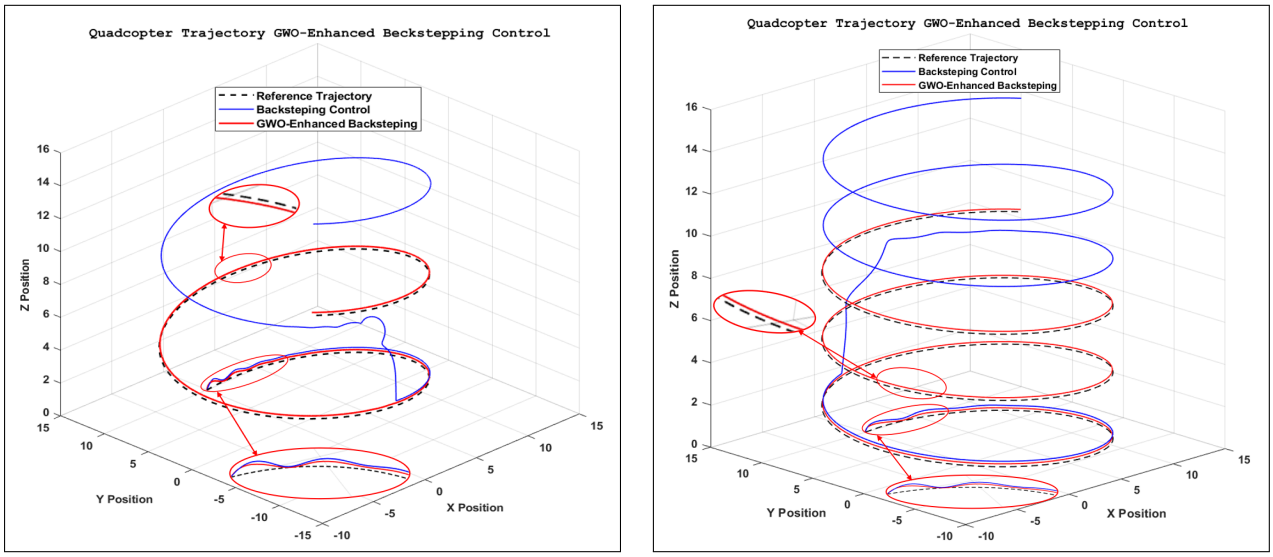
(a) First Trajectory Path (b) Second Trajectory Path

Figure III.16: Quadcopter Trajectory with Backstepping Controller against Wind

B-1 Backstepping with GWO under Wind Disturbance:

While the backstepping controller struggles under wind, the GWO-enhanced controller demonstrates clear improvements, minimizing tracking deviations across all axes and maintaining steady performance throughout the disturbance period.

This enhanced resilience is evident when compared to the unoptimized controller, highlighting the benefits of integrating GWO for robust trajectory tracking in disturbed environments (Figure III.17 (a & b)).

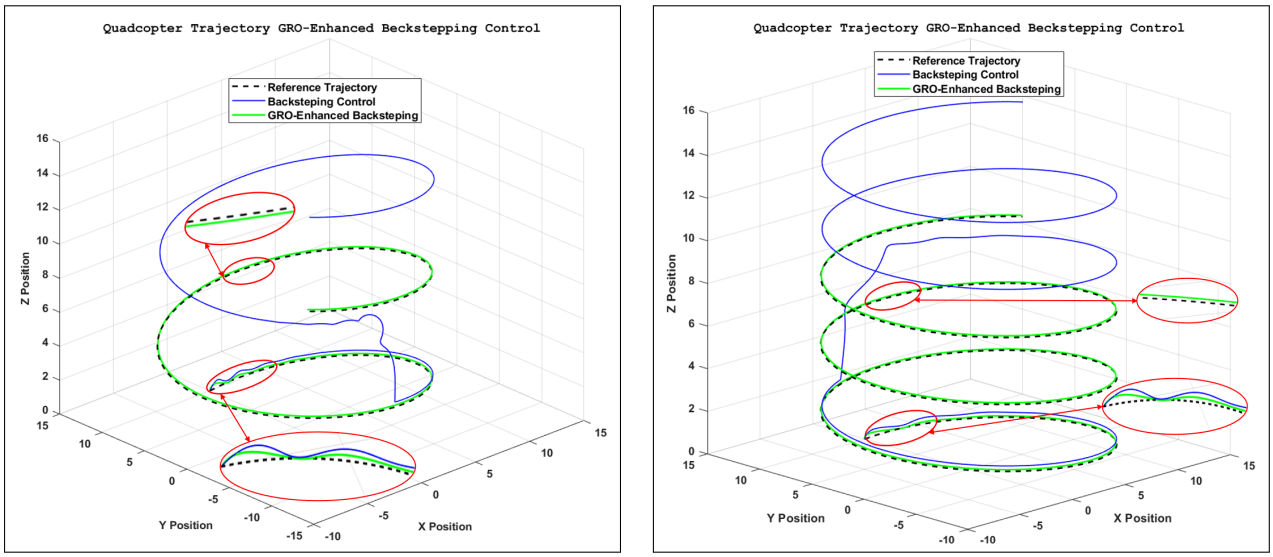


(a) First Trajectory Path (b) Second Trajectory Path

Figure III.17: GWO-Enhanced Backstepping Control of a Quadcopter in Windy Conditions

B-2 Backstepping with GRO under Wind Disturbance:

Applying the GRO-enhanced controller under the same windy conditions yields even stronger performance. It not only maintains accurate tracking along the Z-axis but also significantly reduces deviations on the X and Y axes, outperforming the standard backstepping. The GRO-enhanced controller's robustness under wind demonstrates its effectiveness in precise trajectory alignment despite external disruptions (Figure III.18 (a & b)).

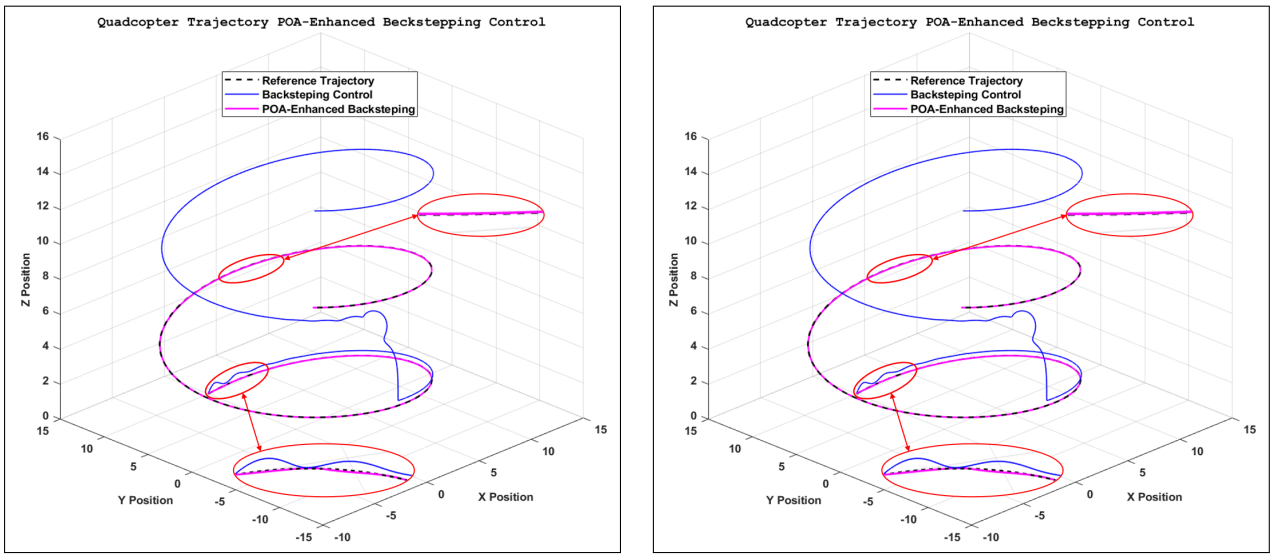


(a) First Trajectory Path (b) Second Trajectory Path

Figure III.18: GRO-Enhanced Backstepping Control of a Quadcopter in Windy Conditions

B-3 Backstepping with POA under Wind Disturbance:

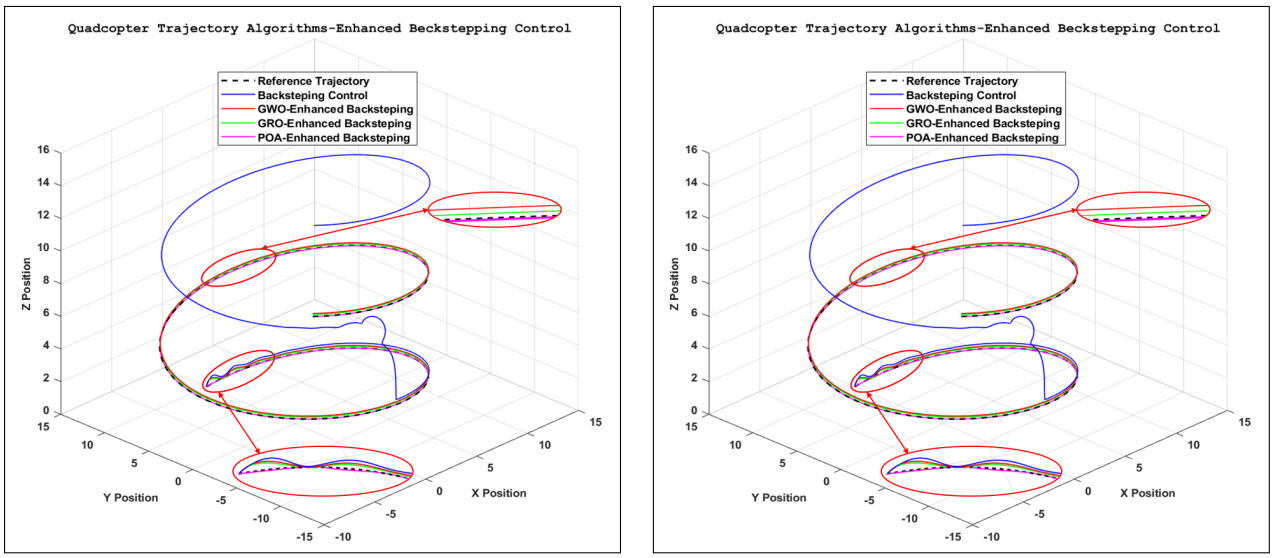
The POA-enhanced controller demonstrates the highest level of resilience among all tested methods. When subjected to wind disturbances, it completely eliminates errors along the Z-axis and reduces deviations on the X and Y axes to levels that are nearly imperceptible, even under challenging environmental conditions. Its ability to maintain near-perfect tracking despite strong external disturbances establishes the POA-enhanced controller as the most robust and reliable solution across both trajectory paths (Figure III.19 (a & b)).



(a) First Trajectory Path (b) Second Trajectory Path

Figure III.19: POA-Enhanced Backstepping Control of a Quadcopter in Windy Conditions

Taken together (Figure III.20 (a & b)), the results from the second scenario demonstrate that while each optimization method improves the backstepping controller's performance under wind, the POA-enhanced controller stands out as the most effective and precise solution.

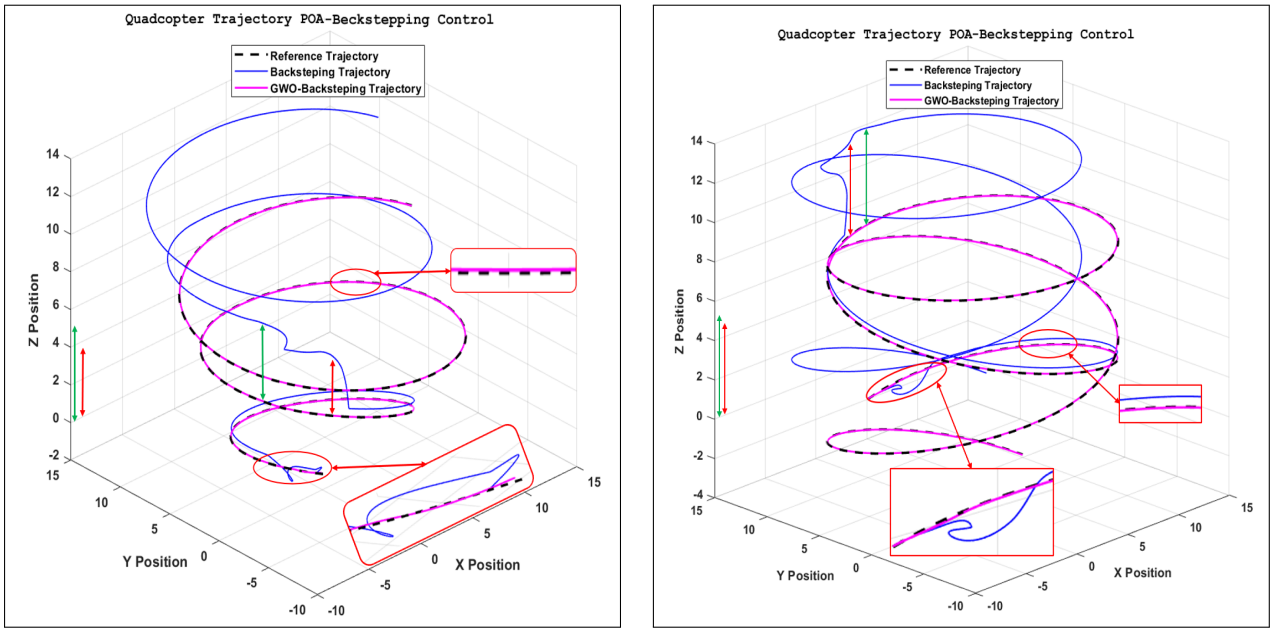


(a) First Trajectory Path (b) Second Trajectory Path

Figure III.20: Comparative Analyses of GWO-GRO-POA-Optimized Backstepping Control Strategies in Windy conditions

III.7.4 Complex Paths Evaluation of POA-Enhanced Backstepping

After confirming the strong performance of the POA-enhanced backstepping controller in reducing tracking errors and significantly improving the standard backstepping method, we now evaluated its capabilities under even more demanding conditions. In this phase, the controller was tested on particularly challenging trajectory paths within a windy environment. The tests included a conical path and a second elliptical path designed to replicate an atomic pattern trajectory. By combining these complex motion paths with the added challenge of wind disturbances, we sought to thoroughly examine the controller's robustness, adaptability, and precision under intensified operational scenarios (Figure III.21a & III.21b). The POA demonstrated remarkable effectiveness by consistently minimizing or eliminating tracking errors on all axes. Even in demanding situations, it closely followed the reference trajectories, showcasing its strong ability to manage complex movements and external disturbances.



(a) Conical Trajectory Path (b) Elliptical Trajectory Path

Figure III.21: New Challenges in Quadcopter Control Using POA-Enhanced Backstepping

III.8 Conclusion

This chapter examined advanced control strategies aimed at improving quadcopter trajectory tracking in the presence of wind disturbances. Specifically, it focused on enhancing backstepping control through the integration of nature-inspired optimization techniques—namely, the Grey Wolf Optimizer (GWO), Garra-Rufa Optimization (GRO), and the Pelican Optimization Algorithm (POA). These approaches were employed to overcome limitations typically associated with classical control schemes such as PID controllers, particularly under dynamic and uncertain conditions.

The study yielded several important insights. All three optimized controllers consistently outperformed the standard backstepping method, with noticeable gains in tracking accuracy. Among them, the POA-based controller stood out, delivering highly precise trajectory tracking and significantly reducing positional and orientation deviations. In terms of robustness, the enhanced controllers demonstrated strong resistance to external disturbances, particularly wind gusts. POA again proved the most effective, maintaining system stability and accuracy even under more demanding scenarios. This resilience highlights its potential for use in real-world environments where external factors can unpredictably affect UAV performance.

Another noteworthy outcome lies in the improved efficiency and adaptability of the control system. By automating the tuning of backstepping gains, the optimization algorithms reduced manual intervention while enabling the controllers to adjust more effectively to changing flight conditions. POA, in particular, achieved an effective balance between exploration and exploitation, leading to faster convergence and greater control precision.

Integrating backstepping with bio-inspired optimization has proven to be a powerful strategy for enhancing quadcopter control. Among the tested methods, POA-enhanced backstepping emerged as the most promising solution, offering a high degree of reliability and precision even in challenging operating environments.

Future studies could focus on real-time implementation of these optimized strategies and assess their scalability across various unmanned aerial systems. Additionally, further exploration could involve applying these algorithms to alternative modeling frameworks, such as state-space representations, to evaluate their effectiveness in energy-efficient control and dynamic response optimization.

CHAPTER IV

ENERGY EFFICIENT WITH PATH FOLLOWING FOR QUADCOPTER USING LQR WITH INTELLIGENT TUNING

IV.1 Introduction

The evolution of UAVs, particularly multirotor platforms such as quadcopters, has significantly expanded the horizons of automation across various domains. Applications now range from logistics and environmental monitoring to infrastructure inspection and surveillance [105]. These aerial systems are increasingly expected to operate autonomously, execute complex maneuvers, and sustain prolonged flight durations, all while navigating dynamic and often unpredictable environments [106]. A key challenge in this context is ensuring that UAVs remain both precise and responsive in their motion, while also operating with high energy efficiency [105].

Among the various UAV configurations, the six-degree-of-freedom (6-DOF) quadcopter presents a particularly complex control problem. It must simultaneously regulate its position and orientation, responding in real time to both environmental disturbances and mission-specific demands [105]. This complexity arises from its nonlinear dynamics and the strong coupling between translational and rotational motion. Traditional control methods, such as the Proportional-Integral-Derivative (PID) controller and the Linear Quadratic Regulator (LQR) [106], have been widely employed due to their simplicity and well-established theoretical foundations [107]. However, their effectiveness diminishes under nonlinear or time-varying conditions unless supported by adaptive control mechanisms.

In response to these limitations, this chapter presents a simulation-based investigation of two optimization-enhanced control strategies aimed at achieving energy-efficient path following for 6-DOF quadcopters. The first approach utilizes an adaptive LQR controller in which the weighting matrix Q, and consequently the feedback gain K, are dynamically adjusted using the Grey Wolf Optimizer (GWO) algorithm [107]. This method retains the foundational strengths of LQR while introducing the adaptability necessary to respond to variations in system dynamics and external disturbances [108]. The second strategy builds on the same LQR framework but introduces a two-stage enhancement using feedforward neural networks (FNNs). Initially, an FNN is employed to adapt the Q matrix, leveraging the neural network's capacity to search and optimize across large parameter spaces [109][110]. This provides a basis for comparison with the GWO-based approach. Subsequently, a second FNN is introduced to adapt the R matrix, with the objective of fine-tuning the thrust and torque distributions to further minimize energy consumption [109].

The chapter begins with a comprehensive overview of the quadcopter's mathematical model, encompassing both its nonlinear dynamics and the linearized state-space representation used in controller design. Based on this model, the proposed control strategies are formulated and evaluated within a unified simulation framework. The results, presented in the concluding sections, offer a comparative assessment of traditional and intelligently tuned controllers, demonstrating their potential to meet the growing demands of autonomous, energy-efficient aerial systems.

IV.2 Mathematical Model and Control Design

To design an effective control strategy for a quadcopter, it is essential to begin with an accurate mathematical representation of its dynamics.
In this study, the quadcopter is modeled as a rigid body operating in three-dimensional space, exhibiting six degrees of freedom. These include three translational motions along the x, y, and z axes, and three rotational motions -roll (ϕ), pitch (θ), and yaw (ψ)- about the corresponding axes [110]. The motion of the vehicle is described using the Newton-Euler equations, which establish the relationship between the forces and torques acting on the system and the resulting linear and angular accelerations [111].
The primary control inputs are the thrust forces produced by the individual rotors and the torques generated by adjusting their relative speeds [111]. The system outputs typically include the quadcopter's position, orientation, and angular velocities. These dynamics are inherently nonlinear and exhibit significant coupling, particularly when factors such as aerodynamic drag, payload variations, and proximity to the ground are considered [111].
For control design purposes, particularly in model-based methods like the Linear Quadratic Regulator (LQR), it is standard practice to simplify the nonlinear model by linearizing it around a fixed operating point, typically a hover condition [111]. This linearization yields a state-space model that is more tractable and suitable for real-time control implementation.

IV.2.1 State Variables and Control Inputs

The state vector $x \in R^{12}$ includes [111]:

$$\left[x, z, y, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}\right]^T$$

The control input vector $u \in R^4$ is defined as:

$$u = [T, \tau_\phi, \tau_\theta, \tau_\psi,]$$

where:

- T is the total thrust from all rotors
- $\tau_\phi \tau_\theta \tau_\psi$ are torques about the roll, pitch, and yaw axes, respectively

The output vector y includes position and orientation variables used for feedback.

IV.2.2 Linearized State-Space Representation

The dynamic model is expressed in the following standard state-space form:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

The state-space representation used for controller design is constructed by linearizing the nonlinear dynamic equations around a steady-state operating point [112]. From this process, the system matrices A, B, C, and D are derived based on the fundamental dynamic relationships previously established in Equations II.39 through II.43. Specifically, the matrix A defines the internal dynamics of the system by describing how the state variables evolve over time in the absence of inputs. The matrix B characterizes how control inputs influence the state variables, effectively linking the actuator commands to changes in the system's behavior. The matrix C maps the internal states to measurable output variables, enabling observation and feedback. Finally, the matrix D accounts for any direct influence of the inputs on the outputs, although in most practical cases involving quadcopters, this matrix is zero due to the absence of such direct feedthrough.

IV.2.3 Matrix Definitions

The matrices used in simulation are defined as follows:

Matrix A: System Dynamics

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix B: Input Mapping

$$B = \begin{bmatrix} 0 & 0.0555 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 10.000 & 0 & 0 & 0 \\ 0 & 10.000 & 0 & 0 \\ 0 & 0 & 6.6667 & 0 \end{bmatrix}$$

Matrix C: Output Mapping

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix D: Feedthrough (Zero)

$$D = \begin{bmatrix} \mathbf{0}_{4\times 12} \end{bmatrix}$$

The linearized state-space model serves as the foundation for the control design strategies developed in this study [113]. The associated matrices represent an idealized hover condition, based on small-angle approximations and the assumption of constant rotor thrust at equilibrium. Although these assumptions simplify the dynamics, the resulting model retains the key characteristics necessary for effective controller synthesis [111]. Its validity is confirmed through simulation by comparing its behavior with that of the full nonlinear model, demonstrating that it captures the essential dynamic responses of the quadcopter under typical operating conditions [111].

IV.3 Linear Quadratic Regulator (LQR) Control

This section focuses on the application of the Linear Quadratic Regulator (LQR) to the linearized 6-DOF quadcopter model. LQR is a state-feedback control technique that aims to determine the optimal control inputs required to minimize deviations from a desired state trajectory, while simultaneously reducing overall energy expenditure [114]. It is particularly well suited for systems in which the full state vector is measurable or observable, offering a well-balanced trade-off between tracking performance and control effort [113].

As an optimal control strategy, LQR stabilizes dynamic systems by minimizing a quadratic cost function that penalizes both deviations in state variables and excessive control actions [114]. This formulation leads to smooth and efficient control inputs, making LQR a robust and computationally efficient choice for real-time applications such as quadcopter flight control [111].

IV.3.1 Optimal Control Objective

The objective of the Linear Quadratic Regulator (LQR) controller is to regulate the state of a dynamic system, such as a quadcopter, in an optimal and efficient manner. In this context, "optimal" refers to minimizing a predefined cost function that captures the trade-off between tracking performance and energy consumption [114]. This balance is particularly important in aerial systems, where aggressive control actions can achieve high-precision maneuvering but often result in excessive energy usage [111]. Conversely, overly conservative control strategies may conserve energy but lead to degraded tracking accuracy and potential instability [115]. The LQR framework addresses this challenge by formulating a control law that ensures both responsive behavior and energy-aware operation [115].
The performance index or cost function to be minimized by LQR is defined as [111]:

$$J = \int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t) \right) dt \tag{IV.1}$$

Where:

- $x \in R^n$ is the state vector, including positions, velocities, and angles of the quadcopter
- $u \in R^m$ is the control input vector (thrust and torques)
- Q is a positive semi-definite matrix that penalizes state error
- R is a positive definite matrix that penalizes large control inputs

The cost function used in the LQR framework evaluates performance by integrating a weighted sum of the state deviations and the control efforts over time, with the goal of minimizing this cumulative measure [111]. It consists of two key components:

- **State penalty term:** $x(t)^T Q x(t)$ This term penalizes deviations of the system's state from its desired trajectory. The weighting matrix Q assigns importance to each state variable, allowing the designer to emphasize certain performance aspects [115]. For example, if maintaining altitude is more critical than controlling the yaw angle, the corresponding diagonal element in Q related to altitude would be assigned a higher value.
- **Control penalty term:** $u(t)^T R u(t)$ This term penalizes the use of control inputs, effectively limiting the intensity of actuator commands. The matrix R specifies the relative cost of applying control actions such as motor thrusts or torques [115]. Larger values in R discourage aggressive maneuvers, promoting energy efficiency and reducing the mechanical strain on the propulsion system [110].

Together, these terms guide the controller to strike an optimal balance between accurate trajectory tracking and prudent use of energy and actuation.

IV.3.2 Control Law and Gain Calculation

Once the control objective has been formally defined through the cost function J, the next step involves determining a control law that minimizes this objective function [116]. Within the Linear Quadratic Regulator (LQR) framework, the resulting control strategy takes the form of a state-feedback law. In this approach, the control input $u(t)$ is calculated as a linear function of the system's current state, effectively feeding back the state variables to determine the appropriate control action [116]. This can be expressed as:

$$u(t) = -Kx(t) \tag{IV.2}$$

Here:

- $x(t)$ is the current state vector of the system
- K is the **state feedback gain matrix**, which determines how much influence each state variable has on the control input

Here, K is the optimal gain matrix that weights each state variable according to its contribution to the overall control objective [116]. The negative sign indicates that the controller acts to reduce deviations from the desired state [116]. This formulation ensures that the control inputs are not only responsive to the system's dynamics but also optimized with respect to performance and energy efficiency.

A. Computing the Feedback Gain K

To find the optimal gain matrix K, one must solve the **Algebraic Riccati Equation (ARE)**:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{IV.3}$$

Where:

- A and B are the state and input matrices from the system's linearized model
- Q and R are the weighting matrices from the cost function
- P is the unique positive semi-definite solution of the Riccati equation

P can be interpreted as a cost-to-go matrix, representing the cumulative future cost starting from a given state. Once the matrix P is determined -typically through numerical methods such as MATLAB's *care* function- the corresponding gain matrix K can be calculated using the expression:

$$K = R^{-1}B^T P \tag{IV.4}$$

This formulation highlights that the optimal control gain is influenced not only by the intrinsic dynamics and structure of the system, characterized by the matrices A and B, but also by the specific trade-offs selected by the designer [116]. In particular, the weighting matrices Q and R reflect the relative importance assigned to performance objectives versus control effort, thereby shaping the final control law accordingly [116].

IV.3.3 B. Interpretation for Quadcopter Systems

In the case of a 6-DOF quadcopter, the feedback gain matrix K is designed to map deviations in position, velocity, and orientation to the appropriate control inputs, specifically thrust and rotational torques [117]. For example, assigning a relatively high gain to the vertical velocity component can lead to quicker stabilization in altitude. Conversely, tuning moderate gains for angular velocity components helps to mitigate overshoot and dampen oscillatory behavior in the drone's orientation [117].
By implementing the feedback control law $u(t) = -Kx(t)$, the controller dynamically modifies the thrust and torque commands in response to the quadcopter's real-time state [117]. This mechanism enables the drone to follow the desired trajectory accurately while minimizing unnecessary energy consumption and excessive actuator activity.

IV.3.4 Benefits of LQR in Quadcopter Control

The Linear Quadratic Regulator (LQR) presents a range of compelling advantages when utilized for controlling quadcopters, particularly in scenarios where the system dynamics are linearized around a specific operating point, such as hovering [117]. Although rooted in classical control theory, the LQR framework remains highly relevant for contemporary unmanned aerial systems (UAS) due to its well-defined mathematical formulation and compatibility with real-time implementation. This section highlights the principal benefits of LQR, emphasizing its effectiveness in enhancing trajectory tracking accuracy, promoting energy-efficient control actions, and contributing to overall flight stability.

A. Balanced Trade-off Between Performance and Energy Use

A fundamental advantage of the Linear Quadratic Regulator (LQR) lies in its capacity to formalize the trade-off between control precision and actuator effort through a well-defined cost function [118]. This capability is particularly important in quadcopter applications, where achieving agile and responsive maneuvers often comes at the expense of increased energy consumption and reduced battery longevity [118]. The flexibility of the LQR design allows for careful tuning of the weighting matrices Q and R, enabling the controller to accommodate a variety of flight scenarios. For instance, the control strategy can be adjusted to prioritize high performance during critical phases of flight, or to favor energy conservation during extended missions, depending on the operational objectives.

B. Smooth and Stable Control Actions

In contrast to gain-scheduled or heuristic control approaches, the Linear Quadratic Regulator (LQR) provides mathematically optimal control signals based on a rigorous formulation. This optimality results in smoother transitions, reduced oscillatory behavior, and minimal overshoot in system response [118]. These characteristics are particularly valuable in quadcopter control, as such systems are inherently unstable and require continuous corrective action to maintain stable flight. The improved stability and precision offered by LQR not only enhance overall flight accuracy but also contribute to a more stable platform for sensitive payloads, including cameras, environmental sensors, and delivery packages.

C. Fast Real-Time Computation

After the gain matrix K has been computed offline, the implementation of the control law $u(t) = -Kx(t)$ becomes computationally efficient. This efficiency makes the Linear Quadratic Regulator (LQR) particularly well-suited for embedded systems that operate with limited processing capabilities. The feedback control requires only a matrix-vector multiplication, a relatively simple operation that can be performed quickly even on low-power microcontrollers. As a result, LQR enables the execution of high-frequency control loops, typically in the range of 200 to 500 Hz, which are essential for maintaining stable and responsive flight in quadcopter systems.

D. Robustness to Small Disturbances

While the Linear Quadratic Regulator (LQR) is inherently formulated for linear systems, it demonstrates robust performance in the vicinity of the operating point, especially when external disturbances are limited and system parameters remain within nominal ranges [117]. In practical applications, this robustness enables the controller to effectively handle minor perturbations, such as wind gusts or sensor noise, without relying on complex adaptive or nonlinear compensation strategies [118]. This capability contributes to the reliability and simplicity of LQR-based control in real-world quadcopter operations.

E. Tuning Flexibility and Design Intuition

The LQR framework allows the designer to directly influence controller behavior through the structure of Q and R. For instance:

- Increasing the weight on position errors (in Q) sharpens trajectory tracking
- Increasing the weight on control inputs (in R) results in more conservative, energy-efficient flight

This flexibility gives the user full control over how the UAV behaves without changing the underlying structure of the controller.

F. Strong Theoretical Foundation

The Linear Quadratic Regulator (LQR) is supported by decades of rigorous research in control theory and engineering practice [119]. Its properties related to stability, convergence, and optimality are well-established, offering a solid theoretical foundation that inspires confidence during both the validation and deployment phases [119]. Furthermore, LQR serves as a cornerstone for the development of more advanced control strategies, including Linear Quadratic Gaussian (LQG) control and Model Predictive Control (MPC). This foundational role makes LQR an excellent starting point for enhancing system performance and exploring more sophisticated control architectures.
When implemented in a quadcopter system, the Linear Quadratic Regulator (LQR) offers several notable advantages [119]:

- It improves trajectory tracking while minimizing overshoot
- It promotes energy efficiency by reducing unnecessary control actions
- It ensures a stable and responsive flight behavior
- And it remains computationally lightweight and analytically tractable

These characteristics position LQR as a highly practical and effective control strategy for a wide range of quadcopter applications, including autonomous navigation, aerial surveillance, and payload delivery. Its structured and tunable design [119], combined with its suitability for energy-limited platforms, makes it particularly valuable for modern unmanned aerial systems operating in dynamic and constrained environments [119].

IV.4 Neural Network Integration in Quadcopter Control

Neural networks possess a remarkable capacity to model and adapt to complex, nonlinear, and time-varying systems [120]. This section examines how artificial neural networks (ANNs) can complement and enhance traditional control methods such as the Linear Quadratic Regulator (LQR), with the aim of improving both energy efficiency and trajectory accuracy in quadcopter systems.
Modern UAVs, including quadcopters, often operate in environments where precise mathematical models are either overly complex or fail to capture important real-world variations [110]. These variations may include wind disturbances, changes in payload, and nonlinear aerodynamic effects, which challenge conventional control approaches [111]. To overcome these challenges, artificial intelligence techniques - particularly neural networks- are increasingly incorporated into control frameworks to enhance system adaptability and robustness [111]. Inspired by the structure of the human brain, a neural network consists of interconnected layers of processing units known as neurons [120]. Through a process of learning from data, these networks can approximate complex nonlinear functions, making them especially suitable for applications where explicit modeling is difficult or impractical [120].
In control systems, neural networks can fulfill a variety of roles, including [110]:

- Modeling the underlying system dynamics
- Estimating states that are difficult or impossible to measure directly
- Learning effective control policies
- Optimizing controller parameters for improved performance

Among the different types of neural networks, Feedforward Neural Networks (FNNs) are commonly employed due to their relative simplicity, strong generalization capabilities, and low computational demands during inference [111]. In the present work, FNNs are integrated into the quadcopter control architecture with two primary objectives: first, to enable adaptive tuning of the LQR controller, and second, to intelligently adjust the trajectory mapping through a movement matrix. These enhancements and their implementation details are presented in the following subsections.

IV.4.1 Adaptive Control with Feedforward Neural Networks (FNNs)

In this study, Feedforward Neural Networks (FNNs) are employed to dynamically adapt the LQR gain structure and refine the mapping between desired trajectories and control inputs in real time [111]. This integration aims to enhance overall system performance while promoting greater energy efficiency during quadcopter operation.

A. Adaptive Gain Tuning

The first Feedforward Neural Network (FNN) is designed to dynamically adjust the LQR controller's weighting matrix Q, and thereby indirectly influence the feedback gain matrix K, based on real-time flight conditions [111]. Unlike conventional approaches that depend on fixed, pre-tuned parameters, this neural module continuously analyzes both current and historical flight data to determine appropriate adjustments to the cost function. As a result, the controller can adaptively modify its trade-off between trajectory precision and energy efficiency, enabling more responsive and context-aware behavior throughout various phases of flight.

Inputs:

- State vector $x(t)$
- Reference signal x_{ref}
- Tracking error $e(t)$
- Estimated disturbances or flight phase indicators

Outputs:

- Updated matrices $Q(t)$
- Automatedly and directly the gain matrix $K(t)$ will be tuned

The Feedforward Neural Network (FNN) is trained offline using a diverse set of simulated scenarios that capture a range of disturbance intensities and mission phases [121].
Once deployed, the network operates in real time, using sensor feedback and mission context to infer the optimal control characteristics. This enables the system to respond intelligently

to changing flight conditions without requiring manual retuning or external intervention.

B. Trajectory Adjustment via Movement Matrix Editing

A second Feedforward Neural Network (FNN) is designed to adapt the weighting matrix R, which directly influences the distribution of control effort across the quadcopter's actuators [111]. Specifically, R is implemented as a 4×4 diagonal matrix, with each entry corresponding to the control cost associated with thrust and the rotational torques around the roll (τ_x) pitch (τ_y), and yaw (τ_z) axes [111]. By adjusting this matrix in real time, the FNN enables the controller to modulate how aggressively it responds to trajectory errors, effectively tuning the balance between control intensity and energy expenditure [121]. This adaptive behavior is particularly valuable under conditions involving model mismatch or external disturbances (if exist), as it allows the system to reinterpret desired trajectories more flexibly through a feedforward mechanism that complements the primary feedback control.

Function:

$$u(t) = M(t) \cdot (x_{ref} - x(t))$$

Inputs:

- Desired trajectory the main Matrix $M \in R^{4\times14}$
- Current state
- Mission context (e.g., hovering, cruising, turning)

Output:

- Updated matrix $M(t)$

This neural adjustment ensures more precise tracking, even in conditions where the physical model alone is insufficient to capture the true system behavior.

IV.4.2 Advantages of Neural-Augmented LQR Control

By integrating neural network-based adaptation with the classical Linear Quadratic Regulator (LQR), the proposed hybrid control strategy offers several key advantages [122]:

- **Enhanced adaptability**: The system can adjust in real time to varying flight phases and environmental disturbances[122].
- **Improved energy efficiency**: By dynamically tuning cost function weights and trajectory mappings, the controller avoids unnecessary control actions[122].
- **Reduced modeling complexity**: The neural components help compensate for unmodeled dynamics, reducing the need for repeated derivation of precise mathematical models[122].
- **Stable and smooth response**: Neural modules are designed to operate continuously and seamlessly, without compromising system stability[122].

This hybrid approach preserves the analytical rigor and stability guarantees of traditional LQR while incorporating the flexibility and learning capability of neural networks[123]. As a result, it forms a robust and intelligent control framework, well-suited for energy-conscious and performance-critical quadcopter applications.

IV.5 Simulation and Results

This section presents and interprets the simulation results obtained using the proposed control strategies applied to a 6-DOF quadcopter model. The analysis begins with the implementation of a standard Linear Quadratic Regulator (LQR), which serves as a baseline controller. Gradually, more advanced improvements are introduced, including optimization techniques based on metaheuristics and adaptive neural network components. This work draws inspiration from the findings of Chun-Wei, who showed that modifying the trajectory shape within an LQR framework can significantly reduce energy consumption, lowering it from **38 units** to approximately **11.9 units**, without degrading control accuracy.Taking this result as a reference, the current study aims to achieve even greater energy efficiency, beginning from the same initial value of **38 units**. To accomplish this, several enhancements are applied, as discussed in previous sections. These include dynamic tuning of control gains, adaptive adjustment of cost function weights in real time, and intelligent modification of the desired trajectory through feedforward correction.
In addition to energy optimization, this study also explores the practical task of guiding the quadcopter through trajectories that outline specific surface areas. As a first step, the drone is directed to reach a set of key positions that define a closed path forming a quadrilateral, typically a square or a rectangle. The path is defined using only the lengths of two adjacent sides. This aspect of the work not only facilitates analysis of energy consumption during area coverage maneuvers, but also allows investigation of how surface-constrained trajectory design affects both control performance and energy efficiency.

IV.5.1 Simulation

The simulation is based on a nonlinear dynamic model of a quadcopter system using the linearized state-space form defined by:

$$\dot{x} = A \cdot x + B \cdot uy = C \cdot x + D \cdot u$$

This 6-DOF model employed in this study captures the complete motion of the quadcopter in three-dimensional space, incorporating both translational and rotational dynamics. This comprehensive model was developed and implemented using MATLAB/Simulink, as illustrated in Figure IV.1. To achieve accurate trajectory tracking, the system was initially controlled using a Linear Quadratic Regulator (LQR), serving as the foundational control strategy for subsequent enhancements.

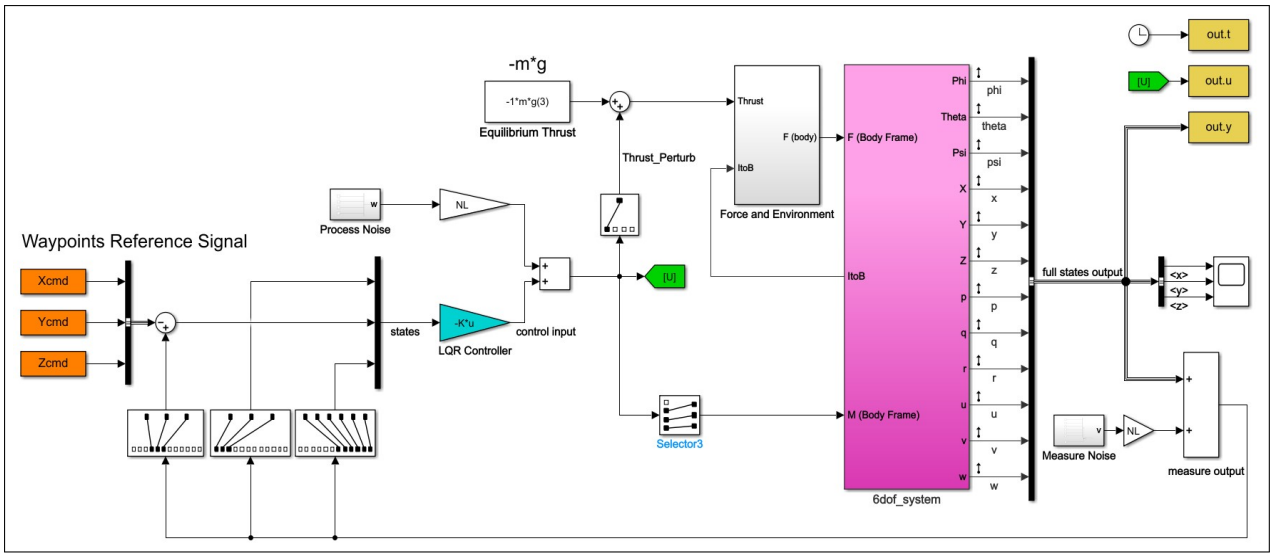


Figure IV.1: Simulation Block Diagram of LQR controlling a Quadcopter

A. LQR Design Process

To implement a Linear Quadratic Regulator (LQR) for a quadcopter system, the nonlinear 6-DOF model is first linearized around a hovering condition [124]. This process yields a linear state-space representation, which is stored in the file "Model.mat" for controller design and analysis.

A.1 System States

The model considers a total of twelve state variables, capturing both the rotational and translational dynamics of the quadcopter[125]. These include three rotational angles (roll, pitch, and yaw), three positional coordinates (X, Y, and Z), three angular velocities (p, q, and r), and three linear velocities (u, v, and w)[125].

A.2 Selection of Cost Matrices

In the LQR's framework, the design of the cost matrices Q and R plays a pivotal role in shaping the controller's behavior. The matrix Q penalizes deviations in the state vector from the desired trajectory, encouraging accurate tracking. In contrast, R penalizes the use of control inputs, promoting energy-efficient behavior. To simplify the tuning process and decouple the influence of individual states and inputs, both matrices are selected as diagonal matrices[116].

For this particular system, Q is defined as a 12×12 diagonal matrix. The first six diagonal entries correspond to penalties on roll (ϕ), pitch (θ), yaw (w), and the translational positions along the X, Y, and Z axes. Specifically:

- $Q(1,1)$ addresses roll deviation
- $Q(2,2)$ targets pitch deviation
- $Q(3,3)$ penalizes yaw deviation
- $Q(4,4)$, $Q(5,5)$, and $Q(6,6)$ penalize position errors in the X, Y, and Z directions, respectively

The remaining six elements, which relate to angular and linear velocities, may be assigned smaller weights or zeros, depending on the control objectives.

The R matrix is defined as a 4×4 diagonal matrix that regulates the control inputs: total thrust and torques about the roll, pitch, and yaw axes. More precisely:

- $R(1,1)$ penalizes total thrust
- $R(2,2)$ regulates roll torque
- $R(3,3)$ adjusts pitch torque
- $R(4,4)$ governs yaw torque

By appropriately tuning Q and R, a balance can be achieved between aggressive control performance and energy conservation.

A.3 Gain Matrix Computation

Once Q and R, are defined, the LQR algorithm is employed to compute the optimal feedback gain matrix K [116]. This matrix is then integrated into the linearized system to close the feedback loop and enforce the desired dynamic behavior.

A.4 Integration into the Nonlinear Model

After verifying the controller's effectiveness in the linearized environment, the computed gain matrix K is implemented in the full nonlinear 6-DOF quadcopter model to assess real-world applicability [126].

In the Simulink environment, the K matrix is applied to the nonlinear model by placing it ahead of the "6dof_system" block. The controller operates on a 12-dimensional state error vector, representing the difference between the current and target states [127]. It calculates the control input vector u using the relationship $u = -Kx$, where x is the state error vector. The resulting outputs—total thrust and three torque commands—are fed directly into the nonlinear dynamics block to control and stabilize the quadcopter [127].

A.5 Simulation and Validation

Throughout the simulation, desired reference trajectories (typically in terms of X, Y, and Z positions) are provided as input setpoints. The LQR controller continuously updates the control inputs to minimize the error between the actual and target states, thereby driving all components of the state error vector toward zero. This transition from linear model validation to nonlinear system simulation serves as a critical step in evaluating the robustness and practical viability of the LQR controller in realistic operating conditions [127].

B. The reference trajectories Matrix

The reference trajectory for the quadcopter is designed to follow the edges of a **square surface**. This test scenario is used to evaluate the system's ability to handle sharp corners and maintain stable flight while completing the loop. The simulation showed that the basic LQR controller offers accurate trajectory tracking. However, due to frequent and sharp directional changes required to follow the square corners, the total energy consumed by the drone during this mission was measured at **38 units**.

C. Quadcopter parameters:

Key parameters such as total mass, arm length, thrust and drag coefficients, as well as the moments of inertia about each principal axis, are essential for accurately modeling the system and designing effective control algorithms. These values form the foundation for both theoretical analysis and practical implementation. Table IV.1 provides a detailed summary of the specific parameters adopted for the quadcopter utilized in this research.

Table IV.1: LQR-Quadcopter parameters

Type	**Parameters**	**Units**
X, Y moment of inertia	I_{xx}, I_{yy}	$10 \times 10^{-2}\,\mathrm{kg.m^2}$
Z moment of inertia	I_{zz}	$15 \times 10^{-2}\,\mathrm{kg.m^2}$
Acceleration Gravity	g	$9.8\,\mathrm{m/s^2}$
Quad Mass	m	$1.8\,\mathrm{kg}$
Grey Wolf Optimization Parameters	Max of wolves	50
	Max Iterations	100
Sampling time	Ts	$0.01s$
Wind	W	$0\,\mathrm{m/s}$
Body rate	u, v, w	0

IV.5.2 Optimization-Based Tuning of the LQR Controller Using GWO

While the Linear Quadratic Regulator (LQR) controller, designed using manually selected Q and R matrices, performs adequately under simplified linear dynamics, its effectiveness can be significantly reduced when applied to the complete nonlinear model of the quadcopter. This reduction in performance is primarily due to the inability of linear models to fully capture the complex, nonlinear behavior inherent in real quadcopter systems. To address this limitation and enhance both robustness and trajectory tracking capabilities, the Grey Wolf Optimizer (GWO) algorithm is utilized to automatically adjust controller gains [128]. This data-driven tuning process is informed by the nonlinear system's actual response, allowing for improved adaptability and more reliable control performance [129].

A. Concept Overview

The objective of this approach is to employ the Grey Wolf Optimizer (GWO) to automatically tune the diagonal elements of the Q matrix within the LQR control framework [130]. This method replaces conventional manual tuning with a systematic optimization process that identifies the most suitable weighting values for each state variable. The Q matrix is represented as a 12-dimensional vector $Q = \mathrm{diag}(q_1, q_2, \ldots, q_{12})$, where each element corresponds to key state features. These include attitude errors $(e_\phi, e_\theta, e_\psi)$, position errors (e_x, e_y, e_z), wind disturbances (p, q, r), and linear velocity components (u, v, w). By framing the tuning process as a multi-objective optimization problem, the GWO algorithm is tasked with minimizing a cost function that balances trajectory tracking precision with control effort.

Through this adaptive tuning strategy, the LQR controller becomes more responsive to nonlinear dynamics, potential environmental disturbances, and varying mission demands. The resulting optimized Q matrix enhances the controller's ability to deliver robust, accurate, and energy-efficient performance in complex flight conditions.

A.1 Problem Formulation

The GWO algorithm searches for the optimal diagonal values of the Q matrix, where $Q = \text{diag}(q_1, q_2, \ldots, q_{12})$. where each Q_i is $(e_\phi, e_\theta, e_\psi, e_x, e_y, e_z, p, q, r, u, v, w)$ is constrained within a reasonable range to prevent instability or excessive control effort [130]. The goal is to minimize a cost function J computed from the time-domain simulation of the nonlinear system.

A.2 Cost Function Design

A multi-objective cost function is designed to penalize both tracking error and control effort:

$$J = \int_0^T \left(\|x(t) - x_{\text{ref}}(t)\|^2 + \lambda \|u(t)\|^2 \right) dt \tag{IV.5}$$

where:

- $x(t)$ is the actual state,
- $x_{\text{ref}}(t)$ is the reference trajectory,
- $u(t) = -Kx(t)$ is the control input derived using the LQR gain K computed with a candidate Q,
- λ is a weighting factor that balances tracking precision and energy efficiency.

Each candidate solution Q is passed into the LQR function to compute K, and the resulting closed-loop system is simulated using the **nonlinear 6-DOF model**.

A.3 GWO Implementation Steps

This algorithm we have already used it in the chapter III so know we will be using it by adjust the algorithm too parameters of our system.

1. **Initialization**: Generate an initial population of wolves, where each wolf is a candidate Q matrix represented by a 12-dimensional vector [111].
2. **Evaluation**: For each candidate:
   - Compute $K = \text{lqr}(A, B, Q, R)$
   - Simulate the nonlinear model using this K
   - Compute the cost J using the defined performance metric
3. **Update**: Use GWO's social hierarchy and position updating mechanism (Alpha, Beta, Delta) to guide the search towards lower-cost solutions [111].
4. **Termination**: Stop when a maximum number of iterations is reached or the improvement falls below a threshold. The best Q matrix found is then used in the final controller.

A.4 Integration with Simulation

The optimization framework is developed and executed in MATLAB, where the Grey Wolf Optimizer (GWO) algorithm is linked to a detailed Simulink model representing the nonlinear dynamics of the quadcopter [128]. The cost function is implemented as a function handle that allows each candidate Q matrix to be evaluated by running the corresponding simulation in Simulink. This setup ensures a smooth and automated interaction between the optimization process and the simulation environment. After identifying the optimal Q matrix, the resulting LQR controller is tested in separate validation scenarios [131]. These scenarios include reference trajectories and external disturbances not used during the training phase, in order to evaluate the controller's ability to generalize its performance and maintain robustness under diverse and realistic flight conditions.

B. Result and Discussion

In this simulation, the reference trajectory was carefully adjusted by slightly reducing the initial delay and extending the path beyond its final destination. These modifications enabled the quadcopter to start its motion earlier and complete the overall trajectory in a smoother manner. As a result, the controller's ability to handle transition phases could be evaluated more effectively. The trajectory used in this study is inspired by the approach proposed by Chun-Wei, who modified the flight path to follow a square-like pattern. This method led to a notable decrease in energy consumption, reducing it from **38 joules** to approximately **11.9 joules**, while maintaining accurate tracking. Building on this foundation, the present work introduces the Grey Wolf Optimizer (GWO) to further improve energy efficiency. Specifically, the GWO algorithm is applied to the optimized trajectory to adjust the diagonal values of the control weighting matrix R, defined as $R = \mathbf{diag}(r_1, r_2, r_3, r_4)$. The objective is to reduce control effort and enhance energy savings without compromising performance.

The results indicate a significant enhancement in control efficiency when compared to the baseline LQR controller, which relied on manually selected gain values. By employing the GWO-optimized R matrix, the total energy consumption was reduced from **38 units** to **9.3 units** over a simulation period of 30 seconds as shown in Figure IV.2. This outcome not only surpasses the previous benchmark of **11.9 units** but also underscores the effectiveness of the optimization process. The improvement reflects the optimizer's ability to precisely adjust the control weightings, thereby minimizing unnecessary energy expenditure while maintaining a high level of responsiveness in the system.

In terms of performance, the system achieved a mean square error (**MSE**) of **9.72**. This value is within acceptable bounds for practical applications. The marginally increased MSE can be attributed to the fixed nature of the optimized R matrix, which does not dynamically adapt to changing system conditions or external disturbances in real time. The details of all this result and each plot trajectory in Figure IV.2 are included in **ANNEX II**.

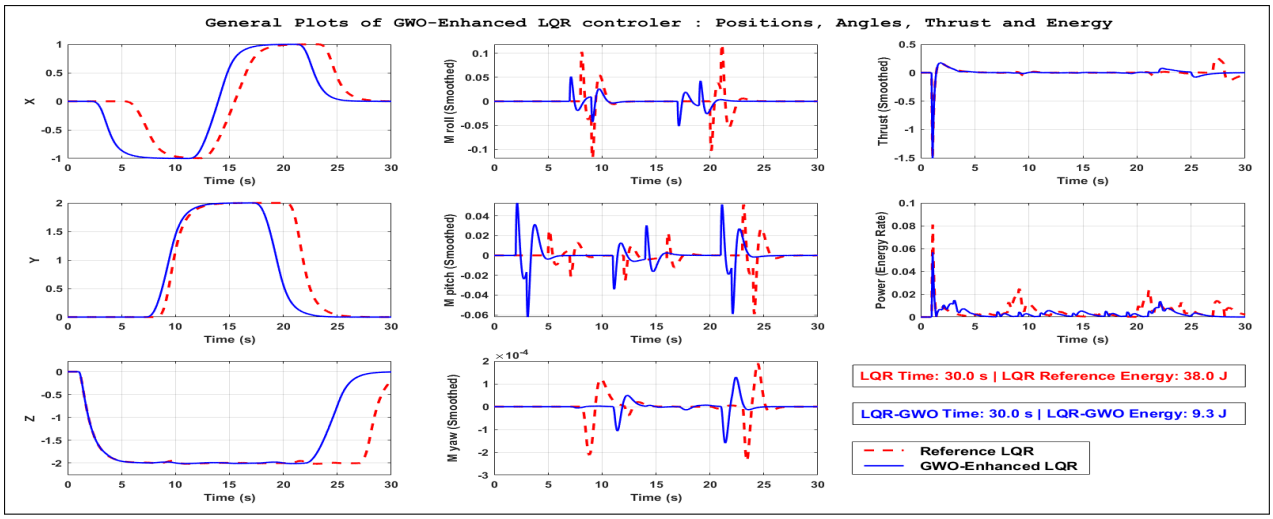


Figure IV.2: General Plots of trajectories of GWO-LQR Enhancment

IV.5.3 Adaptive LQR Using Neural Networks Adjusting Position

To improve the adaptability of the LQR controller under varying operating conditions, a Feedforward Neural Network (FNN) is integrated to dynamically adjust the diagonal elements of the Q matrix. Instead of relying on manual tuning or offline metaheuristic optimization methods such as the Grey Wolf Optimizer (GWO), the neural network is trained to estimate appropriate values based on the current flight state or environmental context [122]. This approach introduces an adaptive mechanism that enables the controller to respond more effectively to changes in system dynamics and external disturbances.

A. Concept Overview

The core objective is to train the FNN to map a set of system state features or environmental inputs to suitable values for the diagonal entries of the Q matrix, i.e., $\text{FNN}_{Q,R}$: Features $\rightarrow \{q_1, q_2, \ldots, q_{12}\}$. In doing so, the LQR controller becomes capable of adjusting its weighting strategy either in real time or across different simulation scenarios [122]. This adaptivity aims to enhance tracking accuracy and overall robustness, particularly in situations involving uncertain or dynamic operating environments.

A.1 Input Feature Selection

The input to the neural network is structured as a feature vector that captures the current state of the quadcopter system as well as the specific demands of the mission [122]. These features are carefully selected to provide the neural network with sufficient contextual information, enabling it to adapt the cost Q matrix used in the controller in a meaningful and effective manner. Key elements of the feature vector include the initial magnitudes of position and attitude errors, which reflect the deviation of the system from its desired state [122]. The mission context is also taken into account, particularly whether the quadcopter is operating in a hovering state or executing high-speed or aggressive maneuvers. Furthermore, characteristics of the reference trajectory, including its curvature and speed, significantly influence the required level of controller aggressiveness and responsiveness [122].

Based on these considerations, the feature vector can be expressed as $f = [e_\phi, e_\theta, e_\psi, e_x, e_y, e_z, p, q, r, u, v, w]$, where u, v, w represent the trajectory speeds corresponding to the **errors in the linear velocities** of the drone in the **body frame**. Specifically, u is the velocity along the body X-axis, v is the velocity along the body Y-axis, and w is the velocity along the body Z-axis. Similarly, p, q, r denote the angular velocities about the body X-, Y-, and Z-axes, respectively, representing the rotational motion errors of the drone.

A.2 Output Definition

The output of the neural network is structured as a vector containing the diagonal elements of the Q matrix, which is subsequently used by the LQR controller to compute the optimal gain matrix. Specifically, the network generates 12 elements corresponding to the weights assigned to individual state variables, denoted as $y = [q_1, q_2, \ldots, q_{12}]$. Each element q_i reflects the relative importance of a particular state component, allowing the controller to make informed trade-offs between tracking accuracy and control effort based on the current context [122]. In the present implementation, the problem is simplified by focusing only on the most influential elements of the Q matrix. Specifically, the neural network is tasked with predicting values for q_1 through q_6, which correspond to attitude errors (roll, pitch, and yaw) and position errors along the X, Y, and Z axes. The remaining elements of the matrix, which relate to angular velocities and trajectory speed (error velocities), are held constant throughout the simulations. This reduction in output dimensionality eases the complexity of the training process while still allowing the neural network to exert meaningful control over the quadcopter's most critical dynamic behaviors.

A.3 Network Architecture

A typical architecture will have these parameters:

- `neurons1 = 10;` % Number of neurons in each hidden layer
- `epochs1 = 300;` % Maximum number of training epochs
- `net1 = feedforwardnet(neurons1 neurons1);` % Use type FeedForward
- `net.trainFcn = 'trainlm';` % Levenberg-Marquardt backpropagation
- `net1.trainParam.epochs = epochs1;`
- `net1.trainParam.goal = 1e-3;`
- `net1.trainParam.max.fail = 10;`
- `numSamples1 = 500;`
- `net1.divideParam.trainRatio = 0.7;`
- `net1.divideParam.valRatio = 0.15;`
- `net1.divideParam.testRatio = 0.15;`
- `inputs1 = zeros(12, numSamples);` % 12 features: [attitude, position, angular velocities, error velocities]
- `targets1 = zeros(6, numSamples);` % Only Q1-Q6 to be adjusted

B. Results and Discussion

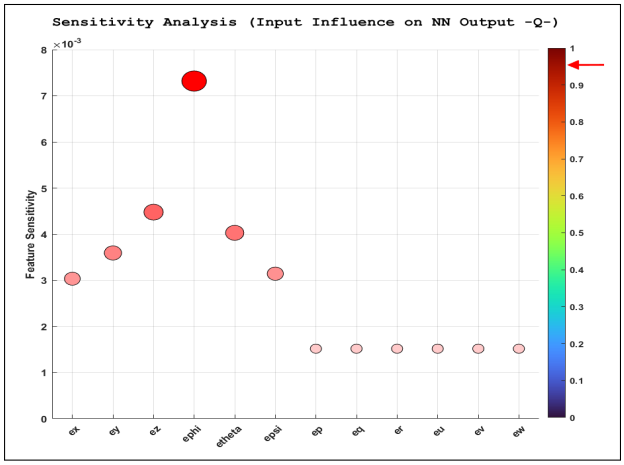
The subsequent sections detail the results obtained from incorporating a neural network into the Linear Quadratic Regulator (LQR) control loop of the quadcopter, with particular attention to the modification of the Q matrix. The discussion highlights the neural network's performance throughout the training process and examines the effects of its integration on the system's control efficiency and energy usage.

B.1 Neural Network Training Performance

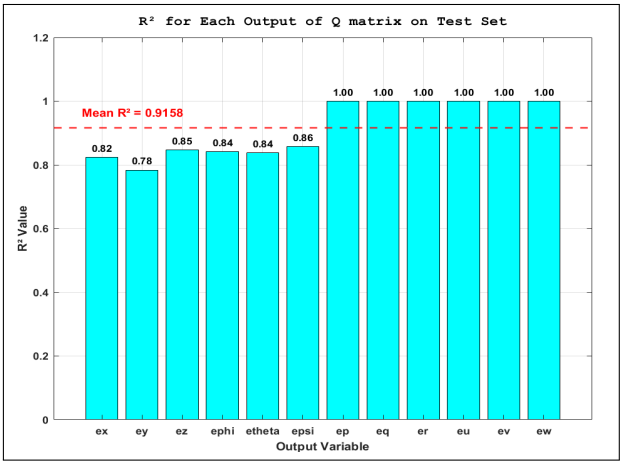
The neural network demonstrated stable and consistent convergence throughout the training process. In the primary model, training was conducted over 300 iterations, ultimately reaching a final mean squared error (**MSE**) **of 8.02**. During this phase, validation performance remained steady, as evidenced by 14 validation checks, which indicate robust generalization capabilities without any noticeable signs of overfitting. **The training gradient** converged to a value **of 1.15**, and the total training time was approximately **4.1 seconds**. These results suggest a relatively smooth optimization trajectory across the loss landscape.

The coefficient of determination (R^2) values for the key training targets were as follows: $R^2_{e_\phi} = 0.82$, $R^2_{e_\theta} = 0.78$, $R^2_{e_\psi} = 0.85$, $R^2_{e_x} = 0.84$, $R^2_{e_y} = 0.84$, and $R^2_{e_z} = 0.86$, others R^2 are equal to 1 because there is no adjustment in the other 6 parameters on the Q Matrix. The **mean** R^2 across these four dominant features was calculated to be **0.9158**, indicating a high degree of fit between the predicted and true Q-matrix components, also Sensitivity Analysis show that the most significant parameter effect is the error of ϕ then error on Z (Figure IV.3a & Figure IV.3b).

Collectively, Q matrices affirm that the FNN effectively learned the underlying mapping between the quadcopter's state conditions and the corresponding optimal values for the Q matrix tuning parameters. The convergence trends and error indicators support the suitability for integration into the control loop, enabling real-time adaptation during flight.



(a) Sensitivity Analyses of Adjustique Q Matrix

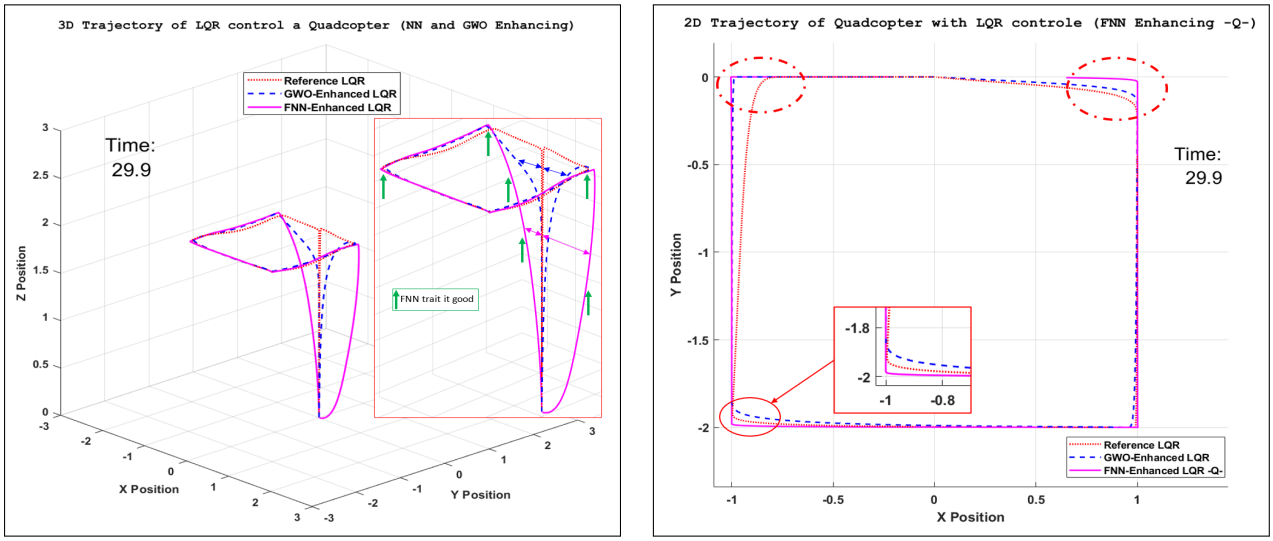


(b) R-Square of Globale and for each parameter

Figure IV.3: The Analyses performnance of FNN-LQ

B.2 Generalization and Output Trends

The trained neural network exhibited reliable generalization across a wide range of state variables (Figure IV.4a & IV.4b). It adapted the cost function parameters in response to varying flight conditions, demonstrating sensitivity to the system's dynamic behavior. Notably, larger deviations in attitude or position led to an increase in the respective q_i values, thereby enhancing the corrective action of the controller in those dimensions. Although the network did not directly produce weight values for wind disturbances or velocity terms, their inclusion as input variables allowed the model to infer appropriate adjustments. This enabled the network to scale the cost function dynamically, resulting in context-aware parameter tuning. Such adaptability contributed to maintaining stable performance even under the influence of external disturbances or sudden changes in system behavior.



(a) 3D vision of Trajectories (b) 2D Vision Showing the Outline of a Square

Figure IV.4: Trajectory of Quadcopter FNN-LQ controled (3D and 2D)

B.3 Controller Performance and Energy Efficiency

Integrating the trained neural network into the LQR control structure produced notable enhancements in both flight stability and energy efficiency. In a series of dynamic simulations, the adaptive LQR controller consistently outperformed the conventional fixed-parameter version in several key performance indicators. Most significantly, total control energy consumption was reduced **from 38 units to 5.9 units**, reflecting a substantial improvement in energy utilization, as demonstrated in Figure IV.5. This improvement also underscores the system's ability to effectively manage trajectory tracking while performing surface inspection tasks, such as those involving square or rectangular areas. Also, the adaptive controller achieved additional performance benefits, including:

- Improved accuracy in trajectory tracking.
- Smoother transitions between different flight modes.
- Decreased oscillations and overshoot in response to disturbances.

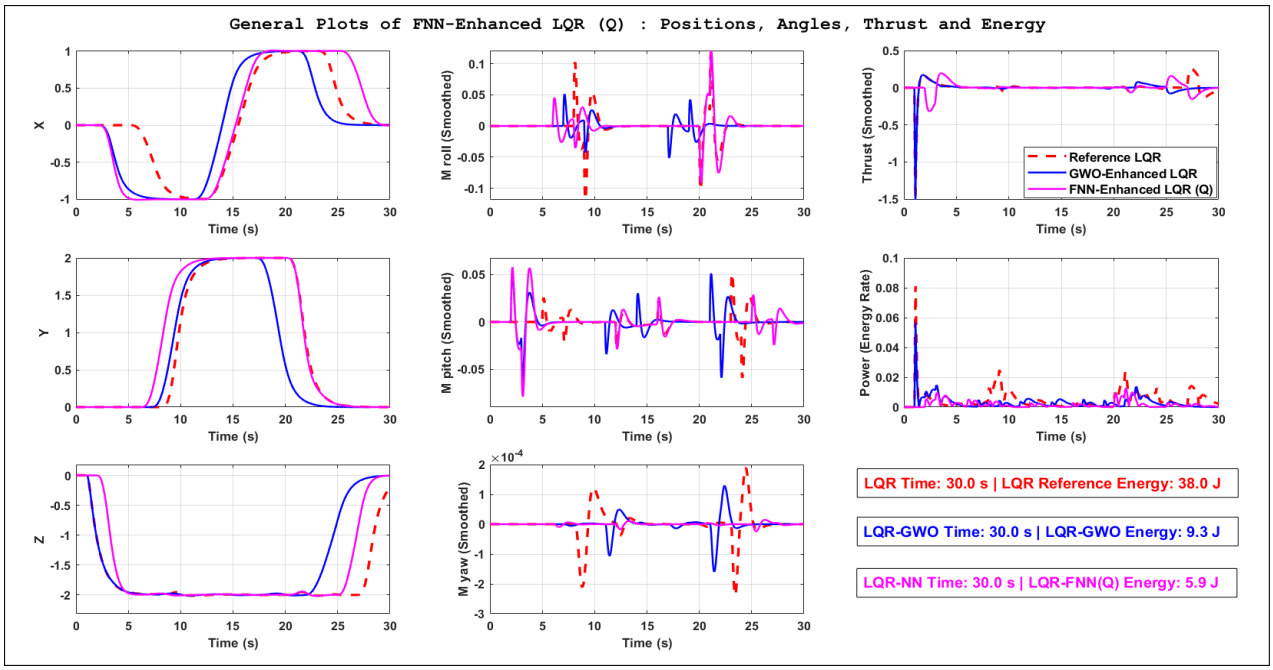


Figure IV.5: Globale Trajectories Plot of FNN-LQ Enhancement

These findings demonstrate the practical value of employing a neural network for online tuning of control parameters, particularly in scenarios that demand high responsiveness and adaptability under uncertain or rapidly changing flight conditions.

IV.5.4 Adaptive LQR Using Neural Networks for Adjusting Torques

To further increase the adaptability of the Linear Quadratic Regulator (LQR) controller, a second Feedforward Neural Network (FNN) is introduced. This network is designed to dynamically adjust the elements of the R matrix, which directly influence the control effort associated with the quadcopter's torque inputs. Its integration builds upon the functionality of the previously implemented neural network responsible for tuning the Q matrix, together forming a more robust and energy-efficient control architecture.

While the initial neural network focuses on optimizing the penalties related to position and orientation through modifications to the Q matrix, the second network, referred to as **FNN_R**, is specifically tasked with adapting the torque-related components of the R matrix. It generates weight values corresponding to the control inputs, including thrust effort, roll torque (τ_x), pitch torque (τ_y), and yaw torque (τ_z).

A. Concept Overview

The primary objective of this second neural network is to provide real-time, context-aware adjustment of torque penalties within the LQR framework. By tuning the R matrix according to the quadcopter's current state and operational objectives, the controller is better equipped to regulate actuator demands. This dynamic adjustment not only supports the preservation of system stability and responsiveness but also contributes to improved energy efficiency by minimizing unnecessary control effort.

In mathematical terms, the behavior of the second neural network can be described as follows: FNN_R : Features $\rightarrow \{r_1, r_2, r_3, r_4\}$, where: r_1: thrust effort weight, r_2: roll torque weight (τ_x), r_3: pitch torque weight (τ_y) and r_4: yaw torque weight (τ_z).

A.1 Input Feature Selection

To maintain consistency and ensure a shared contextual understanding between both neural networks, the same input feature structure used in FNN_q is employed for FNN_R. The input vector encompasses critical state information, including drone attitude, positional data, wind disturbances, and velocity errors. This comprehensive set of inputs equips the network with the necessary context to adapt the control effort weights effectively. By capturing both environmental and dynamic system factors, the model supports real-time, intelligent tuning of the R matrix, enhancing overall controller responsiveness and energy efficiency.

A.2 Output Definition

The neural network outputs four non-negative values that form the diagonal of the R matrix: $R = \text{diag}(r_1, r_2, r_3, r_4)$. These weights directly influence the actuator demand for thrust and torques in the LQR cost function, allowing adaptive balancing between performance and energy consumption.

A.3 Network Architecture

The architecture used for FNN_R is lighter compared to the Q-adapting network, reflecting its narrower output scope. Training was conducted using the same Levenberg-Marquardt algorithm. The key training parameters were :

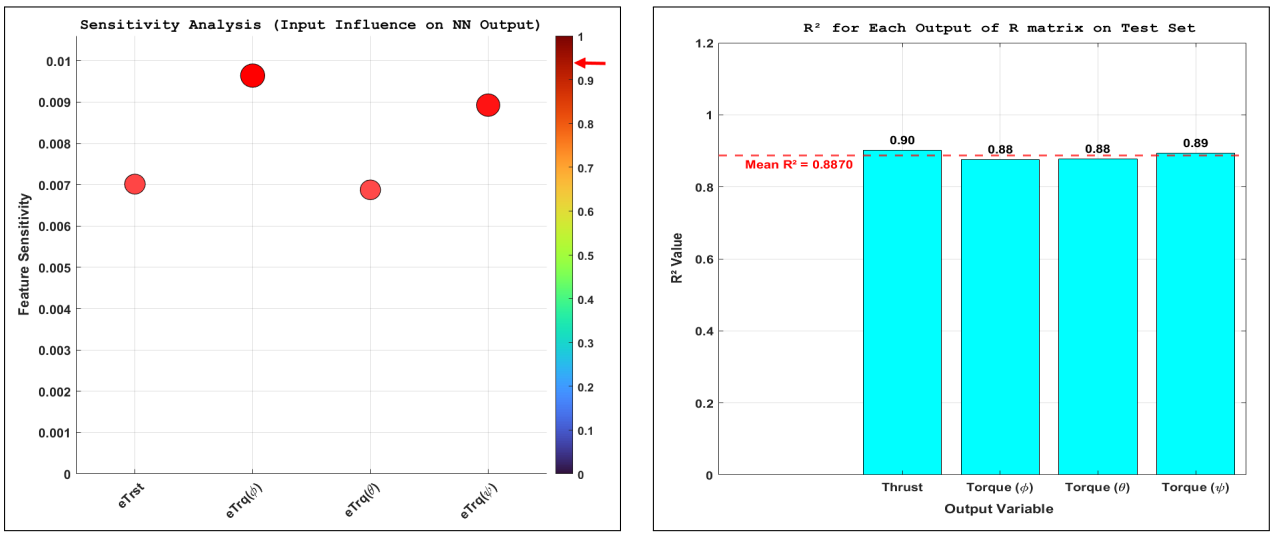
- `neurons2 = 4;` % Number of neurons in each hidden layer
- `epochs2 = 200;` % Maximum number of training epochs
- `net2 = feedforwardnet(neurons1 neurons1);` % Use type FeedForward
- `net2.trainFcn = 'trainlm';` % Levenberg-Marquardt backpropagation
- `net2.trainParam.epochs = epochs1;`
- `net2.trainParam.goal = 1e-7;`
- `net2.trainParam.max.fail = 10;`
- `numSamples2 = 600;`
- `net2.divideParam.trainRatio = 0.70;`
- `net2.divideParam.valRatio = 0.10;`
- `net2.divideParam.testRatio = 0.20;`
- `inputs = zeros(4, numSamples2);` % 4 features: [Thrust, $Torque_\phi$, $Torque_\theta$, $Torque_\psi$]
- `targets = zeros(4, numSamples2);`

B. Results and Discussion

The following sections present the outcomes of incorporating the second Feedforward Neural Network FNN_R into the control architecture. Particular attention is directed toward the network's training characteristics and its influence on both energy efficiency and overall control performance. The analysis aims to demonstrate how the adaptive tuning of the control-related matrix by FNN_R contributes to more effective thrust and torque regulation, ultimately reducing energy expenditure while maintaining or improving trajectory accuracy.

B.1 Neural Network Training Performance

The second, more compact Feedforward Neural Network (FNN_R) completed its training over **100 epochs**, achieving a final mean squared error (**MSE**) of **3.48**. Notably, the training process exhibited no validation failures, indicating consistent generalization across the training dataset. The total training time was approximately **5.2 seconds**. The final gradient value reached **0.462**, which was lower than that observed in the Q-tuning network, signifying stable and effective convergence. The training curve, presented in Annex II, displays a steady decline in the loss function over the training epochs, reinforcing the observation of stable learning behavior. Despite its relatively simple architecture, the network demonstrated sufficient capacity to successfully learn the mapping required for adjusting the four torque-related control weights.



(a) Sensitivity Analyses FNN-R Matrix (b) R-Square of Globale, each parameter

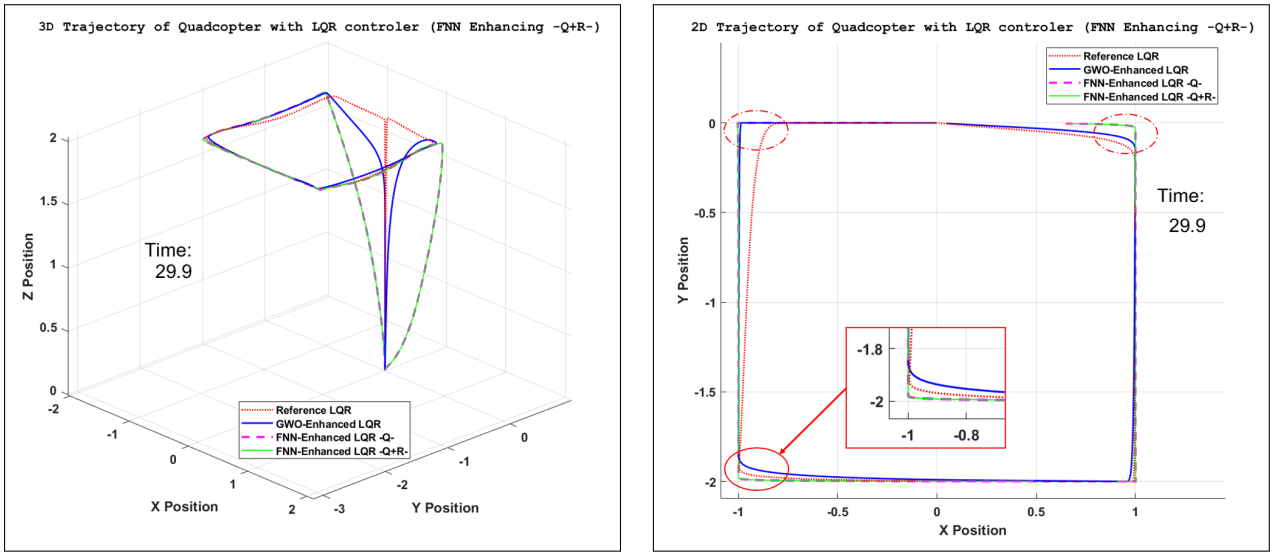
Figure IV.6: The Analyses performnance of FNN-LR

The mean (R^2) associated with the predicted outputs of the R matrix on the test dataset were determined as follows: $R^2_{\text{Thrust}} = 0.90$, $R^2_{\text{Torque}_\phi} = 0.88$, $R^2_{\text{Torque}_\theta} = 0.88$, and $R^2_{\text{Torque}_\psi} = 0.89$. These values demonstrate a strong alignment between the model's outputs and the corresponding target variables, suggesting that the regression model is capable of accurately estimating the dynamic behaviors represented by thrust and the three rotational torque components.

Moreover, the **average** R^2 value across these four critical outputs was calculated to be **0.8870**, which reflects a high level of overall prediction reliability and stability. These findings support the model's effectiveness in reproducing the essential dynamics of the system (Figure IV.6a & IV.6b).

B.2 Generalization and Output Trends

Post-training analysis revealed that FNN_R effectively adjusted the torque weighting values in response to varying flight conditions. Specifically, in situations involving significant angular deviation or rapid maneuvering, the network slightly reduced the torque weights to enable faster and more agile control responses. Conversely, under stable or hovering conditions, it increased the torque weights, thereby limiting control effort and conserving energy. The incorporation of this work, with the FNN adjusting the Q matrix, enabled the network to adjust its output judiciously. This adaptive behavior prevented excessive actuator commands during flight scenarios, enhancing overall energy efficiency. Also, this feature didn't make a valuable change in trajectory tracking (Figure IV.7).



(a) 3D vision of Trajectories (b) 2D Vision Showing the Outline of a Square

Figure IV.7: Trajectory of Quadcopter FNN-LQR controled (3D and 2D)

B.3 Controller Performance and Energy Efficiency

Integrating both FNN_Q and FNN_R into the control framework produced the most energy-efficient and responsive strategy among all tested configurations. Simulations showed a substantial reduction in total control energy consumption, decreasing the **11.9 units** (Chun-Wei results) to only **3.8 units** with the combined approach. This represents a marked improvement over both the static LQR baseline and the partially adaptive controller.
In addition to enhanced energy efficiency (Figure IV.8 & IV.9), the dual-network controller maintained high-performance standards in several key areas:

- Improved tracking accuracy relative to reference trajectories
- Rapid and smooth transitions between dynamic flight modes

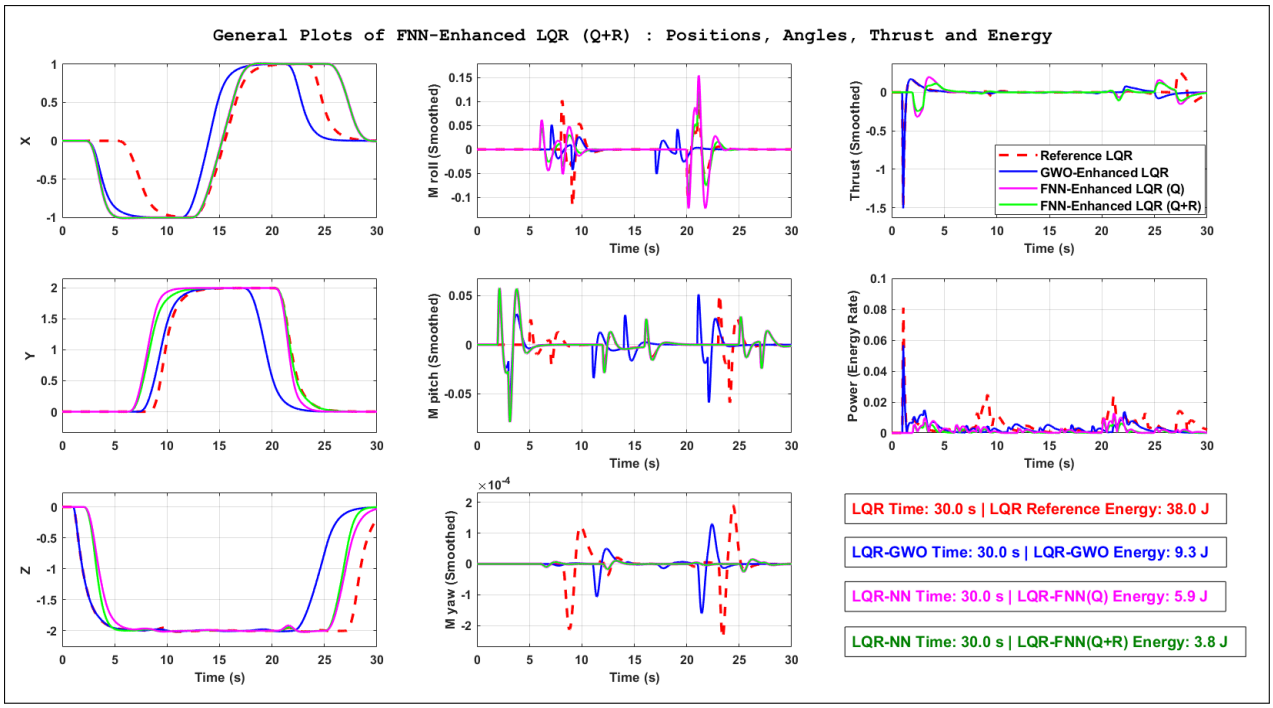


Figure IV.8: General Results of FNN-Enhanced LQR (Q, R Matrices)

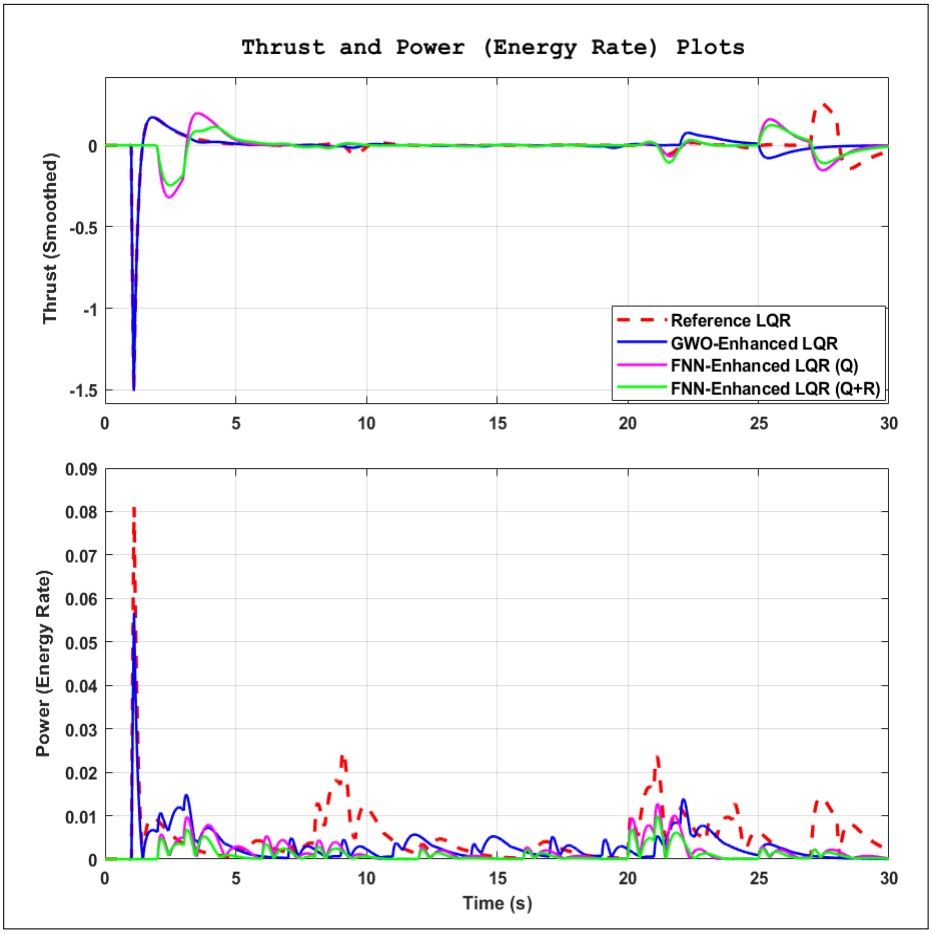


Figure IV.9: Thrust and Energy Rate Results.

These results underscore the value of using neural networks to adaptively tune both Q and R matrices in real time. This hybrid approach offers a more intelligent, robust control (Q Matrix) solution and energy awareness (R Matrix) for quadcopter operations.

The key performance indicators are summarized in the following table:

Table IV.2: Comparative Analysis of Three Methods for Quadcopter Control Tuning

Control Strategy	**Tuning Method**	**Energy (Joul)**	**MSE**	**Time**	**Adaptability**	**Remarks**
GWO-Optimized LQR (GWO-R)	Grey Wolf Optimizer (R matrix)	9.3	9.72	3.8	Low	Effective in reducing energy; limited flexibility for varying flight conditions and controlling trajectory.
Adaptive LQR with First Neural Network (FNN-Q)	Neural Network for Q matrix	5.9	8.02	4.1	Moderate	Dynamically adjusts state penalties; improved energy use and trajectory tracking.
Fully Adaptive LQR (FNN-Q + FNN-R)	Neural Networks for Q and R matrices	3.8	3.48	5.2	High	Best results in terms of energy, adaptability, and smooth control actions.

IV.6 Conclusion

This chapter has presented a comprehensive investigation into enhancing the energy efficiency and trajectory tracking performance of a quadcopter system using the LQR framework with intelligent tuning techniques. Starting from the foundational principles of LQR control applied to a linearized 6DOF X4 model, demonstrating the advantages of classical optimal control strategies in stabilizing flight dynamics and achieving balanced control efforts.

To address the inherent limitations of fixed-parameter controllers in complex, nonlinear, and time-varying environments, two key extensions were explored. The first enhancement involved optimization of the LQR weighting matrices using the Grey Wolf Optimizer (GWO), which provided improved performance by refining the control gains of the R matrix. This method successfully reduced energy consumption while maintaining acceptable tracking accuracy. The second and more advanced approach integrated neural network-based adaptation mechanisms to dynamically tune both the Q and R matrices. By employing two dedicated Feedforward Neural Networks (FNNs), the control system gained the ability to adjust in real time to varying flight conditions and operational contexts. This dual adaptation led to the most favorable results across all evaluated metrics, significantly improving energy efficiency and tracking precision, while also offering increased robustness and responsiveness.

Simulation outcomes confirmed that the proposed intelligent control schemes not only reduced energy expenditure from 38 units to as low as 3.8 units, but also enhanced flight stability and control smoothness. These findings underscore the effectiveness of combining classical control methods with artificial intelligence for advanced unmanned aerial system applications. The proposed hybrid control architecture offers a promising foundation for future developments in adaptive and energy-aware UAV control systems.

GENERAL CONCLUSION

This thesis has contributed to the advancement of intelligent control strategies for quadcopter UAVs by addressing some of the most pressing challenges in autonomous aerial navigation - namely, accurate trajectory tracking, robustness to dynamic disturbances, obstacle interaction, and efficient energy consumption. Through the development and implementation of advanced nonlinear and hybrid control approaches, this work has demonstrated that high-performance flight control can be achieved without compromising adaptability or stability.

Central to this research was the design of an optimized Backstepping control architecture, which was enhanced using metaheuristic algorithms such as the Grey Wolf Optimizer (GWO), Garra Rufa Optimizer (GRO), and the Pelican Optimization Algorithm (POA). These algorithms enabled automatic tuning of control gains, resulting in significant improvements in tracking precision. The optimized controllers performed remarkably well across a variety of simulated scenarios, including spiral and zigzag trajectories, complex path transitions, and wind disturbances. Notably, the system exhibited a high degree of stability and responsiveness, maintaining accurate trajectory adherence even in the presence of external perturbations or environmental variability.

In addition to precision tracking, the research also addressed the need for intelligent energy management. By dynamically adapting thrust in response to trajectory changes and optimizing flight paths in real time, the proposed controllers succeeded in reducing unnecessary energy expenditure. This capability is particularly relevant for extending the operational time and range of lightweight drones, which are inherently limited by battery capacity. The ability to maintain mission objectives while minimizing power usage represents a significant step toward practical deployment in real-world conditions.

Another noteworthy aspect of this work is its ability to maintain control performance in obstacle-laden environments. While obstacle avoidance was not the central focus, the robustness of the control architecture ensured that trajectory deviations due to sudden disturbances or redirections were minimal and self-correcting. The system consistently realigned with the intended path while minimizing cumulative error, thereby ensuring mission continuity even under non-ideal flight conditions.

The culmination of these efforts is a control system that is not only theoretically robust but also practically viable. By integrating classical techniques with bio-inspired optimization and adaptive learning elements, this thesis has shown that it is possible to construct a UAV control framework that balances mathematical rigor with real-time adaptability. The results demonstrate that the primary objectives, which achieving accurate trajectory tracking, enabling intelligent energy use, and ensuring system robustness.

Looking forward, this work lays the foundation for several future directions, including hardware implementation, real-world testing under varying environmental conditions, and extension to cooperative UAV systems. Furthermore, the integration of these controllers with real-time obstacle detection and avoidance mechanisms, possibly through onboard vision or LiDAR systems, would further enhance the autonomy and resilience of the platform.

In summary, the results obtained confirm the thesis hypothesis: intelligent and optimized control strategies can significantly improve the accuracy, efficiency, and adaptability of quadcopter UAVs. This research therefore not only advances the state of the art in UAV control but also contributes a viable step toward the realization of fully autonomous, energy-aware, and mission-capable aerial systems.

BIBLIOGRAPHY (REFERENCES)

[1] Hassani, H., Mansouri, A. & Ahaitouf, A. “Performance Evaluation of Control Strategies for Autonomous Quadrotors: A Review”. *Complexity* 2024.1 (2024), p. 8820378. https://doi.org/10.1155/2024/8820378.

[2] Mendoza-Mendoza, J., Gonzalez-Villela, V., Aguilar-Ibañez, C. & Fonseca-Ruiz, L. “Drone Design Concepts”. In: *Drones to Go: A Crash Course for Scientists and Makers*. 2021, pp. 1–38. https://doi.org/10.1007/978-1-4842-6788-2_1.

[3] Omar, H., Akram, R., Mukras, S. & Mahvouz, A. “Recent advances and challenges in controlling quadrotors with suspended loads”. *Alexandria Engineering Journal* 63 (Jan. 2023), pp. 253–270. https://doi.org/10.1016/j.aej.2022.08.001.

[4] Eid, M., Khedr, E., Mohamed, A., Abofarag, H. & Rizk, J. “Control of A Quadrotor For Medical Service”. *Journal of Engineering Research* 8.5 (2024). https://doi.org/10.70259/engJER.2024.851815.

[5] Kangunde, V., Jamisola Jr, R. & Theophilus, E. “A review on drones controlled in real-time”. *International Journal of Dynamics and Control* 9.4 (Dec. 2021), pp. 1832–1846. https://doi.org/10.1007/s40435-020-00737-5.

[6] Mueller, M., Lee, S. & D’Andrea, R. “Design and Control of Drones”. *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (May 2022), pp. 161–177. https://doi.org/10.1146/annurev-control-042920-012045.

[7] Blanchard, T. “Caractérisation de drones en vue de leur localisation et de leur suivi à partir d’une antenne de microphones”. NNT : 2019LEMA1042. Thèse de doctorat. Le Mans Université, 2019. https://theses.hal.science/tel-02906392/.

[8] (USA), R. F. *AQM-34 Ryan Firebee (USA)*. Originally appeared in the 1960s. 1984. https://www.pbs.org/wgbh/nova/spiesfly/uavs_09.html.

[9] Eskandaripour, H. & Boldsaikhan, E. “Last-Mile Drone Delivery: Past, Present, and Future”. *Drones* 7.2 (Jan. 2023), p. 77. https://doi.org/10.3390/drones7020077.

[10] Tazibt, C., Achir, N. & Djamah, T. “Online Drone-Based Data Gathering Strategies for Ground Sensor Networks”. *International Journal of Sensor Networks* 38.3 (2022), pp. 177–190. https://doi.org/10.1504/IJSNET.2022.121702.

[11] Caballero-Martin, D., Lopez-Guede, J., Estevez, J. & Graña, M. “Artificial Intelligence Applied to Drone Control: A State of the Art”. *Drones* 8.7 (July 2024), p. 296. https://doi.org/10.3390/drones8070296.

[12] Yan, B., Wei, Y., Liu, S., Huang, W., Feng, R. & Chen, X. “A Review of Current Studies on the Unmanned Aerial Vehicle-Based Moving Target Tracking Methods”. *Defence Technology* (2025). ISSN 2214-9147. https://doi.org/10.1016/j.dt.2025.01.013.

[13] Seidu, I., Olowu, B. & Olowu, S. "Advancements in Quadcopter Development through Additive Manufacturing: A Comprehensive Review". *International Journal of Scientific Research in Science, Engineering and Technology* 11.4 (July 2024), pp. 92–124. https://doi.org/10.32628/ijsrset24114109.

[14] Zaid, M., Kadir, M., Shayea, I. & Mansor, Z. "Machine Learning-Based Approaches for Handover Decision of Cellular-Connected Drones in Future Networks: A Comprehensive Review". *Engineering Science and Technology, an International Journal* 55 (July 2024), p. 101732. https://doi.org/10.1016/j.jestch.2024.101732.

[15] Koopman, M., Milliet, Q. & Champod, C. "A Review of Predictive Modelling and Drone Remote Sensing Technologies as a Tool for Detecting Clandestine Burials". *Forensic Science International* (Jan. 2025), p. 112375. https://doi.org/10.1016/j.forsciint.2025.112375.

[16] Wang, R. & Shen, J. "Disturbance Observer and Adaptive Control for Disturbance Rejection of Quadrotor: A Survey". *Actuators* 13.6 (June 2024), p. 217. https://doi.org/10.3390/act13060217.

[17] Khalid, A., Mushtaq, Z., Arif, S., Zeb, K., Khan, M. & Bakshi, S. "Control Schemes for Quadrotor UAV: Taxonomy and Survey". *ACM Computing Surveys* 56.5 (Nov. 2023), pp. 1–32. https://doi.org/10.1145/3617652.

[18] Wei, X., Ye, D., Zhang, Z. & Hu, S. "A Review of Quadrotor Control Methods". *Advances in Engineering Technology Research* 7.1 (Sept. 2023), p. 495. https://doi.org/10.56028/aetr.7.1.495.2023.

[19] Villa, D., Brandão, A. & Sarcinelli-Filho, M. "Cooperative Load Transportation with Quadrotors Using Adaptive RISE Control". *Journal of Intelligent and Robotic Systems* 110.4 (Sept. 2024), p. 138. https://doi.org/10.1007/s10846-024-02174-4.

[20] Zuo, Y., Liu, Y. & Ahmad, A. "Autonomous Blimp Control via H-infinity Robust Deep Residual Reinforcement Learning". In: *IEEE 19th International Conference on Automation Science and Engineering*. Mar. 2023, pp. 1–8. https://doi.org/10.1109/CASE56687.2023.10260561.

[21] Emami, S., Castaldi, P. & Banazadeh, A. "Neural Network-Based Flight Control Systems: Present and Future". *Annual Reviews in Control* 53 (Jan. 2022), pp. 97–137. https://doi.org/10.1016/j.arcontrol.2022.04.006.

[22] Langston, D. "A Comprehensive Review of Trajectory Tracking Control Technologies for Rotary-Wing UAVs: Challenges and Prospects". *Journal of Computer Science and Software Applications* 4.8 (Dec. 2024), pp. 1–5. https://www.mfacademia.org/index.php/jcssa/article/view/174.

[23] Meddahi, Y., Meguenni, K. & Aoued, H. "The Nonlinear Computed Torque Control of a Quadrotor". *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)* 20.3 (Dec. 2020), pp. 1221–1229. http://doi.org/10.11591/ijeecs.v20.i3.pp1221-1229.

[24] Wu, D. "Predictor-Based Neural Network Control for Unmanned Aerial Vehicles with Input Quantization: Design and Application". *Artificial Intelligence Review* 58.2 (Dec. 2024), p. 47. https://doi.org/10.1007/s10462-024-11054-0.

[25] C. Vangasse, A. da, Freitas, J., Raffo, G. & Pimenta, L. "Safe Navigation on Path-Following Tasks: A Study of MPC-Based Collision Avoidance Schemes in Distributed Robot Systems". *Journal of Intelligent and Robotic Systems* 110.4 (Dec. 2024), pp. 1–7. https://doi.org/10.1007/s10846-024-02202-3.

[26] Garcia, P., Lozano, R., Garcia, P. & Albertos, P. "Nonlinear Control of a Small Four-Rotor Rotorcraft". In: *Nonlinear and Adaptive Control: Theory and Algorithms for the User*. 2005, pp. 147–177. https://doi.org/10.1142/9781860947346_0006.

[27] Kim, J., Kang, M. & Park, S. "Accurate Modeling and Robust Hovering Control for a Quad-Rotor VTOL Aircraft". In: *Selected Papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA*. Springer Netherlands, 2009, pp. 9–26. https://doi.org/10.1007/978-90-481-8764-5_2.

[28] Santos, M., Lopez, V. & Morata, F. "Intelligent Fuzzy Controller of a Quadrotor". In: *2010 IEEE International Conference on Intelligent Systems and Knowledge Engineering*. Nov. 2010, pp. 141–146. https://doi.org/10.1109/ISKE.2010.5680812.

[29] Alexis, K., Nikolakopoulos, G. & Tzes, A. "Model Predictive Quadrotor Control: Attitude, Altitude and Position Experimental Studies". *IET Control Theory and Applications* 6.12 (Aug. 2012), pp. 1812–1827. https://doi.org/10.1049/iet-cta.2011.0348.

[30] Argentim, L., Rezende, W., Santos, P. & Aguiar, R. "PID, LQR and LQR-PID on a Quadcopter Platform". In: *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*. May 2013, pp. 1–6. https://doi.org/10.1109/ICIEV.2013.6572698.

[31] Mofid, O., Mobayen, S. & Fekih, A. "Adaptive Integral-Type Terminal Sliding Mode Control for Unmanned Aerial Vehicle under Model Uncertainties and External Disturbances". *IEEE Access* 9 (Apr. 2021), pp. 53255–53265. https://doi.org/10.1109/ACCESS.2021.3070400.

[32] Abro, G., Zulkifli, S., Asirvadam, V. & Ali, Z. "Model-Free-Based Single-Dimension Fuzzy SMC Design for Underactuated Quadrotor UAV". *Actuators* 10.8 (Aug. 2021), p. 191. https://doi.org/10.3390/act10080191.

[33] Razmi, H. “Adaptive Neural Network Based Sliding Mode Altitude Control for a Quadrotor UAV”. *Journal of Central South University* 25.11 (Dec. 2018), pp. 2654–2663. https://doi.org/10.1007/s11771-018-3943-0.

[34] Hanna, Y., Khater, A., El-Nagar, A. & El-Bardini, M. “Polynomial Recurrent Neural Network-Based Adaptive PID Controller with Stable Learning Algorithm”. *Neural Processing Letters* 55.3 (June 2023), pp. 2885–2910. https://doi.org/10.1007/s11063-022-10989-1.

[35] Castillo-Zamora, J. J., Camarillo-Gomez, K. A., Perez-Soto, G. I. & Rodriguez-Resendiz, J. “Comparison of PD, PID and sliding-mode position controllers for V–tail quadcopter stability”. *IEEE Access* 6 (2018), pp. 38086–38096. https://doi.org/10.1109/ACCESS.2018.2851223.

[36] Guettal, L. “Modeling and Robust Control of Flying Robots Using Intelligent Approaches”. In French. PhD thesis. Université Mohamed Khider Biskra, 2023. http://thesis.univ-biskra.dz/6233/.

[37] Satla, Z. “Contribution à la modélisation et à la commande d’un drone miniature”. In French. PhD thesis. Université of Sidi Bel Abbes-Djillali Liabes, 2019. https://bucket.theses-algerie.com/files/repositories-dz/2519265048435545.pdf.

[38] Chiter, O., Tali, N. & Bouadi, H. “Contribution to quadrotor modeling and attitude stabilization based on an adaptive flight control strategy”. MA thesis. National High School of Technology, Algeria, 2023. http://dspace.ensta.edu.dz/jspui/handle/123456789/187.

[39] Aiche, M. & Aiche, A. “Réalisation et commande d’une plateforme volante à quatre rotors”. In French. MA thesis. University of M’sila, 2021. https://repository.univ-msila.dz/items/2cc62745-3d1e-4fb8-9087-aab84f2bfa6f.

[40] Kuantama, E., Vesselenyi, T., Dzitac, S. & Tarca, R. “PID and Fuzzy-PID control model for quadcopter attitude with disturbance parameter”. *International Journal of Computers Communications and Control* 12.4 (2017), pp. 519–532. https://doi.org/10.15837/ijccc.2017.4.2962.

[41] Iriarte, I., Gorostiza, J., Iglesias, I., Lasa, J., Calvo-Soraluze, H. & Sierra, B. “An overactuated aerial robot based on cooperative quadrotors attached through passive universal joints: Modeling, control and 6-DoF trajectory tracking”. *Robotics and Autonomous Systems* 180 (2024), p. 104761. https://doi.org/10.1016/j.robot.2024.104761.

[42] Sadi, M. A., Jamali, A., Abang Kamaruddin, A. M. bin & Jun, V. Y. “Cascade model predictive control for enhancing UAV quadcopter stability and energy efficiency in wind turbulent mangrove forest environment”. *e-Prime - Advances in Electrical Engineering, Electronics and Energy* 10 (2024), p. 100836. https://doi.org/10.1016/j.prime.2024.100836.

[43] Glida, H. E., Abdou, L., Chelihi, A., Sentouh, C. & Hasseni, S. E. "Optimal model-free backstepping control for a quadrotor helicopter". *Nonlinear Dynamics* 100 (2020), pp. 3449–3468. https://doi.org/10.1007/s11071-020-05671-x.

[44] Feng, L. & Katupitiya, J. "Vector field based control of quadrotor UAVs for wildfire boundary monitoring". *Journal of Intelligent and Robotic Systems* 106.1 (2022), p. 27. https://doi.org/10.1007/s10846-022-01731-z.

[45] Candan, F., Beke, A., Mahfouf, M. & Mihaylova, L. "A real-time fuzzy interacting multiple-model velocity obstacle avoidance approach for unmanned aerial vehicles". *Journal of Intelligent and Robotic Systems* 110.2 (2024), p. 61. https://doi.org/10.1007/s10846-024-02075-6.

[46] Madeiras, J., Cardeira, C. & Oliveira, P. "Position and Attitude Tracking Controllers Using Lyapunov Transformations for Quadrotors". *Journal of Intelligent and Robotic Systems* 110.1 (2024), p. 9. https://doi.org/10.1007/s10846-023-02016-9.

[47] Nikhilraj, A., Simha, H. & Priyadarshan, H. "Modeling and Control of port dynamics of a tilt-rotor quadcopter". In: *IFAC-PapersOnLine*. Vol. 55. 1. 2022, pp. 746–751. https://doi.org/10.1016/j.ifacol.2022.04.122.

[48] Bennaceur, S. & Azouz, N. "Modelling and control of a quadrotor with flexible arms". *Alexandria Engineering Journal* 65 (2023), pp. 209–231. https://doi.org/10.1016/j.aej.2022.10.027.

[49] Amiri, M. & Hosseinzadeh, M. "Closed-loop model identification and MPC-based navigation of quadcopters: A case study of parrot bebop 2". In: *IFAC-PapersOnLine*. Vol. 58. 28. 2024, pp. 330–335. https://doi.org/10.48550/arXiv.2404.07267.

[50] Janszen, J., Shahzaad, B., Alkouz, B. & Bouguettaya, A. "Constraint-aware trajectory for drone delivery services". In: *International Conference on Service-Oriented Computing*. Springer International Publishing, 2021, pp. 306–310. https://doi.org/10.48550/arXiv.2111.00339.

[51] Raut, H. S., Seo, J. H. & Mittal, R. "Harnessing Leading-Edge Vortices for Improved Thrust Performance of Wave-Induced Flapping Foil Propulsors". *Scientific Reports* (2025). https://doi.org/10.1038/s41598-025-98585-w.

[52] Ruh, M. L. & Hwang, J. T. "Fast and robust computation of optimal rotor designs using blade element momentum theory". *AIAA Journal* 61.9 (2023), pp. 4096–4111. https://doi.org/10.2514/1.J062611.

[53] Keuter, R. J., Kirsch, B., Friedrichs, J. & Ponick, B. "Design Decisions for a Powertrain Combination of Electric Motor and Propeller for an Electric Aircraft". *IEEE Access* 11 (2023), pp. 79144–79155. https://doi.org/10.1109/ACCESS.2023.3299816.

[54] Abitha, M. A. & Saleem, A. "Quadrotor Modeling Approaches and Trajectory Tracking Control Algorithms: A Review". *International Journal of Robotics and Control Systems* 4.1 (2024), pp. 401–426. https://doi.org/10.31763/ijrcs.v4i1.1324.

[55] Tatjewski, P. "Effective nonlinear predictive and CTC-PID control of rigid manipulators". *Journal of Automation Mobile Robotics and Intelligent Systems* 18 (2024). https://doi.org/10.14313/jamris/2-2024/8.

[56] Hadid, S., Zamoum, R. B. & Refis, Y. "Linear and nonlinear control design for a quadrotor". *Bulletin of Electrical Engineering and Informatics* 14.2 (2025), pp. 940–955. https://doi.org/10.11591/eei.v14i2.8234.

[57] Cai, X., Win, S. K., Bhardwaj, H. & Foong, S. "Modeling, control and implementation of adaptive reconfigurable rotary wings (arrows)". *IEEE/ASME Transactions on Mechatronics* 28.4 (2023), pp. 2282–2292. https://doi.org/10.1109/TMECH.2023.3235346.

[58] Berrah, D., Chapoutot, A. & Garoche, P. L. "SCvxPyGen: Autocoding SCvx Algorithm". In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. 2024, pp. 5086–5093. https://doi.org/10.1109/CDC56724.2024.10886875.

[59] Fernandez, L. F. "Controllability-aware multidisciplinary design optimization of small vertical take-off and landing vehicles". PhD thesis. Ecole Nationale Aviation Civile, 2024. https://enac.hal.science/tel-04998365.

[60] Wang, J., Zhang, H., Zhou, X., Liu, W. & Yuan, D. "Joint resource allocation and trajectory design for energy-efficient UAV assisted networks with user fairness guarantee". *IEEE Internet of Things Journal* (2024). https://doi.org/10.3390/math10203840.

[61] Arowolo, M. "Numerical Modelling of Natural Fibres Reinforced Thermoplastic Composites". PhD thesis. University of Toulouse III Paul Sabatier / INSA Toulouse, 2023. https://doi.org/10.1016/j.mechmat.2025.105318.

[62] Simmons, B. M., Gresham, J. L. & Woolsey, C. A. "Aero-propulsive modeling for propeller aircraft using flight data". *Journal of Aircraft* 60.1 (2023), pp. 81–96. https://doi.org/10.2514/1.C036773.

[63] Guevara, B. S., Recalde, L. F., Varela-Aldás, J., Gandolfo, D. C. & Toibero, J. M. "Quadcopters Control Using Online Dynamic Mode Decomposition". In: *IFAC-PapersOnLine*. Vol. 56. 3. 2023, pp. 589–594. https://doi.org/10.1016/j.ifacol.2023.12.088.

[64] Abad, H. R., Mahmoodi, A. & Pazooki, F. "A New Approach to Analyzing Behavioral Change Propagation in Redesigning of Drone Camera Stabilizer". *Engineering* 13.12 (2021), pp. 707–738. https://doi.org/10.4236/eng.2021.1312051.

[65] Hossain, R. “A short review of the drone technology”. *International Journal of Mechatronics and Manufacturing Technology* 7.2 (2022), pp. 53–68. https://iaeme.com/Home/article_id/IJM_15_01_017.

[66] Vos, R. & Luyten, Q. “Design of an Electronic Speed Controller”. PhD thesis. Delft University of Technology, 2023. https://shorturl.at/M4dPj.

[67] Etewa, M., Safwat, E., Abozied, M. A. & El-Khatib, M. M. “Modeling and systematic investigation of a small-scale propeller selection”. *Journal of Engineering Science and Military Technologies* 7.1 (2023), pp. 15–21. https://doi.org/10.21608/ejmtc.2022.171071.1239.

[68] Jennan, N., Darkaoui, W., Mellouli, E. & Arjdal, E. “Enhanced Quadrotor Drone Control: Integration of Backstepping Method with Butterfly Optimization Algorithm”. In: *2025 5th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*. IEEE, 2025, pp. 1–8. https://doi.org/10.1109/IRASET64571.2025.11008299.

[69] Santiaguillo-Salinas, J., Aranda-Bricaire, E. & Garcia-Lozano, H. “UAV Flight Comparison Using Backstepping: On-board Data and Observers”. *IEEE Latin America Transactions* 23.3 (2025), pp. 182–190. https://doi.org/10.1109/TLA.2025.10879177.

[70] Santoso, F., Garratt, M., Anavatti, S. & Petersen, I. “Robust Hybrid Nonlinear Control Systems for the Dynamics of a Quadcopter Drone”. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50.8 (2018). https://doi.org/10.1109/TSMC.2018.2836922.

[71] Mohammed, T., Benyssaad, Y. & Mostafa, L. “A comparative study between PID and PD-SMC and PD-ASMC control applied on a delta robot”. *Przeglad Elektrotechniczny* 97.10 (2021). https://doi.org/10.15199/48.2021.10.01.

[72] Cetinsaya, B., Reiners, D. & Cruz-Neira, C. “From PID to swarms: A decade of advancements in drone control and path planning-A systematic review (2013–2023)”. *Swarm and Evolutionary Computation* 89 (2024), p. 101626. https://doi.org/10.1016/j.swevo.2024.101626.

[73] Meguro, Y., Masuda, S. & Toyoda, M. “Application and Performance Evaluation of Backstepping Control for Drone 3D Trajectory Control”. *Electrical Engineering in Japan* (2025). https://doi.org/10.1002/eej.23502.

[74] Li, Y., Fan, Y., Li, K., Liu, W. & Tong, S. “Adaptive optimized backstepping control-based RL algorithm for stochastic nonlinear systems with state constraints and its application”. *IEEE Transactions on Cybernetics* 52.10 (2021), pp. 10542–10555. https://doi.org/10.1109/TCYB.2021.3069587.

[75] Boussafa, A., Rabeh, R., Ferfra, M. & Chennoufi, K. "Experimental test of optimizing maximum power point tracking performance in solar photovoltaic arrays based on backstepping control and optimized by genetic algorithm". *Results in Engineering* 23 (2024), p. 102746. https://doi.org/10.1016/j.rineng.2024.102746.

[76] Khaleel, R. Z., Khaleel, H. Z., Al-Hareeri, A. A., Al-Obaidi, M. & Humaidi, A. J. "Improved Trajectory Planning of Mobile Robot Based on Pelican Optimization Algorithm". *Journal Européen des Systèmes Automatisés* 57.4 (Aug. 2024). https://doi.org/10.18280/jesa.570408.

[77] Qiu, S., Dai, J. & Zhao, D. "Path planning of an unmanned aerial vehicle based on a multi-strategy improved pelican optimization algorithm". *Biomimetics* 9.10 (2024), p. 647. https://doi.org/10.3390/biomimetics9100647.

[78] Memari, M., Shakya, P., Shekaramiz, M., Seibi, A. & Masoum, M. "Review on the advancements in wind turbine blade inspection: Integrating drone and deep learning technologies for enhanced defect detection". *IEEE Access* (2024). https://doi.org/10.1109/ACCESS.2024.3371493.

[79] Kistner, J., Neuhaus, L. & Wildmann, N. "High-resolution wind speed measurements with quadcopter uncrewed aerial systems: calibration and verification in a wind tunnel with an active grid". *Atmospheric Measurement Techniques* 17.16 (2024), pp. 4941–4955. https://doi.org/10.5194/amt-17-4941-2024.

[80] Bouziane, G., Benyssaad, Y., Youssouf, M. & Bouziane, M. "Optimizing 6DOF Drone Path Tracking with a GWO-Enhanced Backstepping Controller in Windy Environment". In: *6th International Conference on Applied Engineering and Natural Sciences*. NTFEECE Book Chapter, Part 2, p20, Konya, Turkey. 2024. https://heyzine.com/flip-book/210a054a38.html.

[81] Ferdaus, M., Pratama, M., Anavatti, S. & Garratt, M. "PAC: A Novel Self-Adaptive Neuro-Fuzzy Controller for Micro Aerial Vehicles". *Information Sciences* 512 (2020). https://doi.org/10.1016/j.ins.2019.10.001.

[82] Rodríguez-Abreo, O., Aviles, M., Rodríguez-Reséndiz, J. & García-Cerezo, A. "Neural networks and genetic algorithms-based self-adjustment system for a backstepping controller of an unmanned aerial vehicle". *Alexandria Engineering Journal* 126 (2025), pp. 70–80. https://doi.org/10.1016/j.aej.2025.04.034.

[83] Guerrero-Castellanos, J. F., Vega-Alonzo, A., Durand, S., Marchand, N., Gonzalez-Diaz, V. R., Castañeda-Camacho, J. & Guerrero-Sánchez, W. F. "Leader-Following Consensus and Formation Control of VTOL-UAVs with Event-Triggered Communications". *Sensors* 19.24 (Dec. 2019), p. 5498. https://doi.org/10.3390/s19245498.

[84] Madeiras, J., Cardeira, C., Oliveira, P., Batista, P. & Silvestre, C. "Saturated trajectory tracking controller in the body-frame for quadrotors". *Drones* 8.4 (2024), p. 163. https://doi.org/10.3390/drones8040163.

[85] Novotňák, J., Szőke, Z., Kašper, P. & Šmelko, M. "Quadcopter Modeling Using a System for UAV Parameters Measurement". *Drones* 8.7 (2024), p. 280. https://doi.org/10.3390/drones8070280.

[86] Li, F., Song, W., Song, B. & Zhang, H. "Dynamic modeling, simulation, and parameter study of electric quadrotor system of Quad-Plane UAV in wind disturbance environment". *International Journal of Micro Air Vehicles* 13 (2021). https://doi.org/10.1177/17568293211022211.

[87] Saibi, A., Boushaki, R. & Belaidi, H. "Backstepping control of drone". *Engineering Proceedings* 14.1 (2022), p. 4. https://doi.org/10.3390/engproc2022014004.

[88] Wen, G., Hao, W., Feng, W. & Gao, K. "Optimized Backstepping Tracking Control Using Reinforcement Learning for Quadrotor Unmanned Aerial Vehicle System". *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52.8 (Sept. 2021), pp. 5004–5015. https://doi.org/10.1109/TSMC.2021.3112688.

[89] Vazquez, R., Auriol, J., Bribiesca-Argomedo, F. & Krstic, M. "Backstepping for Partial Differential Equations". *arXiv preprint arXiv:2410.15146* (2024). https://doi.org/10.48550/arXiv.2410.15146.

[90] Bouziane, G., Benyssaad, Y., Youssouf, M. & Bouziane, M. "Comparative Performance of POA and GWO-Enhanced Backstepping Control for Robust 6DOF Drone Path Tracking in Windy Conditions". *Przeglad Elektrotechniczny* (2025). https://doi.org/10.15199/48.2025.05.20.

[91] Martins, L., Cardeira, C. & Oliveira, P. "Inner-outer feedback linearization for quadrotor control: two-step design and validation". *Nonlinear Dynamics* 110.1 (2022), pp. 479–495. https://doi.org/10.1007/s11071-022-07632-y.

[92] Bouziane, G., Benyssaad, Y., Youssouf, M. & Bouziane, M. "Garra_Rufa-Optimized Backstepping Controller for Drone Path Tracking in Windy Conditions". In: *International Conference on Artificial Intelligence and Sustainable Development (ICAISD)*. April 12th–13th. Relizane, Algeria, 2025.

[93] Mirjalili, S., Mirjalili, S. & Lewis, A. "Grey Wolf Optimizer". *Advances in Engineering Software* 69 (2014), pp. 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.

[94] Sun, Y., Lv, B., Yang, H. & Li, X. "Multi-UAV Trajectory Planning Based on Improved Multi-population Grey Wolf Optimizer Algorithm". In: *Chinese Control and Decision Conference*. Chengdu: IEEE, 2024. https://doi.org/10.1109/CCDC62350.2024.10587624.

[95] El Islam, L., M'hamed, O. & Omar, T. "Optimal Tuning Of Integral Saturation Back-Stepping Quadcopter's Controller Using Grey Wolf Optimizer". In: *2nd International Conference on Electrical Engineering and Automatic Control (ICEEAC)*. IEEE, 2024. https://doi.org/10.1109/ICEEAC61226.2024.10576298.

[96] Braik, M., Sheta, A., Aljahdali, S., El-Hefnawi, F., Al-Hiary, H. & Elashmawi, W. "Evolutionary optimization of Yagi–Uda antenna design using grey wolf optimizer". *Neural Computing and Applications* 37.6 (2025), pp. 4155–4183. https://doi.org/10.1007/s00521-024-10806-x.

[97] Jaber, A., Abdulbari, H., Shalash, N. & Abdalla, A. "Garra Rufa-inspired optimization technique". *International Journal of Intelligent Systems* 35.11 (2020), pp. 1831–1856. https://doi.org/10.1002/int.22274.

[98] Chillab, R., Jaber, A., Smida, M. & Sakly, A. "Optimal DG location and sizing to minimize losses and improve voltage profile using Garra Rufa optimization". *Sustainability* 15.2 (2023), p. 1156. https://doi.org/10.3390/su15021156.

[99] Trojovský, P. & Dehghani, M. "Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications". *Sensors* 22.3 (2022), p. 855. https://doi.org/10.3390/s22030855.

[100] Anshory, I., Jamaaluddin, J., Wisaksono, A., Sulistiyowati, I., Rintyarna, B., Fudholi, A., Rahman, Y. & Sopian, K. "Optimization DC-DC boost converter of BLDC motor drive by solar panel using PID and firefly algorithm". *Results in Engineering* 21 (2024), p. 101727. https://doi.org/10.1016/j.rineng.2023.101727.

[101] Konatowski, S. & Tatko, S. "Behaviour of unmanned aircraft in formation". *Przeglad Elektrotechniczny* 99.5 (2023). https://doi.org/10.15199/48.2023.05.10.

[102] Sif-eddine, B. & Mohamed-Amine, K. "Optimizing the tensile strength of friction stir welded joints using pelican optimization algorithm". In: *Proceedings of the First National Conference in Computer Science Research and its Applications (RIA'23)*. Online. Relizane (DZ): University of Relizane, May 2023.

[103] Sousa Aguiar, A., Pereira Pinto, V., Carvalho Sousa, L., Da Silva Pinheiro, J. & Nascimento Sousa, J. do. "Route planning for multiple unmanned aerial vehicles". *Przegląd Elektrotechniczny* 9 (2024). https://doi.org/10.15199/48.2024.09.43.

[104] Huo, Z., Yuan, M., Zhang, S. & Zhang, X. "Observer-Based Adaptive Robust Force Control of a Robotic Manipulator Integrated with External Force/Torque Sensor". *Actuators* 14.3 (2025), p. 116. https://doi.org/10.3390/act14030116.

[105] Altınörs, A. & Kuzu, F. "Dynamic analysis of a quadcopter using PID, adaptive and LQR control methods". *International Journal of Innovative Engineering Applications* 5.2 (2021), pp. 65–74. https://doi.org/10.46460/ijiea.929552.

[106] Ihnak, M. S. & Edardar, M. M. "Comparing LQR and PID controllers for quadcopter control effectiveness and cost analysis". In: *2023 IEEE 11th International Conference on Systems and Control (ICSC)*. 2023, pp. 754–759. https://doi.org/10.1109/ICSC58660.2023.10449763.

[107] Khan, A. A., Kurokawa, S., Do, T. D. & Ali, M. H. "Development of an LQR-Based Control Algorithm for Quadcopter". In: *2024 IEEE 12th Conference on Systems, Process and Control (ICSPC)*. 2024, pp. 142–147. https://doi.org/10.1109/ICSPC63060.2024.10862615.

[108] Ata, B. & Gencal, M. C. "Comparison of optimization approaches on linear quadratic regulator design for trajectory tracking of a quadrotor". *Evolutionary Intelligence* 17.5 (2024), pp. 3225–3240. https://doi.org/10.1007/s12065-024-00928-5.

[109] Dhewa, O. A., Arifin, F., Priyambodo, A. S., Winursito, A. & Mustafa, Y. M. "Attitude UAV Stability Control Using Linear Quadratic Regulator-Neural Network (LQR-NN)". *IIUM Engineering Journal* 25.2 (2024), pp. 246–265. https://doi.org/10.31436/iiumej.v25i2.3119.

[110] Wu, F., Wang, G., Zhuang, S., Wang, K., Keimer, A., Stoica, I. & Bayen, A. "Composing MPC with LQR and neural network for amortized efficiency and stable control". *IEEE Transactions on Automation Science and Engineering* 21.2 (2023), pp. 2088–2101. https://doi.org/10.1109/TASE.2023.3259428.

[111] Sif-eddine, B. & Bouziane, G. "Adaptive LQR Control with Neural Network Tuning for Energy Saving in 6-DOF Quadcopter Systems". In: *International Conference on Artificial Intelligence and Sustainable Development (ICAISD)*. April 12th-13th. Relizane, Algeria, Apr. 2025.

[112] Horvath, L. & Petré, M. *Quadcopter Control System Design and Implementation with LQR assisted by Kalman Filter: An Experimental Study*. https://www.diva-portal.org/smash/get/diva2:1775501/FULLTEXT01.pdf. 2024.

[113] Ahmed, S., Qiu, B., Kong, C. W., Xin, H., Ahmad, F. & Lin, J. "A data-driven dynamic obstacle avoidance method for liquid-carrying plant protection UAVs". *Agronomy* 12.4 (2022), p. 873. https://doi.org/10.3390/agronomy12040873.

[114] Bravo, L. P. & Zuñiga, C. G. "Nonlinear trajectory tracking with a 6DOF AUV using an MRAFC controller". *IEEE Latin America Transactions* 23.2 (2025), pp. 160–171. https://doi.org/10.1109/TLA.2025.10851362.

[115] Zhi, Y., Weiqing, W., Jing, C. & Razmjooy, N. "Interval linear quadratic regulator and its application for speed control of DC motor in the presence of uncertainties". *ISA Transactions* 125 (2022), pp. 252–259. https://doi.org/10.1016/j.isatra.2021.07.004.

[116] Komaee, A. “Dynamic gain adaptation in linear quadratic regulators”. *IEEE Transactions on Automatic Control* 69.8 (2023), pp. 5094–5108. https://doi.org/10.1109/TAC.2023.3344551.

[117] Sahoo, S. R. & Manivannan, P. V. “Hybrid high-level controller (H2LC) using LQR and MRAC for enhanced quadcopter trajectory tracking performance”. *Journal of Control and Decision* (2025), pp. 1–22. https://doi.org/10.1080/23307706.2025.2469887.

[118] Ubadiake, O., Abbe, G., Amoako, T., Bonet, M., Momoh, M., Adeboye, C. & Ter, K. “Robust LQR-Based Autopilot Design for Hybrid Energy Harvesting UAVs”. *FUDMA Journal of Renewable and Atomic Energy* 1.2 (2024), pp. 92–123. https://doi.org/10.33003/fjorae.2024.0102.07.

[119] Nosrati, K., Belikov, J., Tepljakov, A. & Petlenkov, E. “Revisiting the LQR Problem of Singular Systems”. *IEEE/CAA Journal of Automatica Sinica* 11.11 (2024), pp. 2236–2252. https://doi.org/10.1109/JAS.2024.124665.

[120] Sif-eddine, B., Mohamed, S. & Al-Sabur, R. “Parametric Analysis for Torque Prediction in Friction Stir Welding Using Machine Learning and Shapley Additive Explanations”. *Journal of Computational Applied Mechanics* 55.1 (2024), pp. 113–124. https://doi.org/10.22059/jcamech.2024.370055.924.

[121] Muthusamy, P. K., Mohiuddin, M. B., Peringal, A., Suthar, B., Muthusamy, R., Hussain, I., Seneviratne, L. & Zweiri, Y. “Aerial manipulation of long objects using adaptive neuro-fuzzy controller under battery variability”. *Scientific Reports* 15.1 (2025), pp. 1–20. https://doi.org/10.1038/s41598-025-94937-8.

[122] Devarajan, G. “Evolving Neural Network Controllers for Modular Origami Robots Using Embodied Artificial Intelligence”. PhD thesis. Purdue University Graduate School, 2024. https://doi.org/10.25394/PGS.28908764.v1.

[123] Sif-eddine, B. “Numerical and parametric study of FSW process using a pneumatic source”. PhD thesis. University of Relizane, Algeria, 2024.

[124] Lin, J., Miao, Z., Wang, Y., Hu, G., Wang, X. & Wang, H. “Error-state LQR geofencing tracking control for underactuated quadrotor systems”. *IEEE/ASME Transactions on Mechatronics* 29.2 (2023), pp. 1146–1157. https://doi.org/10.1109/TMECH.2023.3292893.

[125] Manaa, Z. M., Abdallah, A. M., Abido, M. A. & Ali, S. S. “Koopman-LQR controller for quadrotor UAVs from data”. In: *2024 IEEE International Conference on Smart Mobility (SM)*. 2024, pp. 153–158. https://doi.org/10.1109/SM63044.2024.10733422.

[126] Kirgni, H. B. & Wang, J. “LQR-based adaptive TSMC for nuclear reactor in load following operation”. *Progress in Nuclear Energy* 156 (2023), p. 104560. https://doi.org/10.1016/j.pnucene.2022.104560.

[127] Elkhatem, A. S. & Engin, S. N. “Robust LQR and LQR-PI control strategies based on adaptive weighting matrix selection for a UAV position and attitude tracking control”. *Alexandria Engineering Journal* 61.8 (2022), pp. 6275–6292. https://doi.org/10.1016/j.aej.2021.11.057.

[128] Huang, G., Cai, Y., Liu, J., Qi, Y. & Liu, X. “A novel hybrid discrete grey wolf optimizer algorithm for multi-UAV path planning”. *Journal of Intelligent and Robotic Systems* 103 (2021), pp. 1–8. https://doi.org/10.1007/s10846-021-01490-3.

[129] Maamri, M., Bouzeboudja, H. & Tandjaoui, M. N. “The use of Grey Wolf Optimizer (GWO) for solving the economic dispatch problems based on renewable energy in Algeria a case study of “Naama Site””. *Przegląd Elektrotechniczny* 95.6 (2019), pp. 32–39. https://doi.org/10.15199/48.2019.06.07.

[130] Choubey, C. & Ohri, J. “GWO-based tuning of LQR-PID controller for a 3-DOF parallel manipulator”. *IETE Journal of Research* 69.7 (2023), pp. 4378–4393. https://doi.org/10.1080/03772063.2021.1958068.

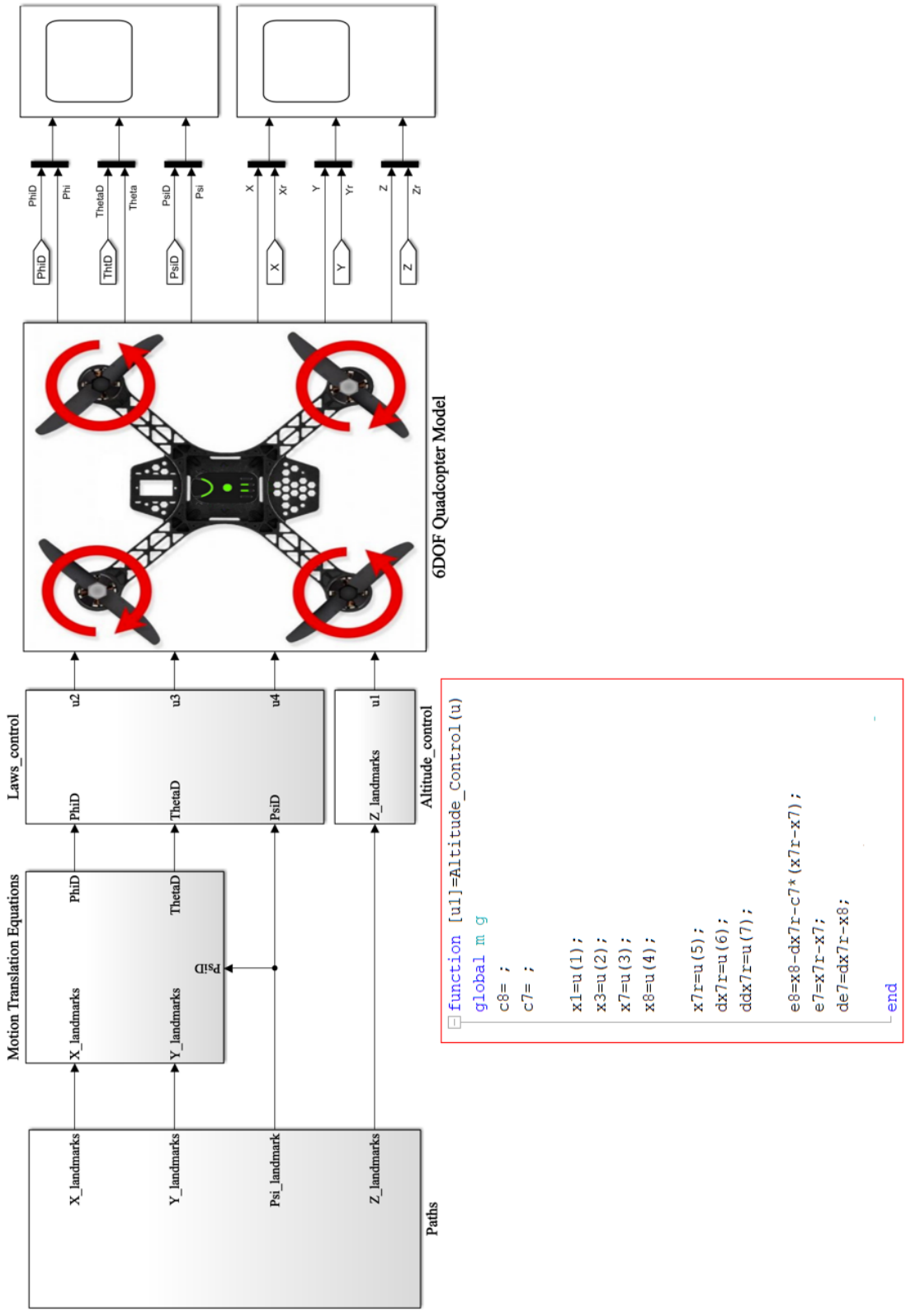
[131] Büyüker, Y. & İlhan, İ. “Parameter Optimization of LQR Controller Applied to Three Degrees of Freedom System with Hybrid Approach”. *Konya Journal of Engineering Sciences* 12.2 (2024), pp. 494–510. https://doi.org/10.36306/konjes.1291710.

ANNEXES

ANNEX I : MATLAB BLOCKS COMPILATION:

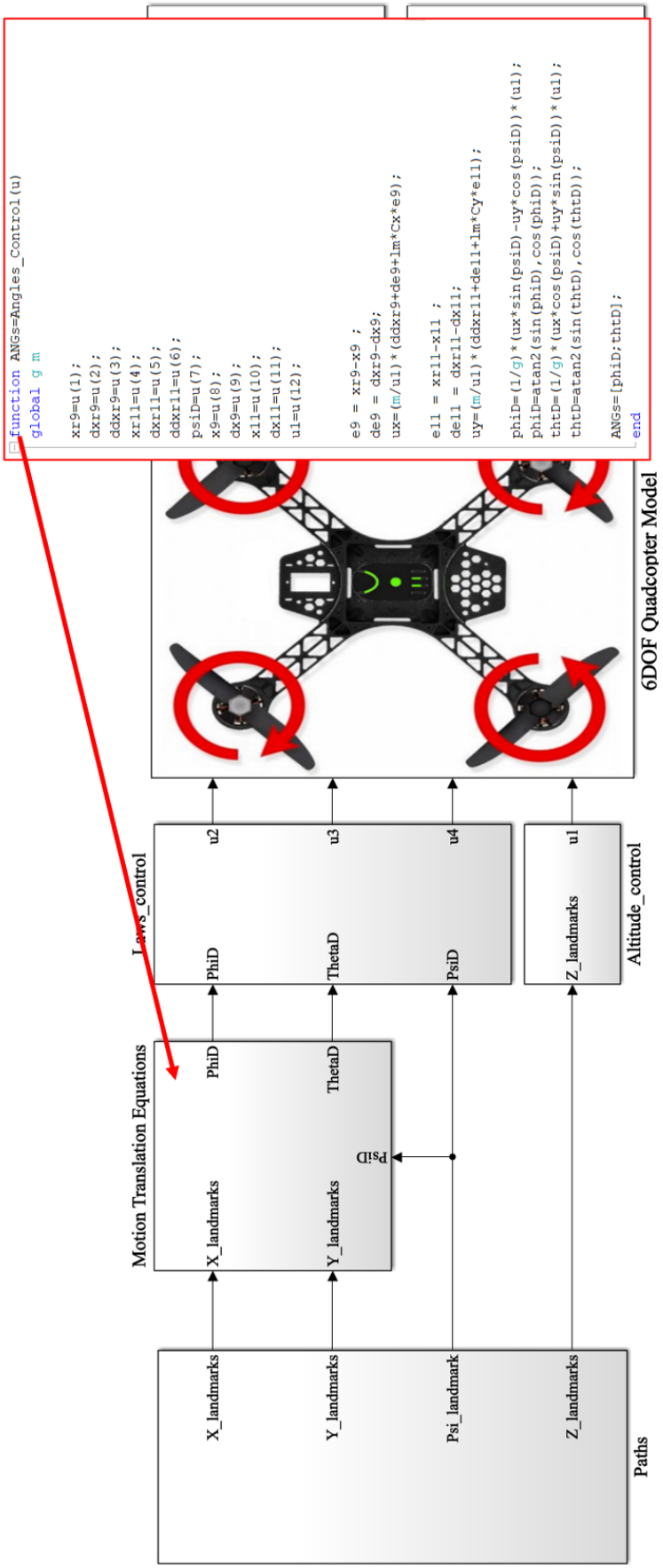
Backstepping Controller Blocks:

- **Altitude Control**:

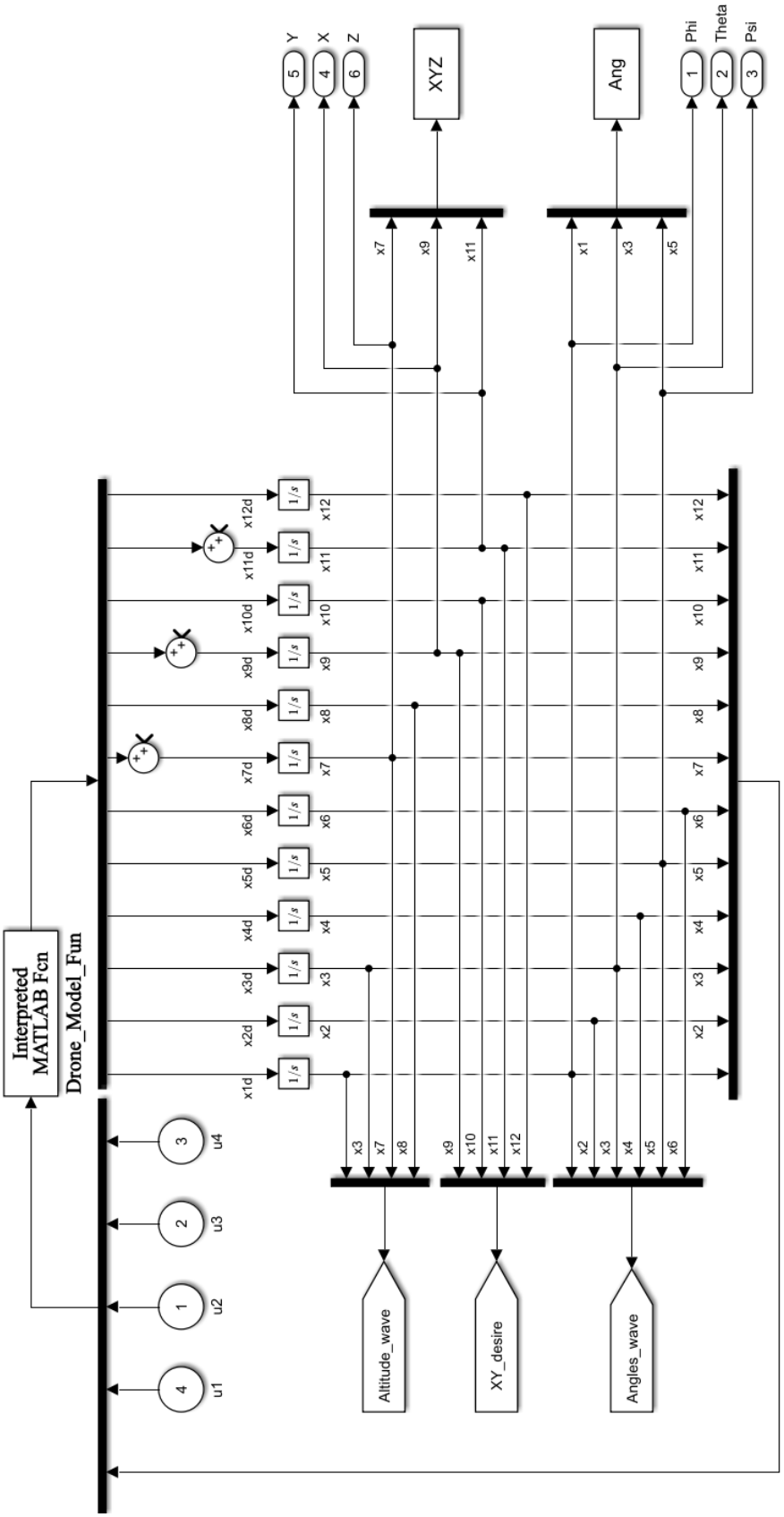


- **Laws Control**:

- **Angels Control**:

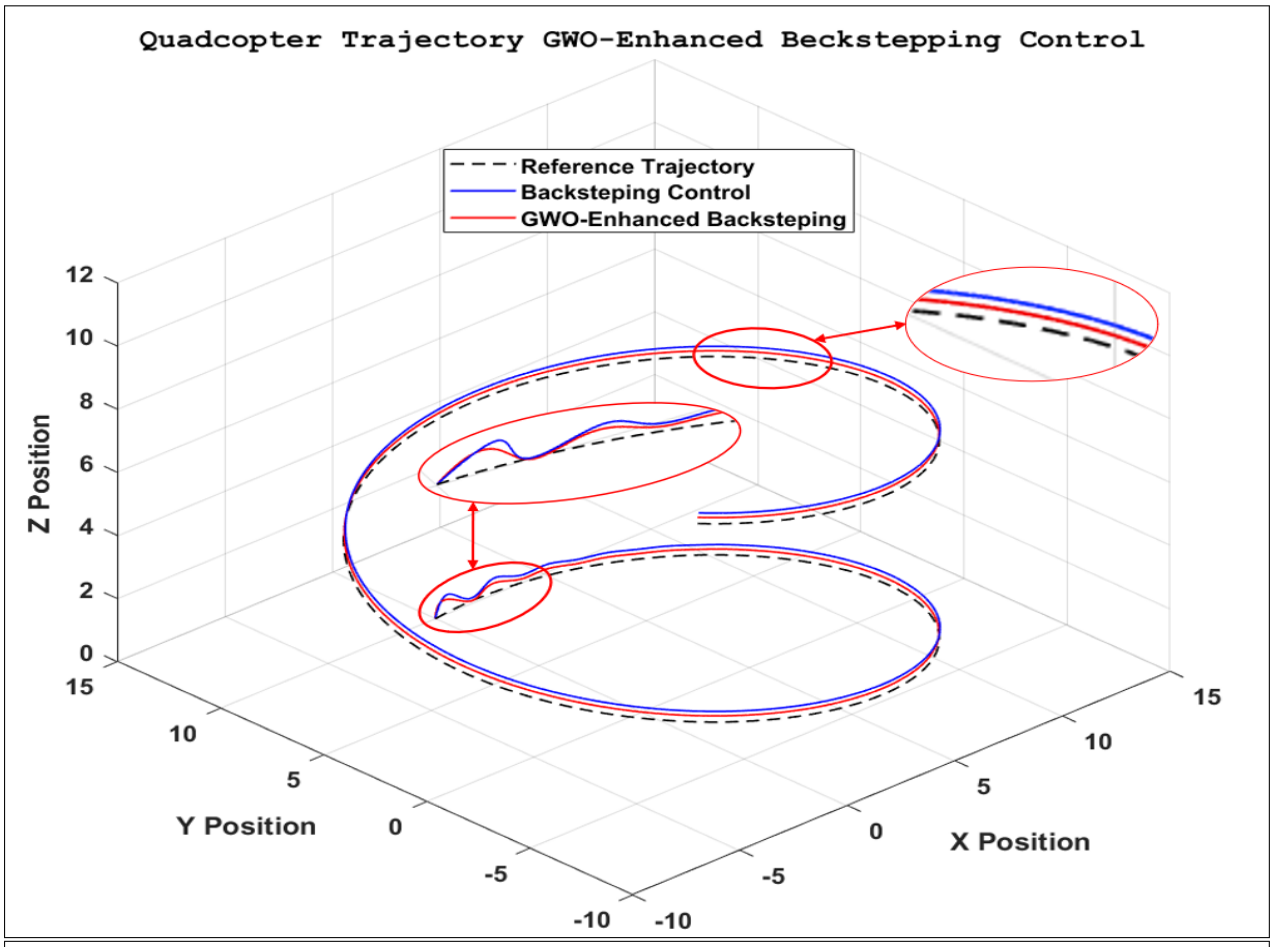


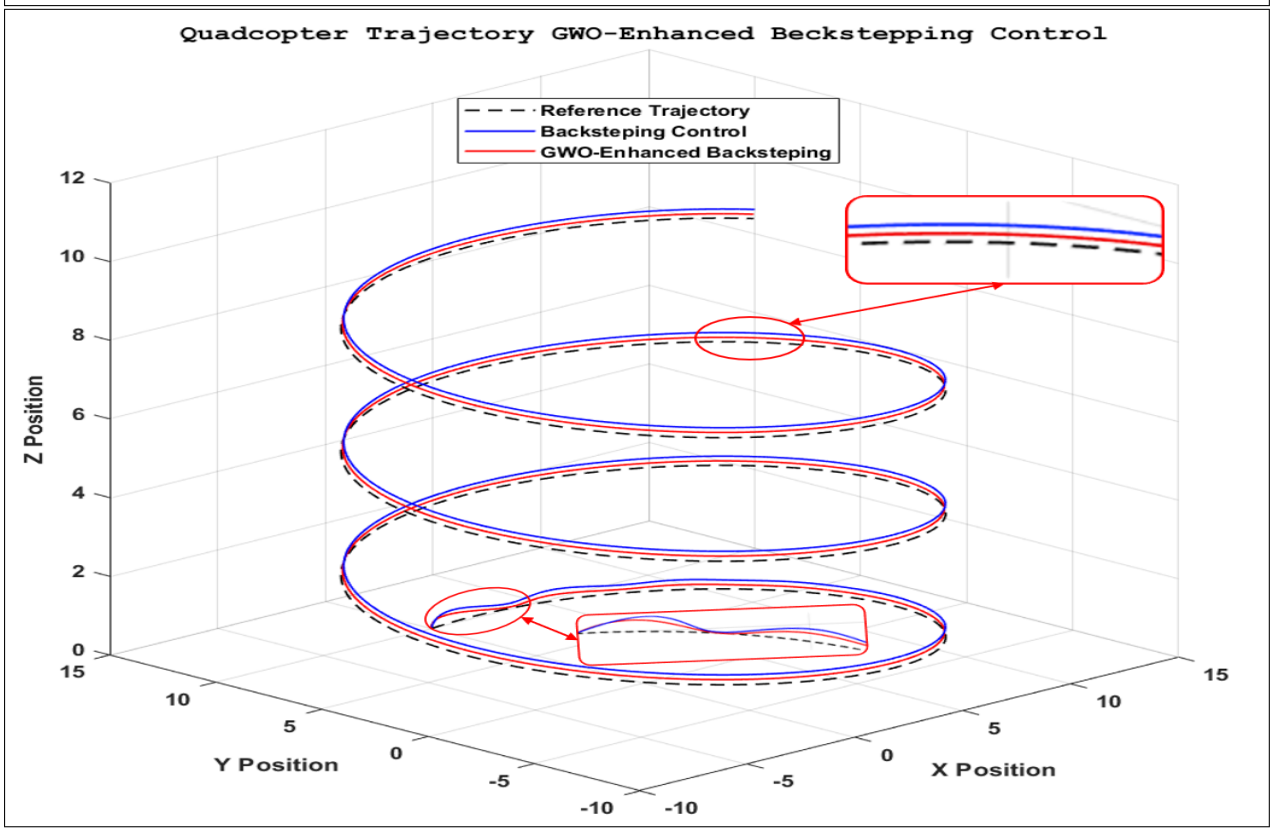
- **Drone Block**:

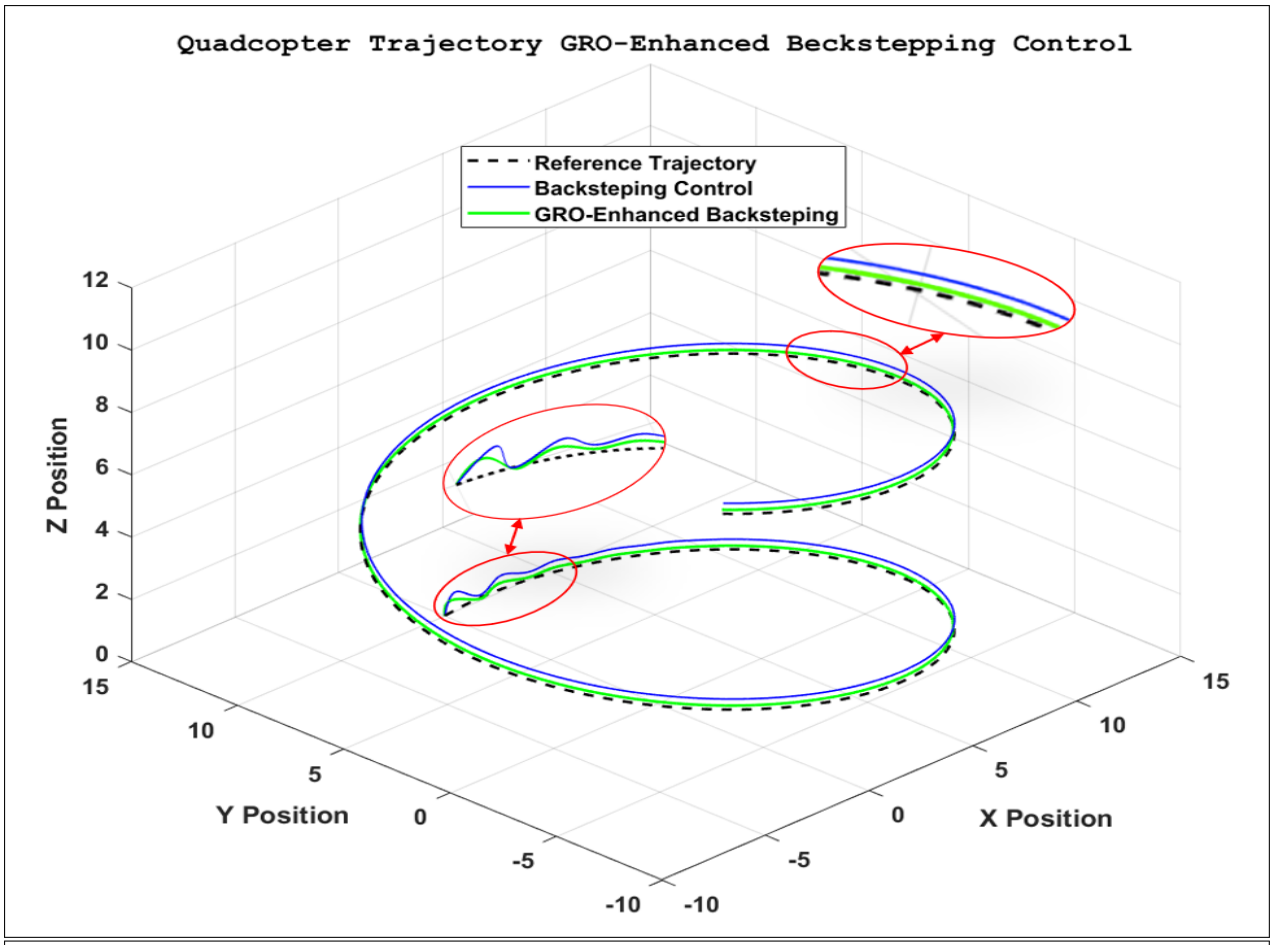

Interpreted MATLAB Fcn
Drone_Model_Fun
u1
u2
u3
u4
Altitude_wave
XY_desire
Angles_wave
XYZ
Ang
Y
X
Z
Phi
Theta
Psi

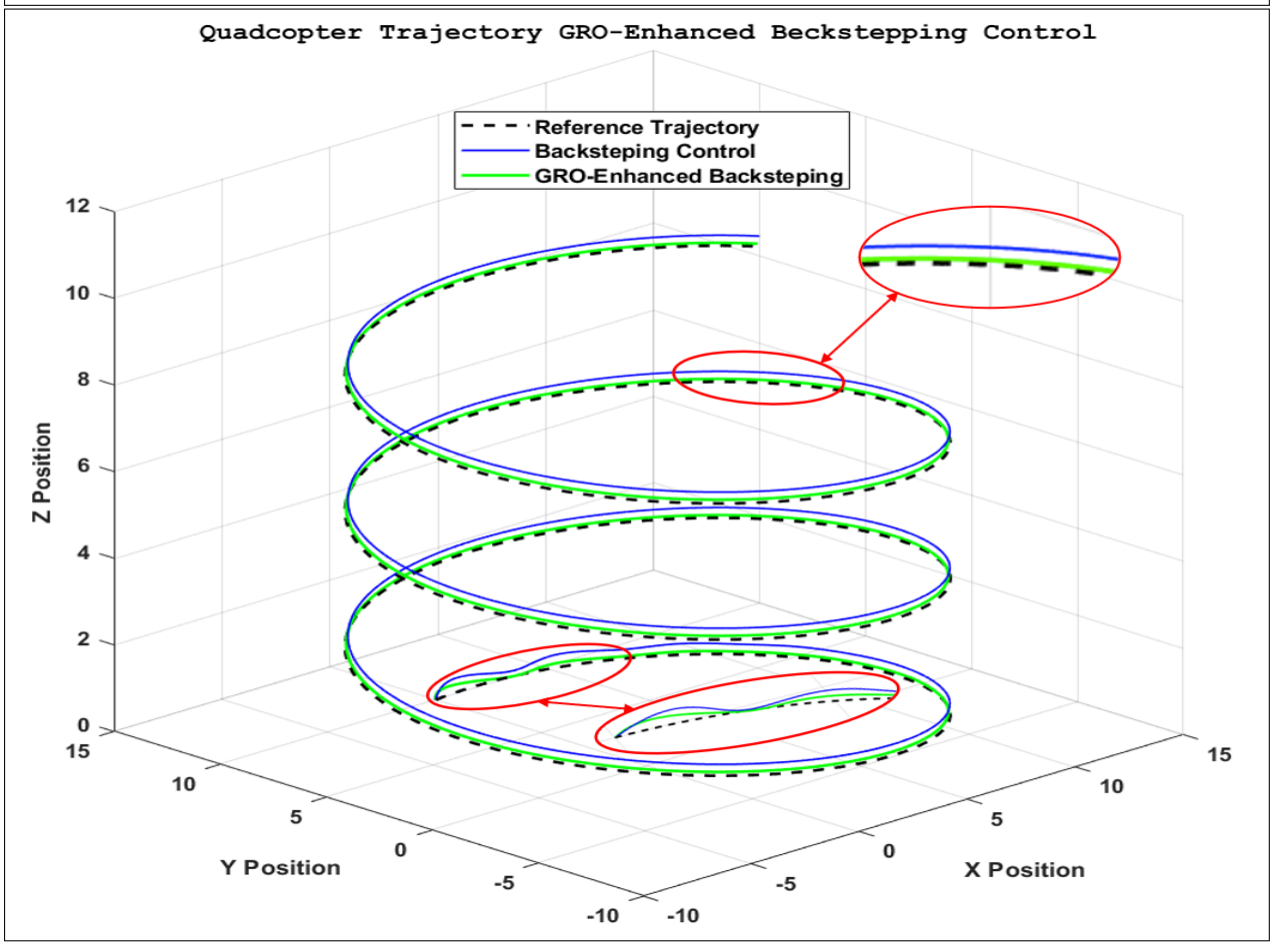
ANNEX II : MATLAB BLOCKS COMPILATION:

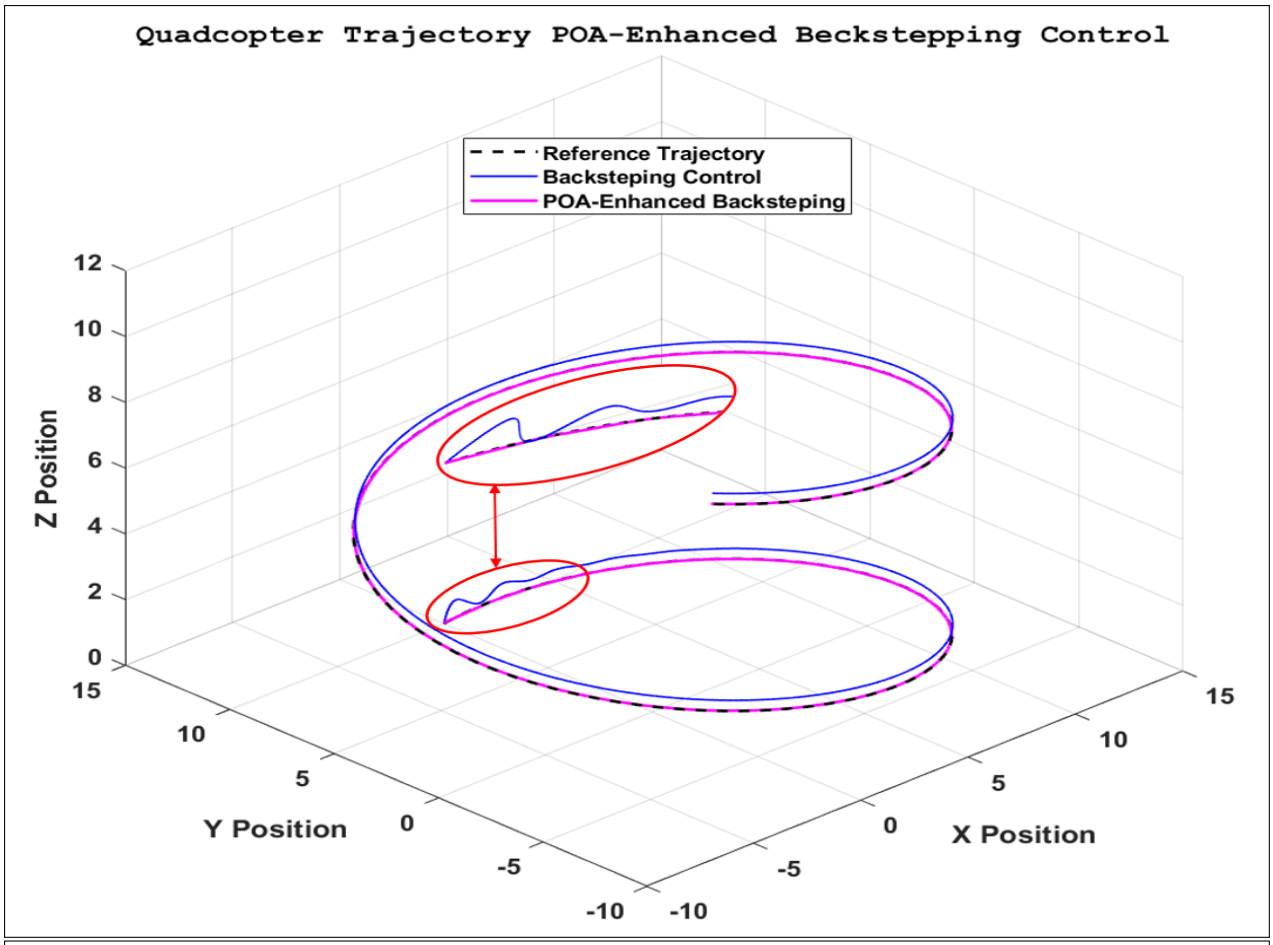
Results of Enhaned Backsteping:

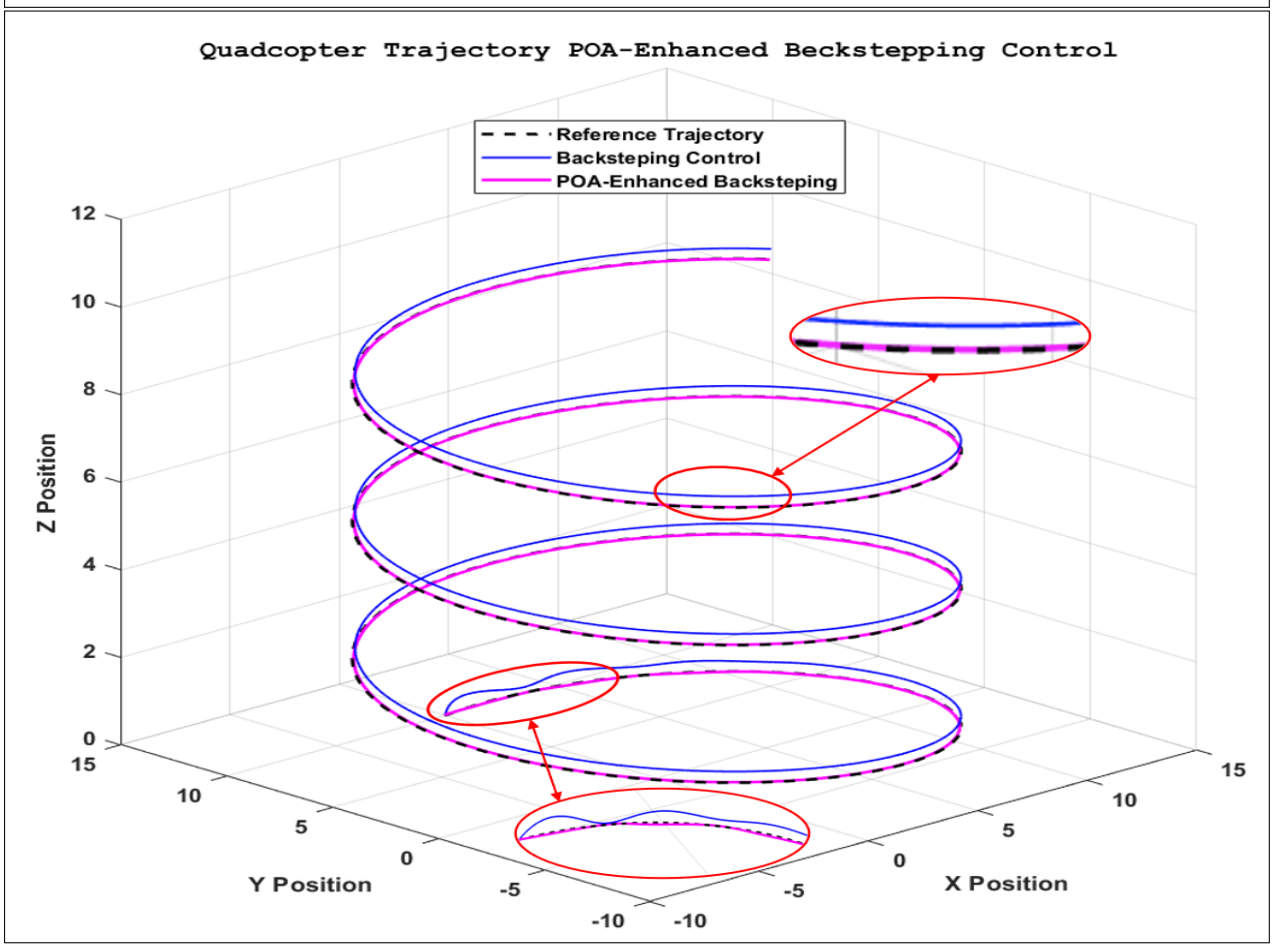


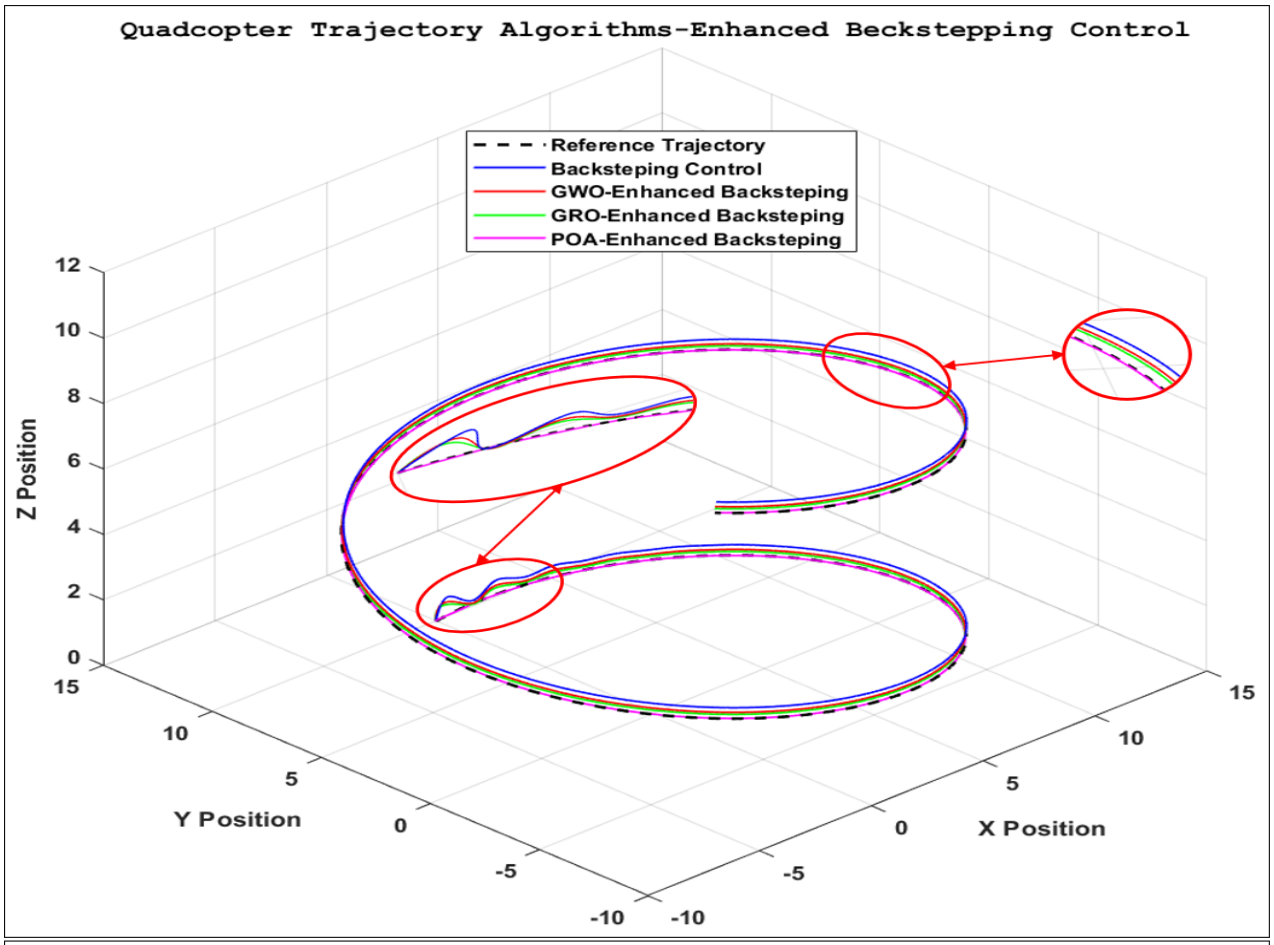


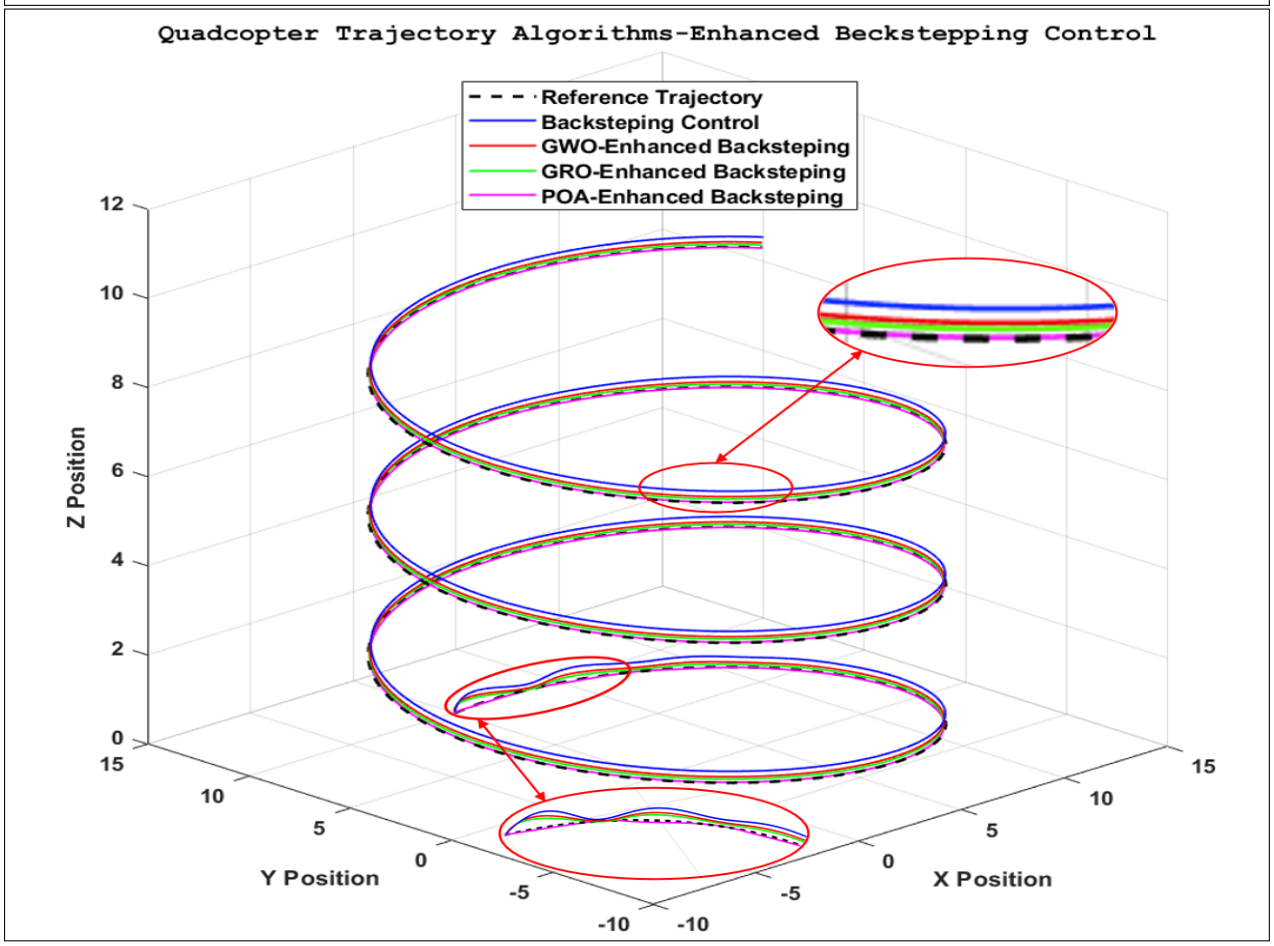

Quadcopter Trajectory GRO-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GRO-Enhanced Backsteping
Z Position
Y Position
X Position

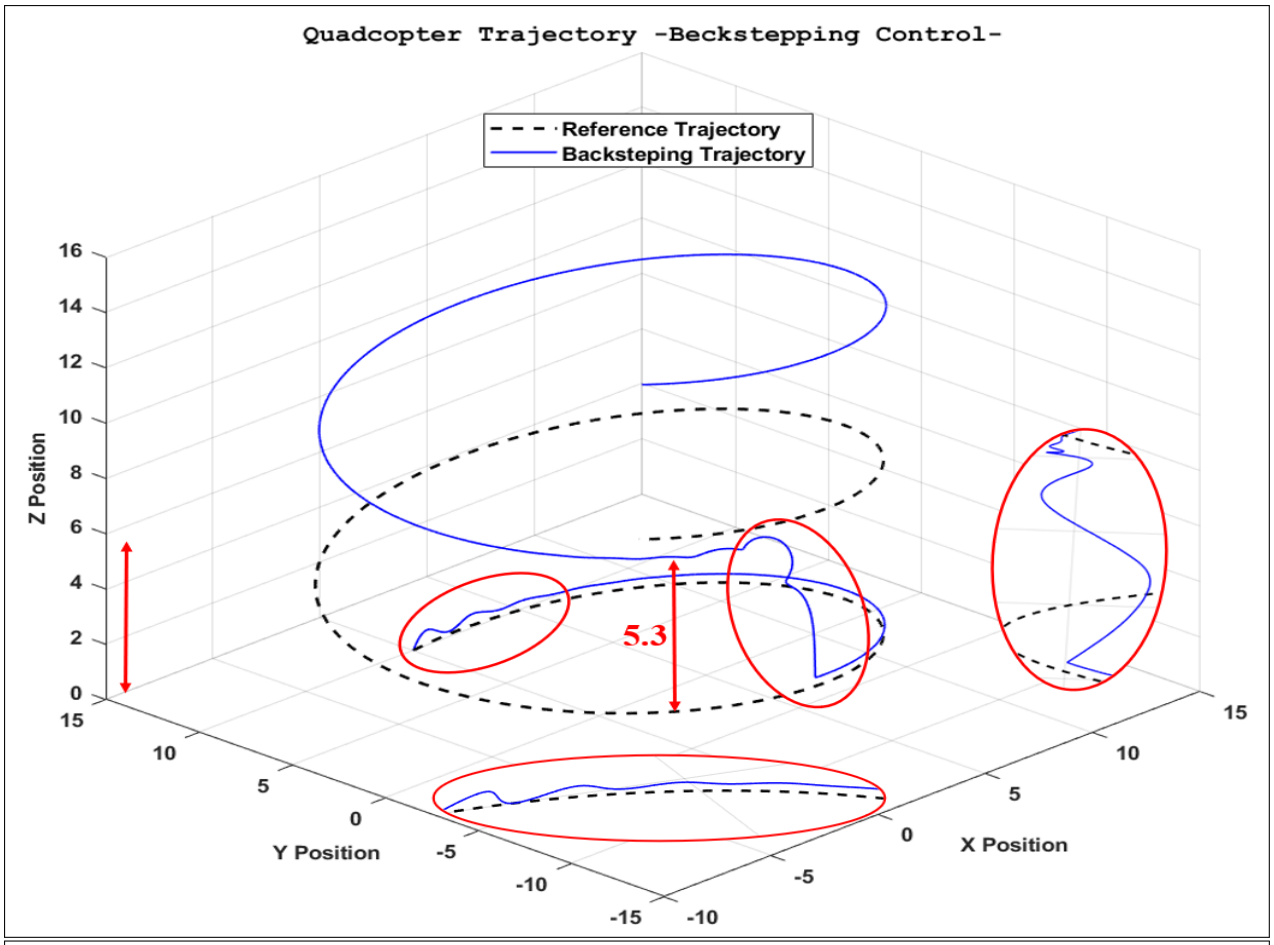

Quadcopter Trajectory GRO-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GRO-Enhanced Backsteping
Z Position
Y Position
X Position

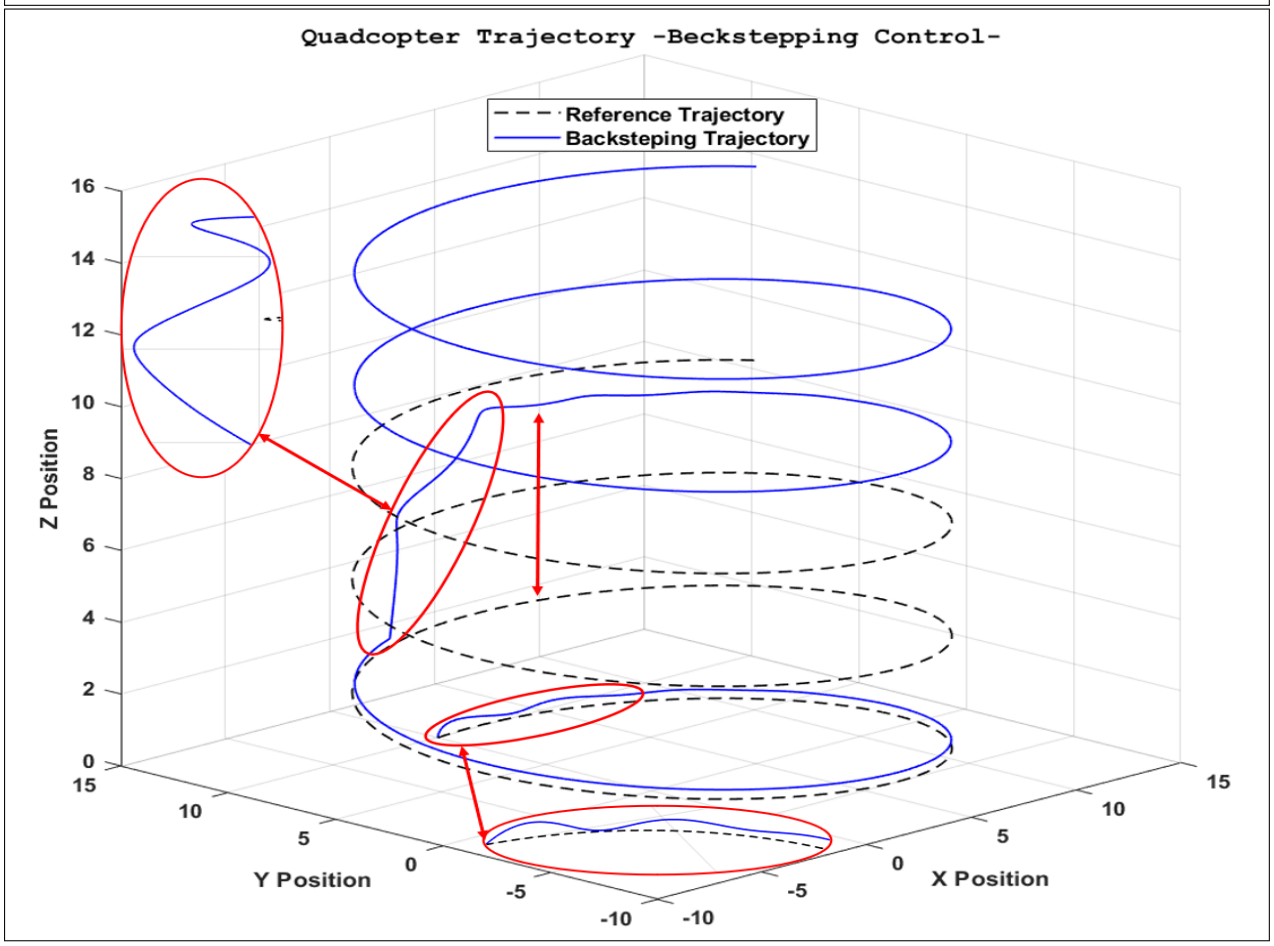

Quadcopter Trajectory POA-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
POA-Enhanced Backsteping
Z Position
Y Position
X Position

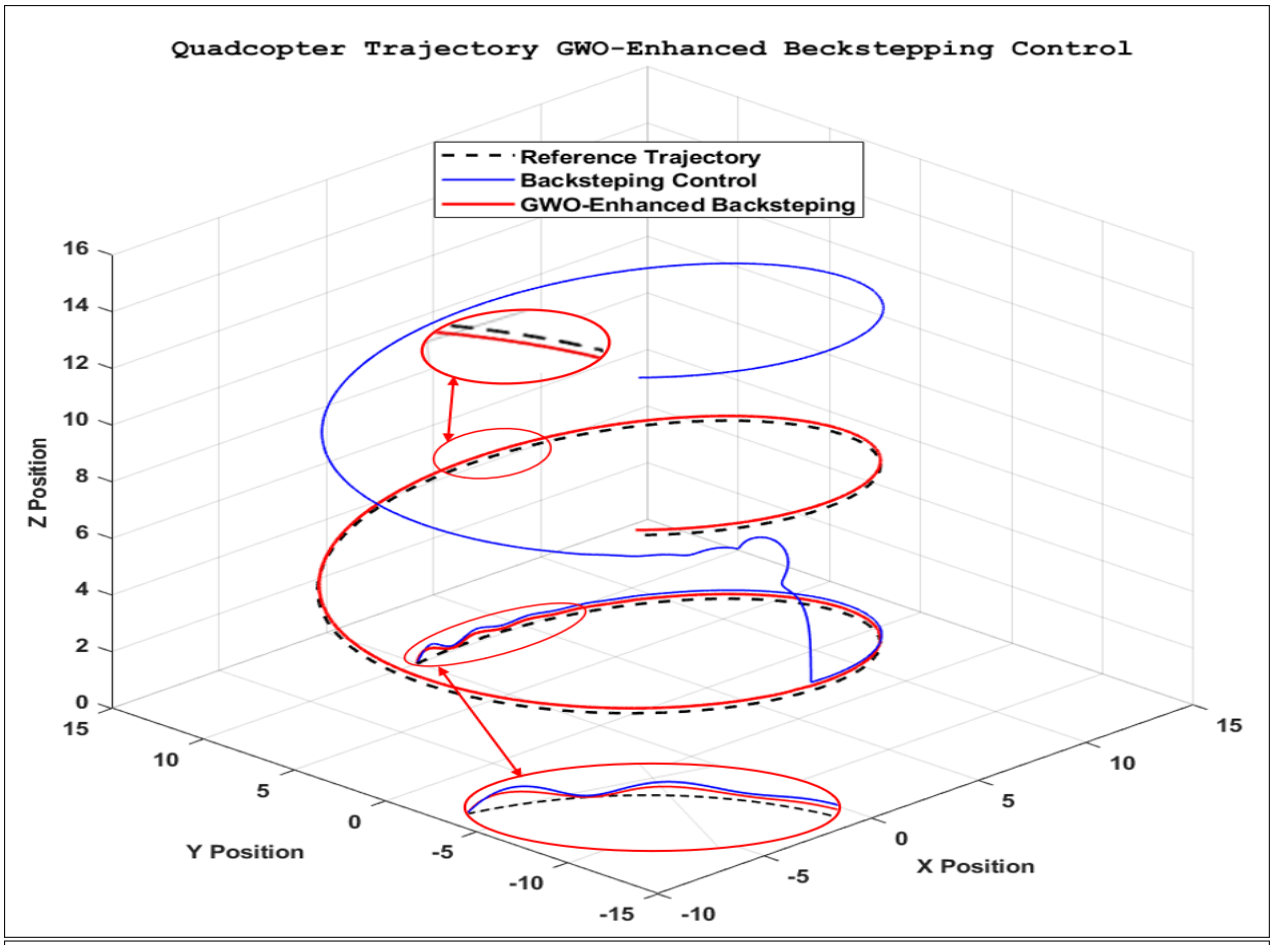

Quadcopter Trajectory POA-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
POA-Enhanced Backsteping
Z Position
Y Position
X Position

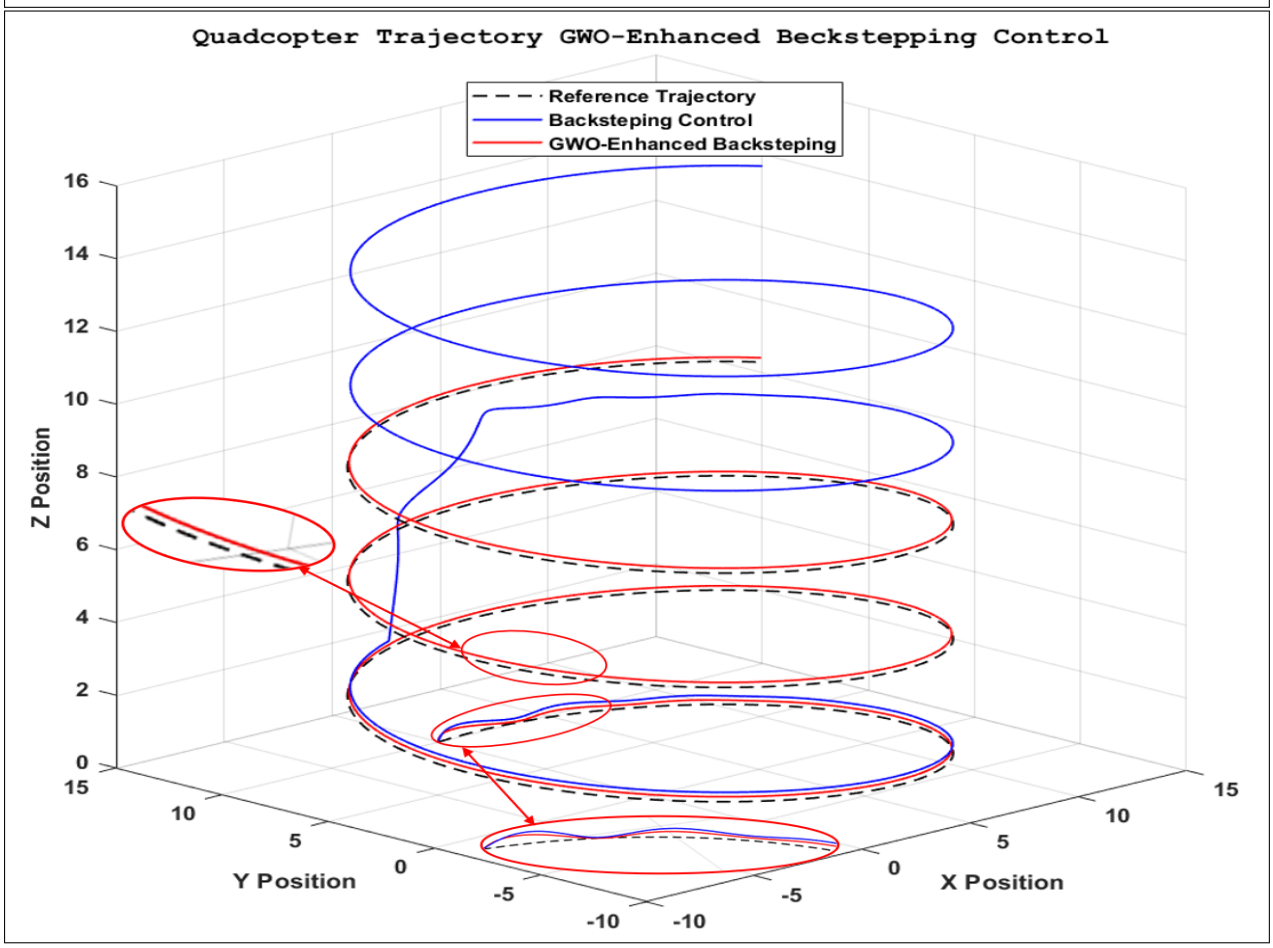

Quadcopter Trajectory Algorithms-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GWO-Enhanced Backsteping
GRO-Enhanced Backsteping
POA-Enhanced Backsteping
Z Position
Y Position
X Position

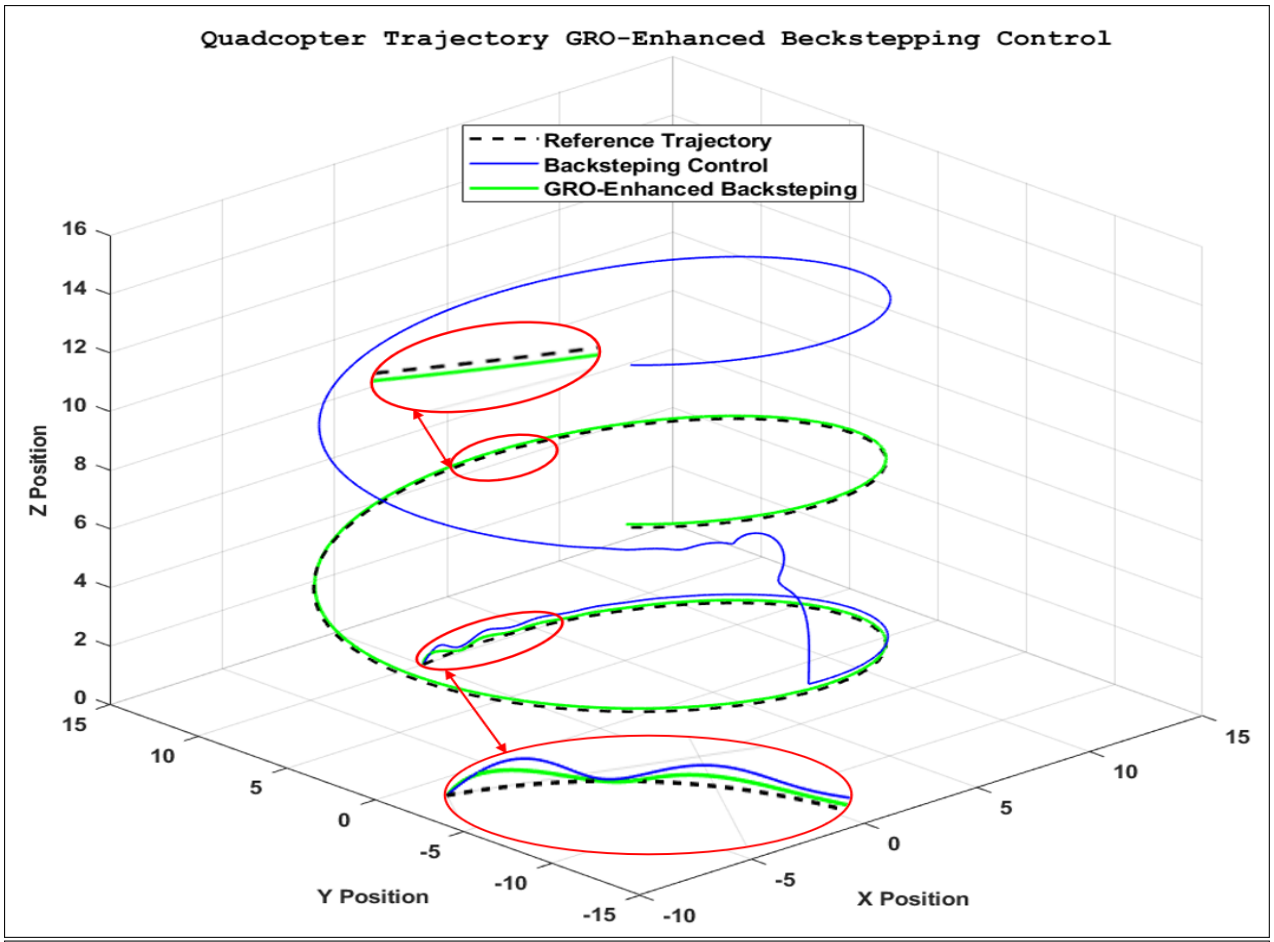

Quadcopter Trajectory Algorithms-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GWO-Enhanced Backsteping
GRO-Enhanced Backsteping
POA-Enhanced Backsteping
Z Position
Y Position
X Position

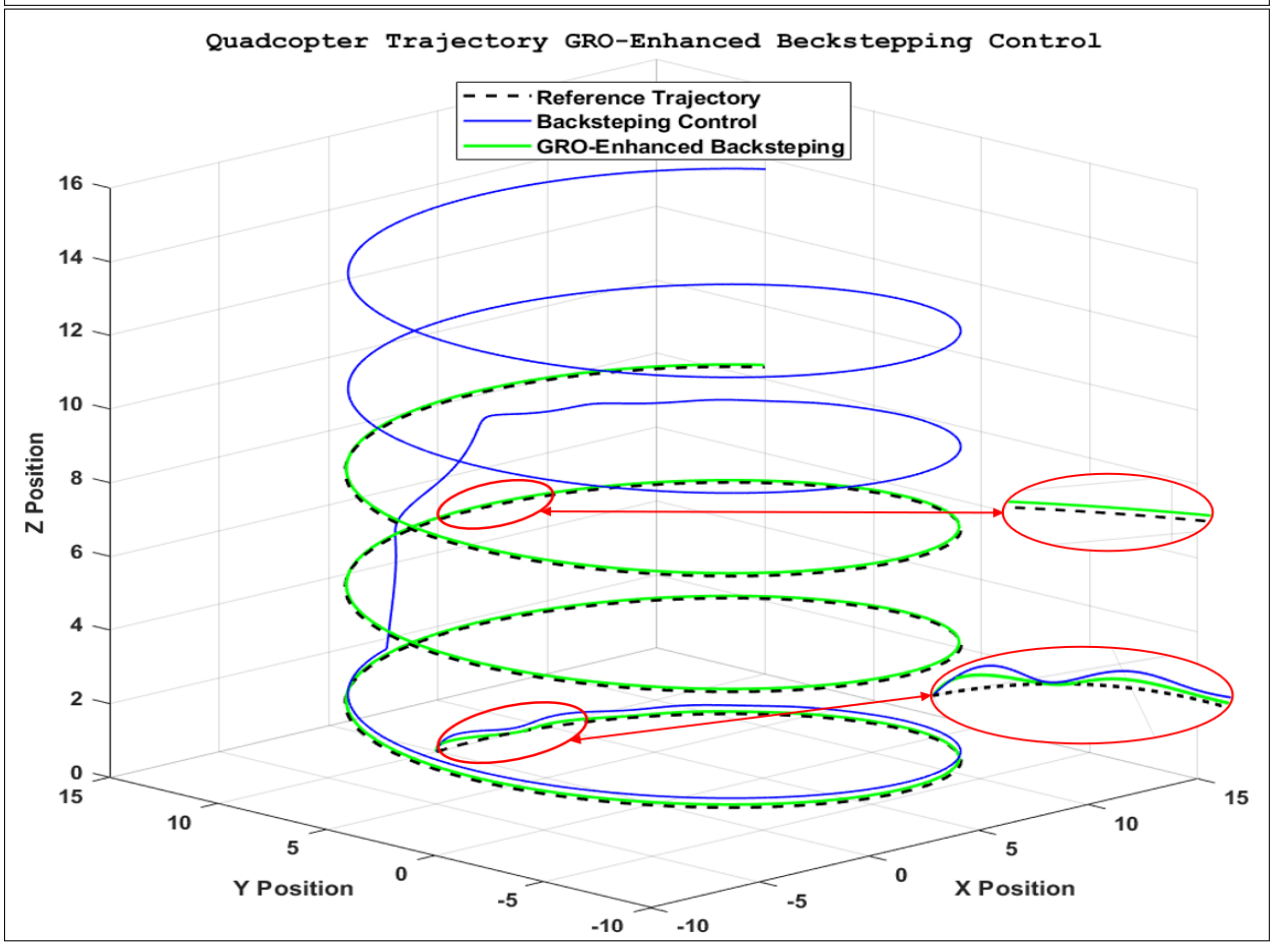

Quadcopter Trajectory -Beckstepping Control-
Reference Trajectory
Backsteping Trajectory
Z Position
Y Position
X Position
5.3

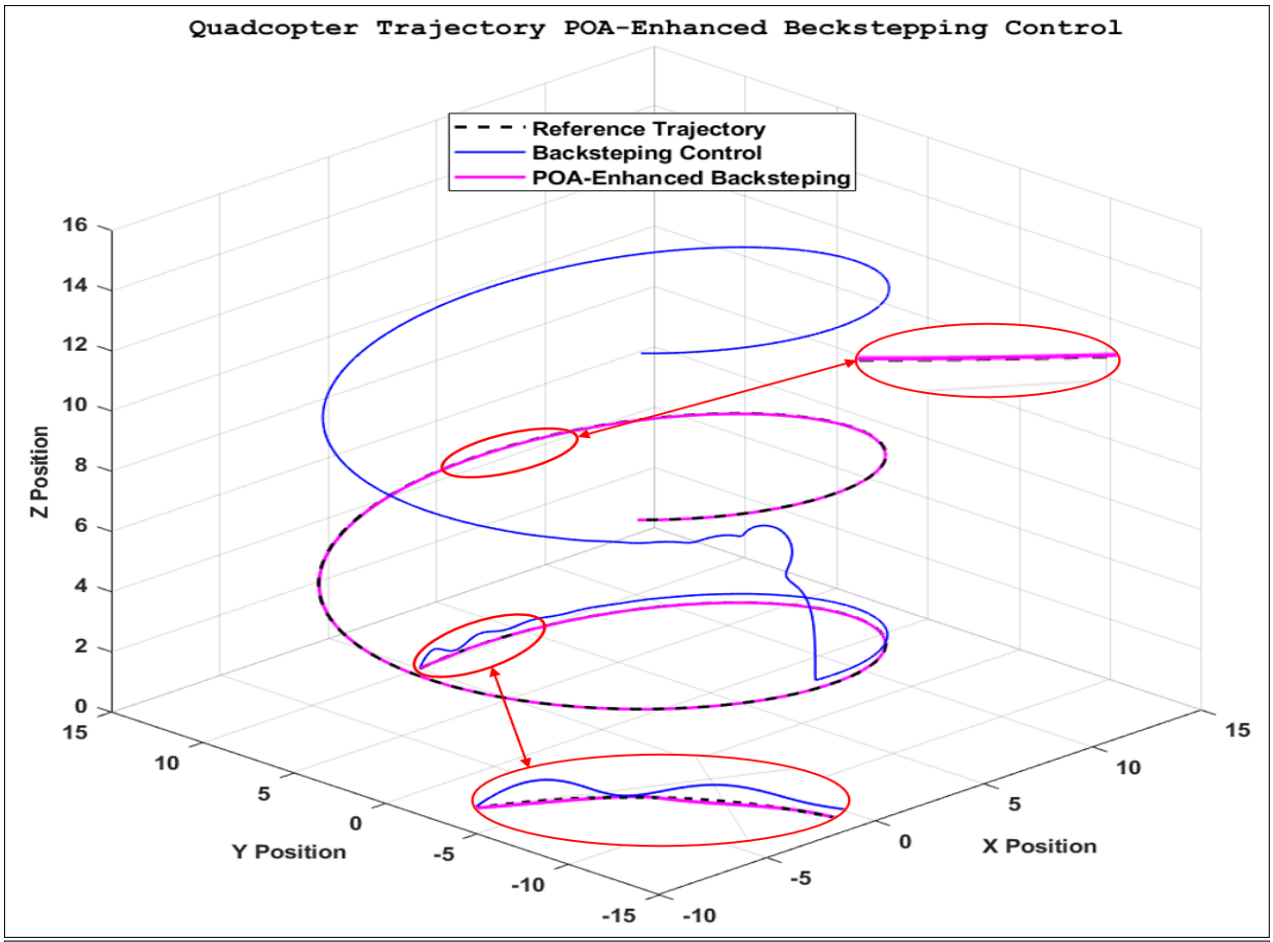

Quadcopter Trajectory -Beckstepping Control-
Reference Trajectory
Backsteping Trajectory
Z Position
Y Position
X Position

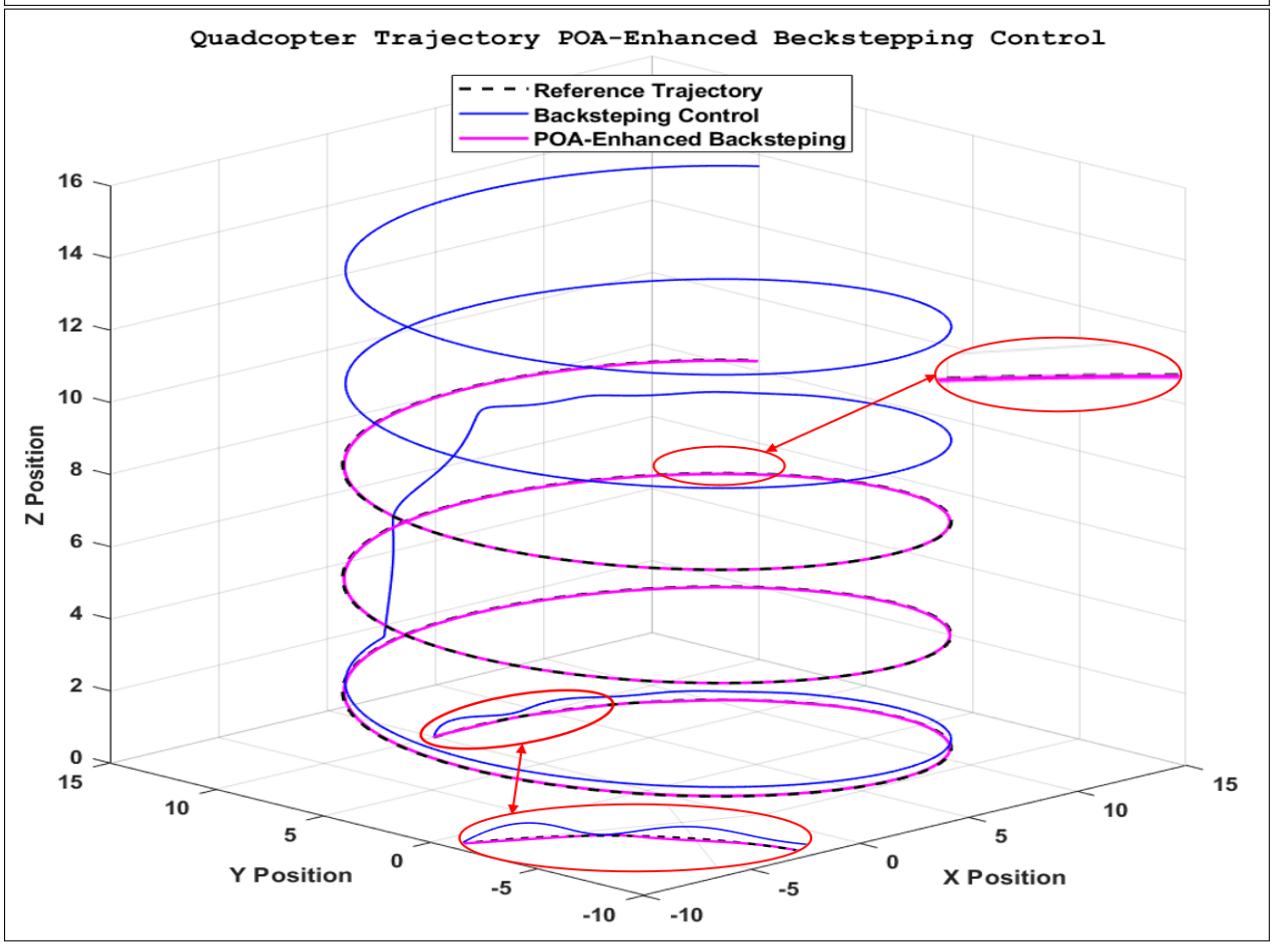

Quadcopter Trajectory GWO-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GWO-Enhanced Backsteping
Z Position
Y Position
X Position

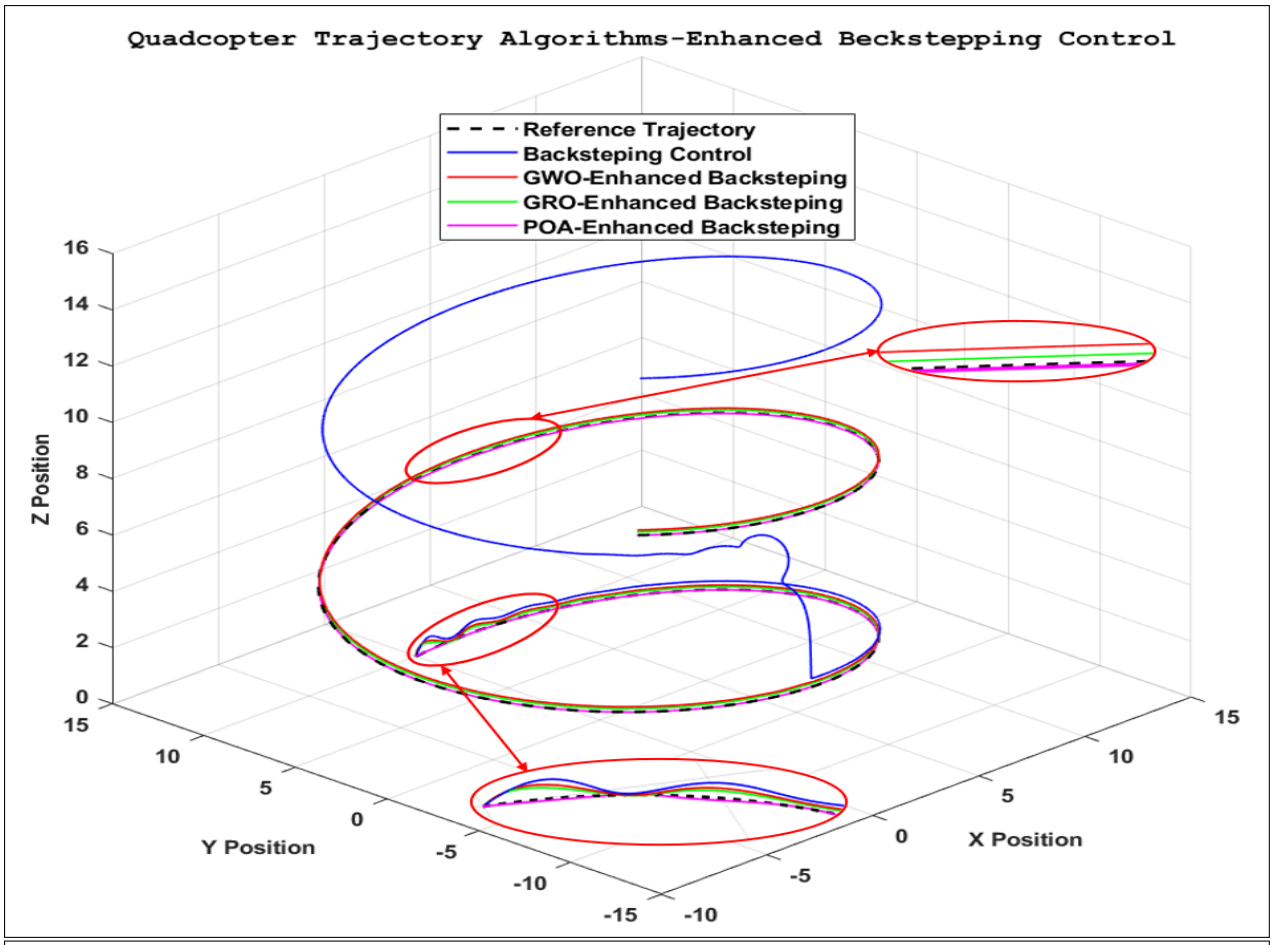

Quadcopter Trajectory GWO-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GWO-Enhanced Backsteping
Z Position
Y Position
X Position

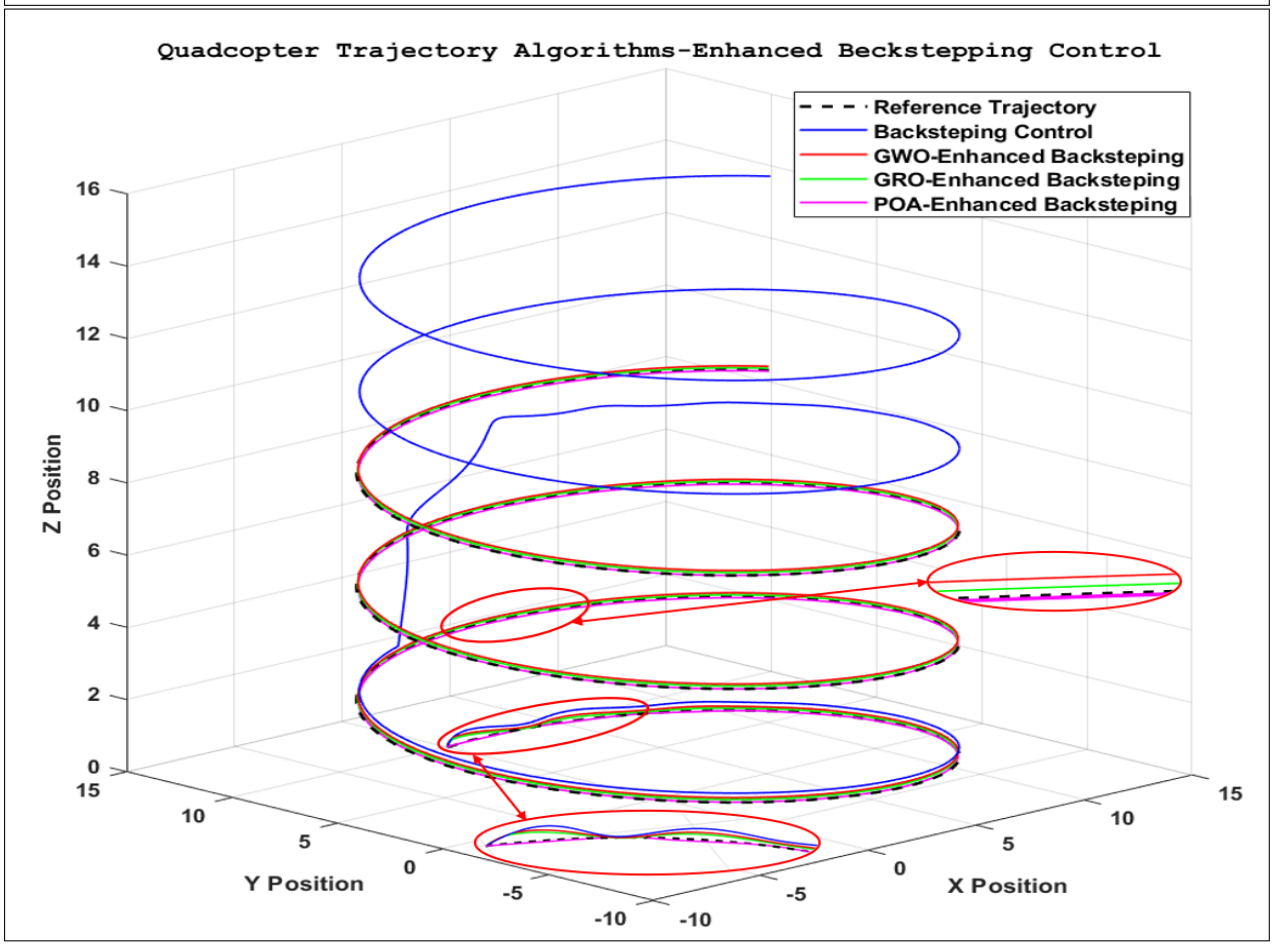

Quadcopter Trajectory GRO-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GRO-Enhanced Backsteping
Z Position
Y Position
X Position

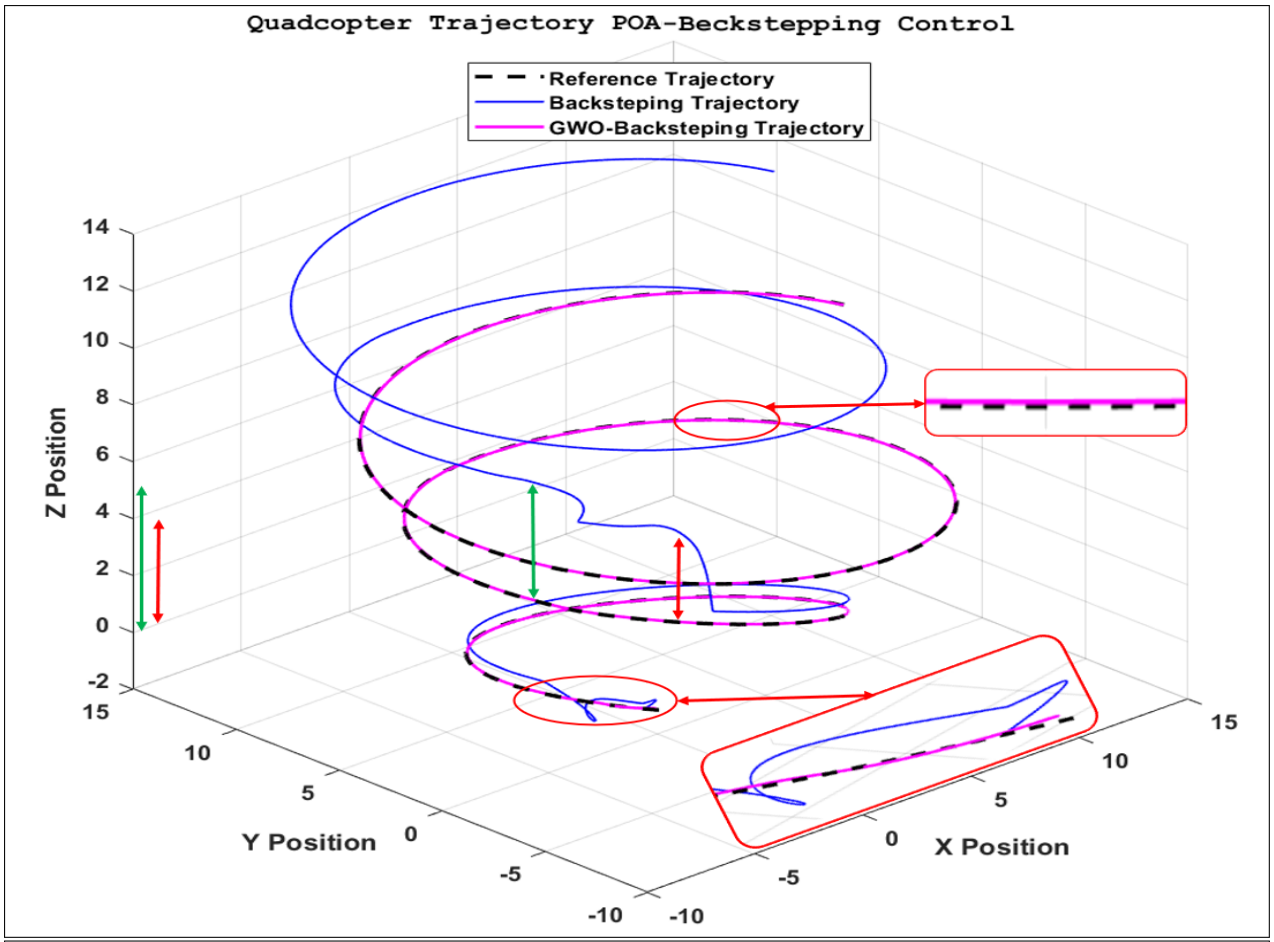

Quadcopter Trajectory GRO-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GRO-Enhanced Backsteping
Z Position
Y Position
X Position

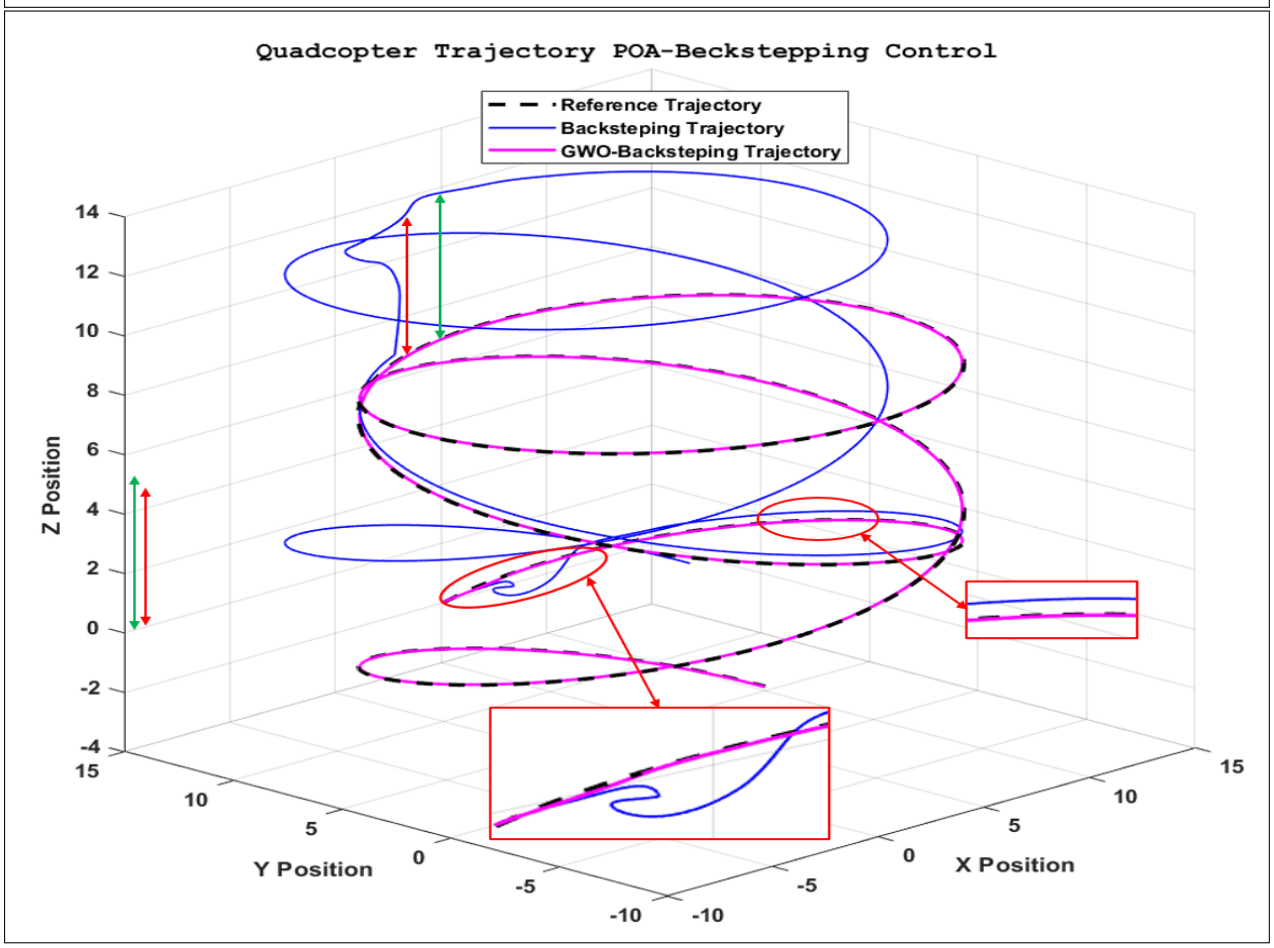

Quadcopter Trajectory POA-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
POA-Enhanced Backsteping
Z Position
Y Position
X Position

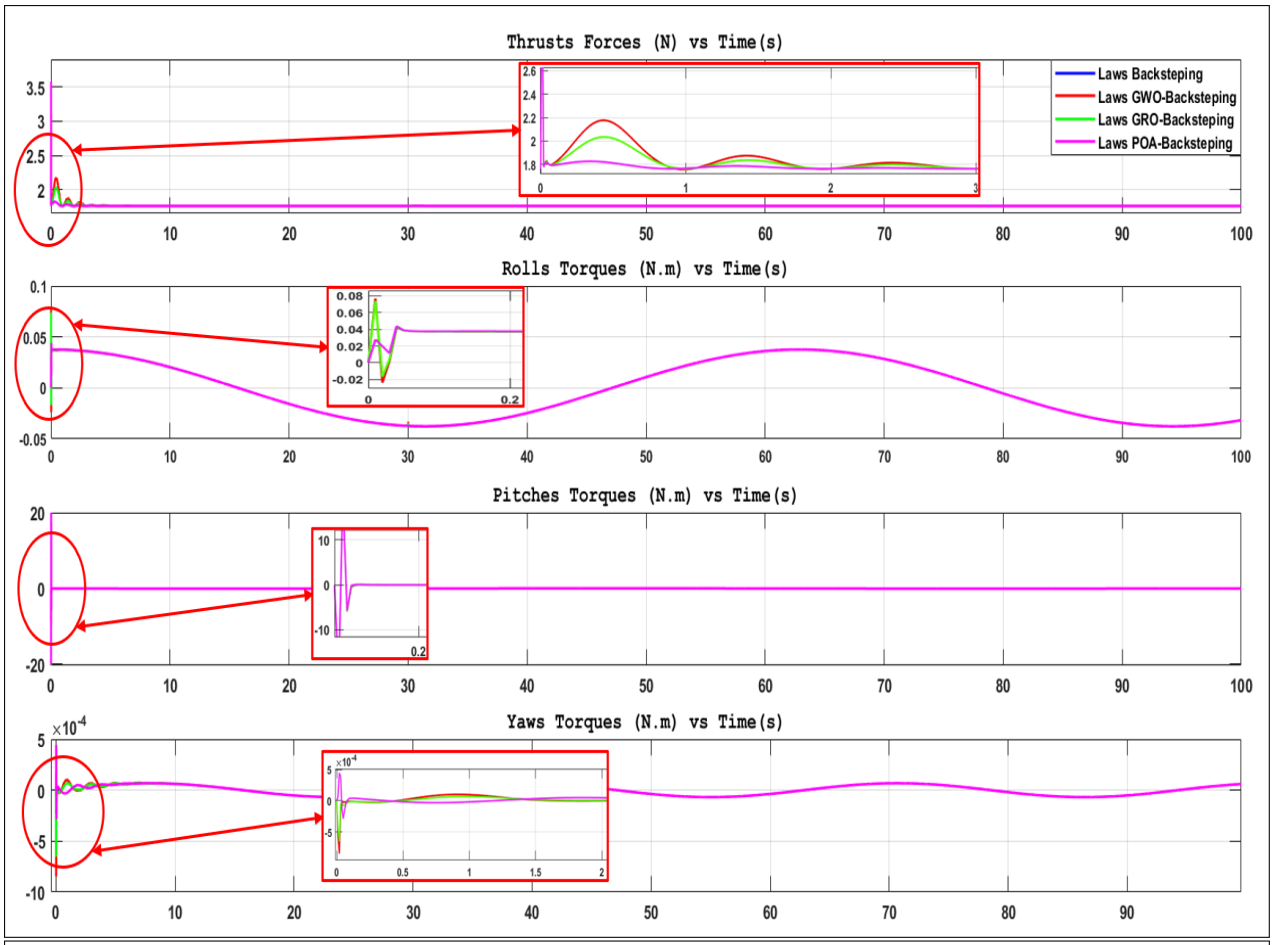

Quadcopter Trajectory POA-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
POA-Enhanced Backsteping
Z Position
Y Position
X Position

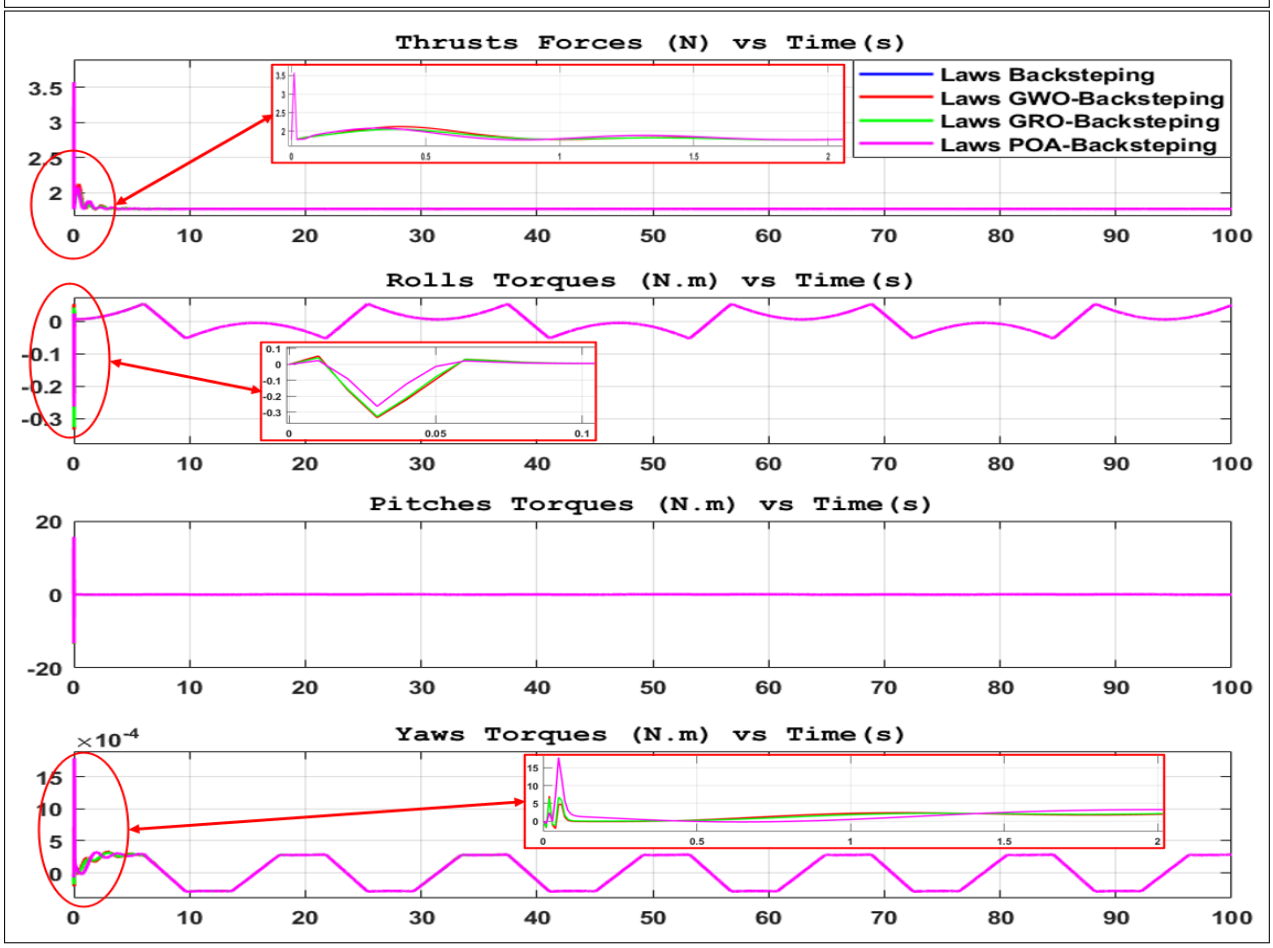

Quadcopter Trajectory Algorithms-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GWO-Enhanced Backsteping
GRO-Enhanced Backsteping
POA-Enhanced Backsteping
Z Position
Y Position
X Position

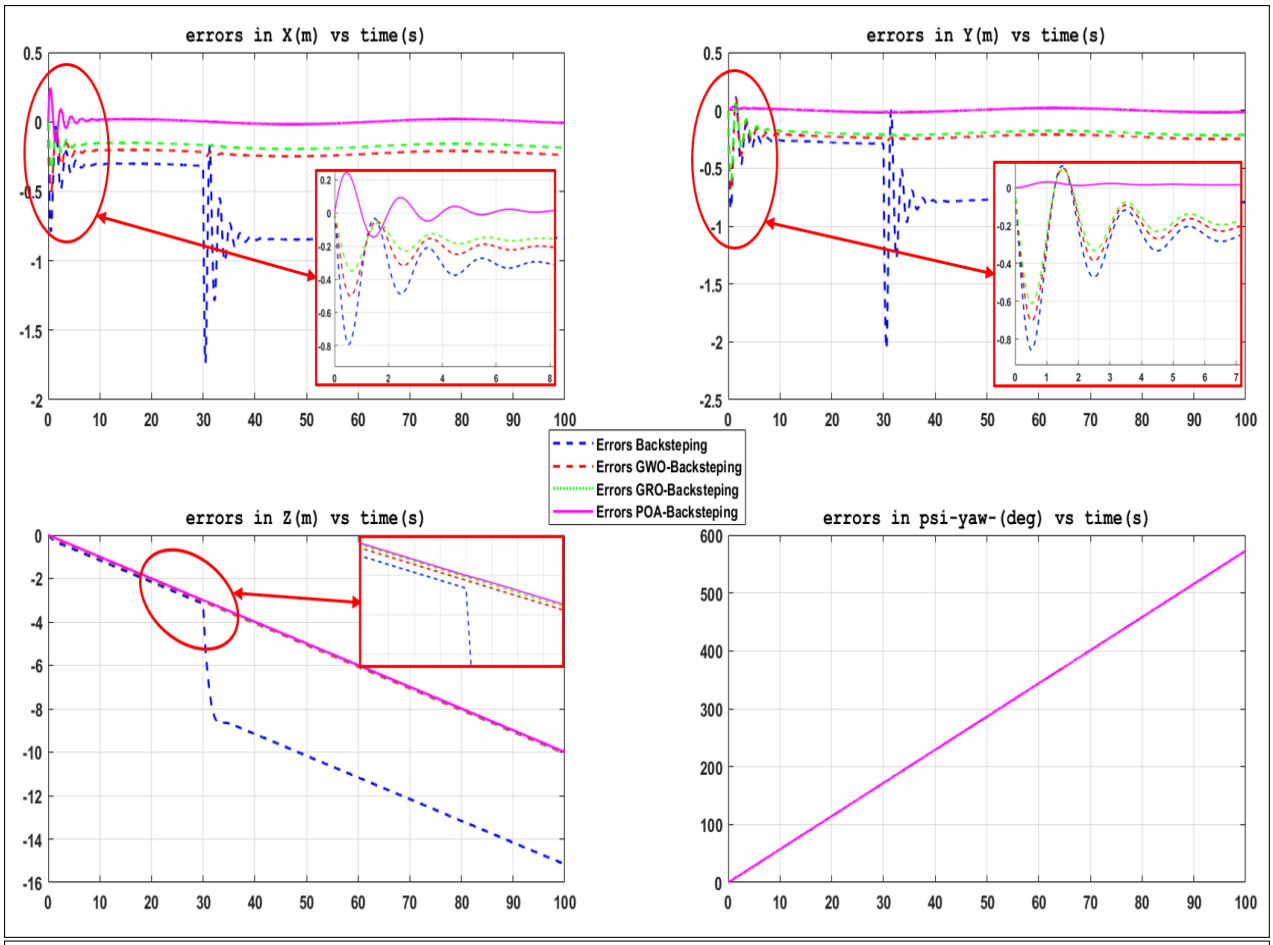

Quadcopter Trajectory Algorithms-Enhanced Beckstepping Control
Reference Trajectory
Backsteping Control
GWO-Enhanced Backsteping
GRO-Enhanced Backsteping
POA-Enhanced Backsteping
Z Position
Y Position
X Position

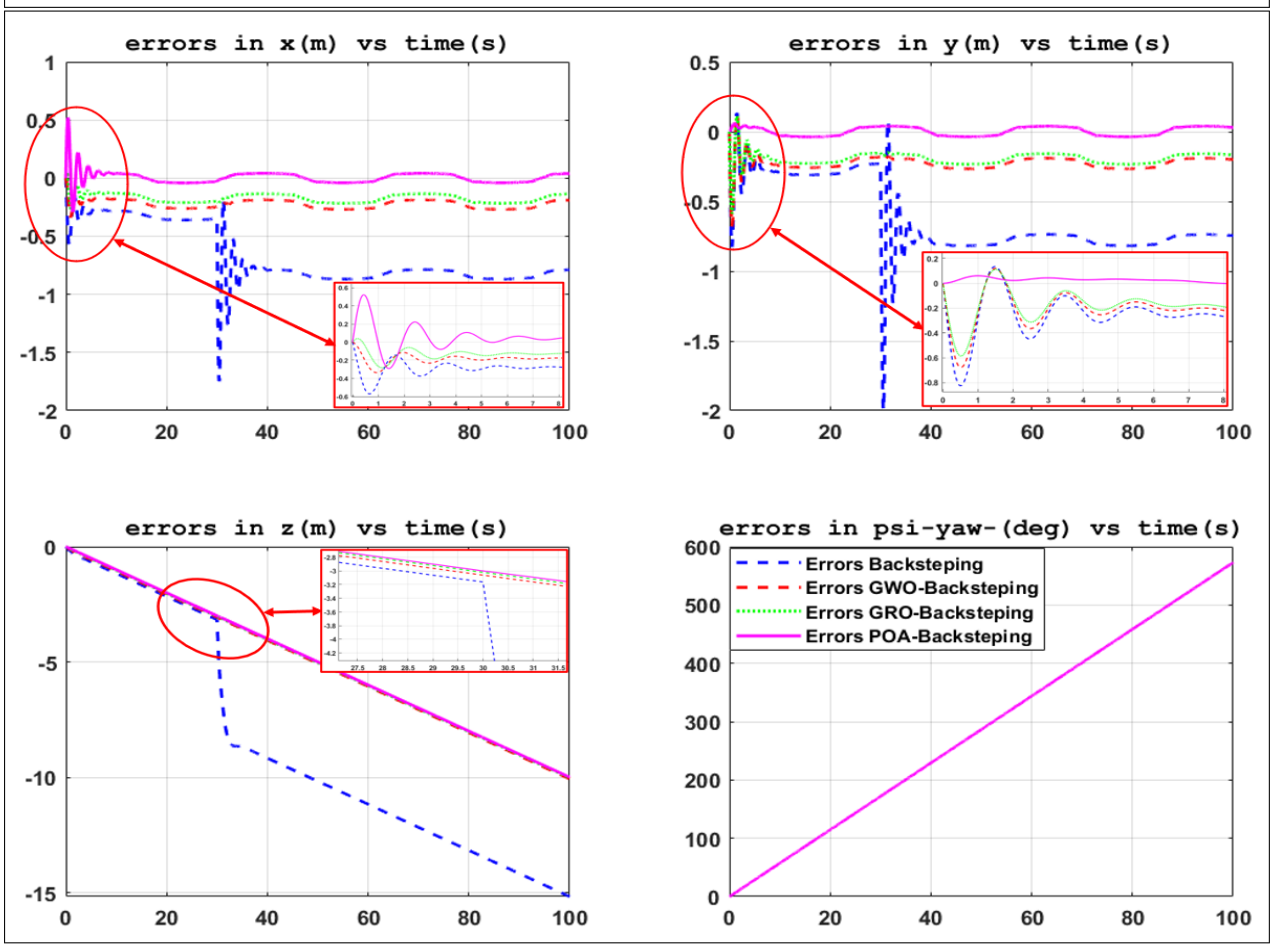

Quadcopter Trajectory POA-Beckstepping Control
Reference Trajectory
Backsteping Trajectory
GWO-Backsteping Trajectory
Z Position
Y Position
X Position

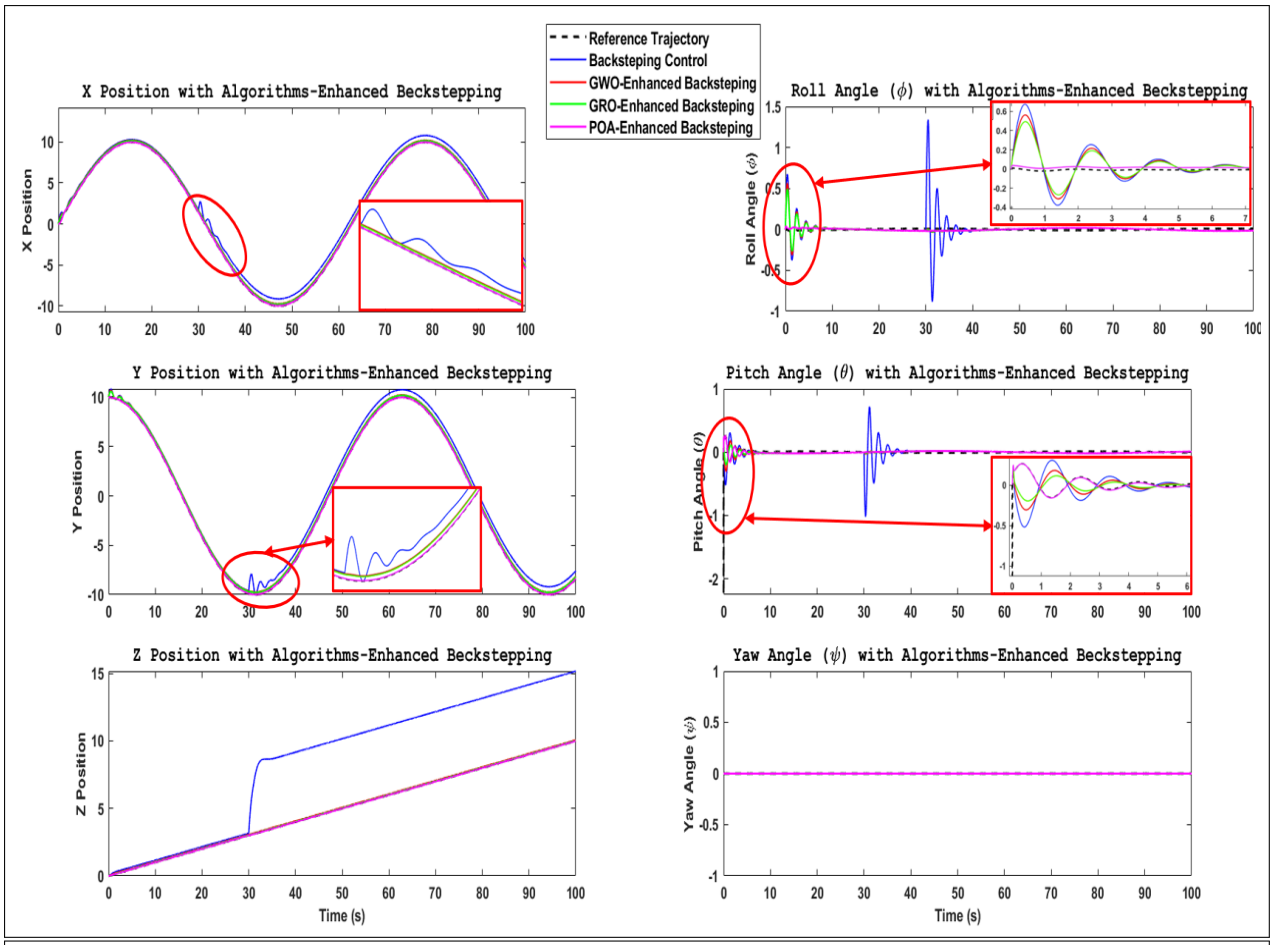

Quadcopter Trajectory POA-Beckstepping Control
Reference Trajectory
Backsteping Trajectory
GWO-Backsteping Trajectory
Z Position
Y Position
X Position

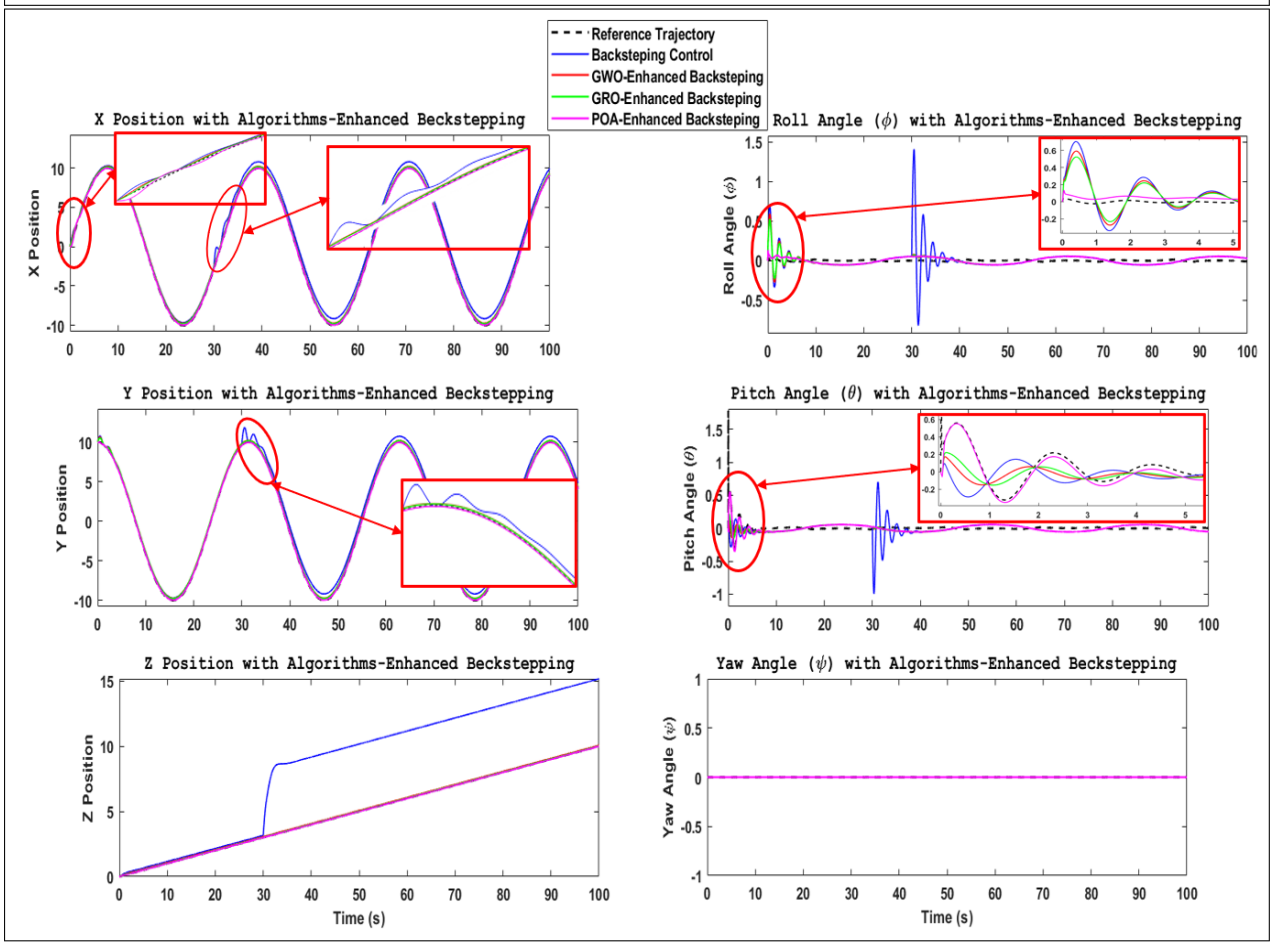

Thrusts Forces (N) vs Time(s)
Laws Backsteping
Laws GWO-Backsteping
Laws GRO-Backsteping
Laws POA-Backsteping
Rolls Torques (N.m) vs Time(s)
Pitches Torques (N.m) vs Time(s)
Yaws Torques (N.m) vs Time(s)

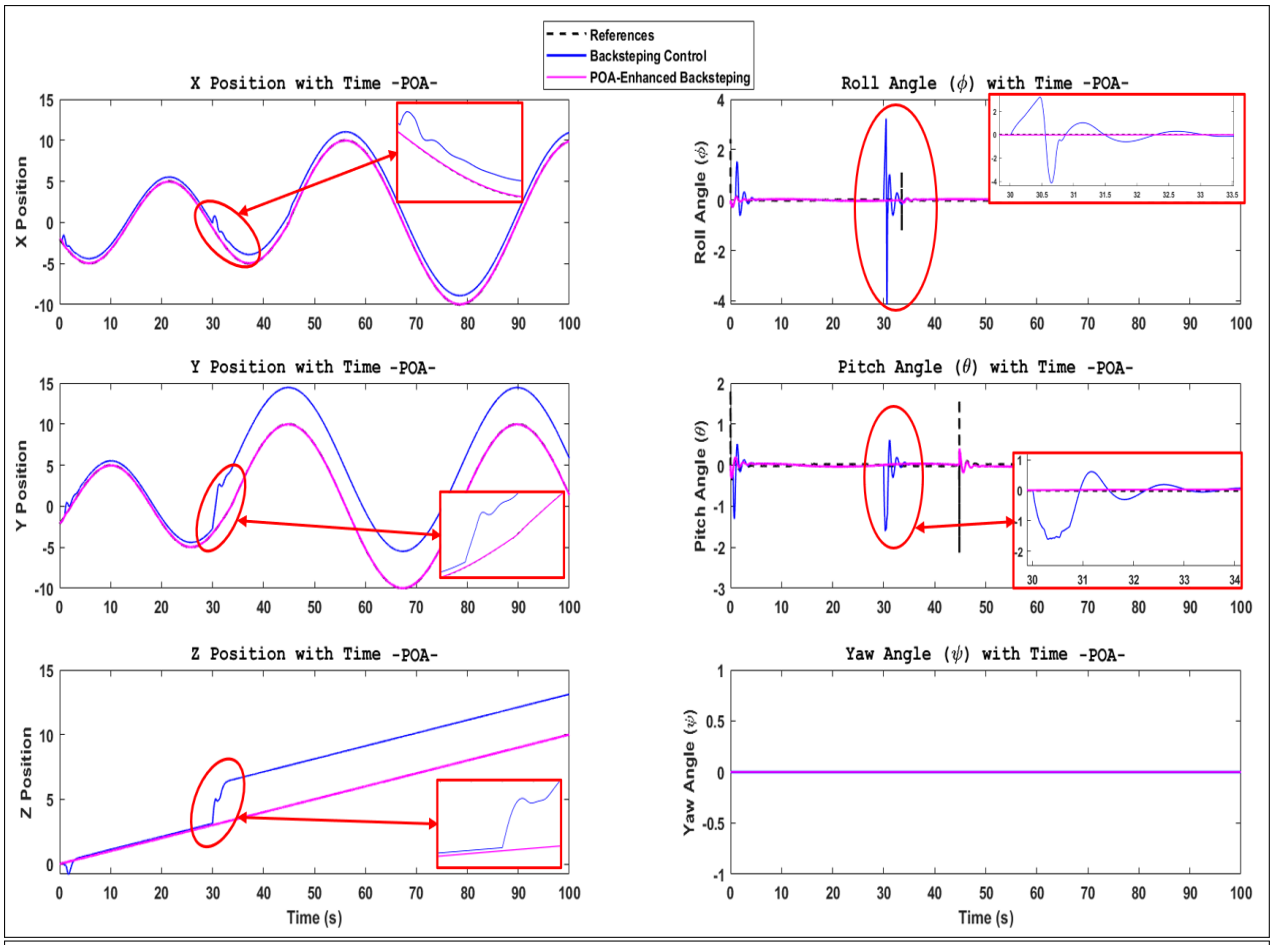

Thrusts Forces (N) vs Time(s)
Laws Backsteping
Laws GWO-Backsteping
Laws GRO-Backsteping
Laws POA-Backsteping
Rolls Torques (N.m) vs Time(s)
Pitches Torques (N.m) vs Time(s)
Yaws Torques (N.m) vs Time(s)

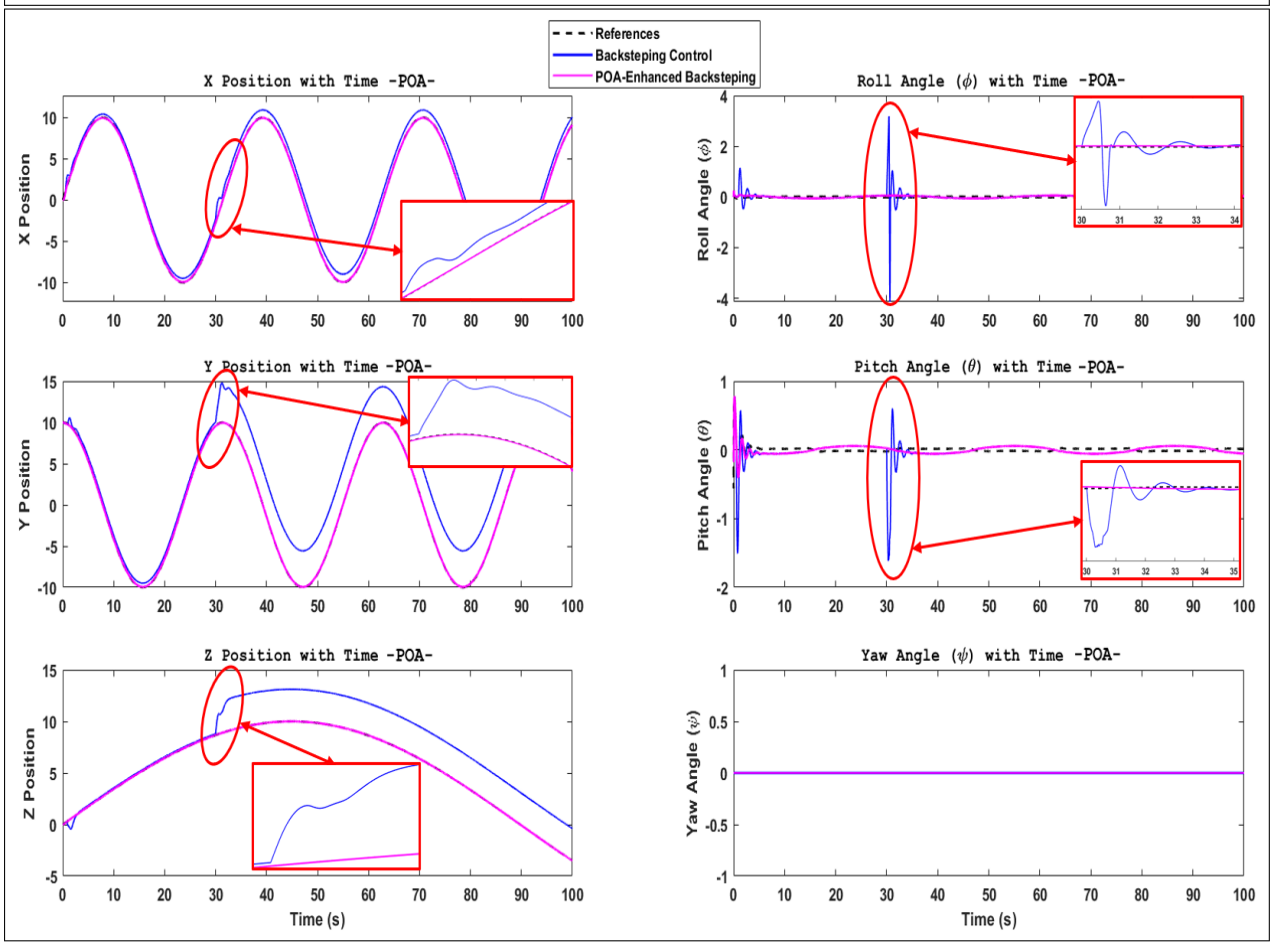

errors in X(m) vs time(s)
errors in Y(m) vs time(s)
errors in Z(m) vs time(s)
errors in psi-yaw-(deg) vs time(s)
Errors Backsteping
Errors GWO-Backsteping
Errors GRO-Backsteping
Errors POA-Backsteping


errors in x(m) vs time(s)
errors in y(m) vs time(s)
errors in z(m) vs time(s)
errors in psi-yaw-(deg) vs time(s)
Errors Backsteping
Errors GWO-Backsteping
Errors GRO-Backsteping
Errors POA-Backsteping


Reference Trajectory
Backsteping Control
GWO-Enhanced Backsteping
GRO-Enhanced Backsteping
POA-Enhanced Backsteping
X Position with Algorithms-Enhanced Beckstepping
X Position
Roll Angle (ϕ) with Algorithms-Enhanced Beckstepping
Roll Angle (ϕ)
Y Position with Algorithms-Enhanced Beckstepping
Y Position
Pitch Angle (θ) with Algorithms-Enhanced Beckstepping
Pitch Angle (θ)
Z Position with Algorithms-Enhanced Beckstepping
Z Position
Yaw Angle (ψ) with Algorithms-Enhanced Beckstepping
Yaw Angle (ψ)
Time (s)


Reference Trajectory
Backsteping Control
GWO-Enhanced Backsteping
GRO-Enhanced Backsteping
POA-Enhanced Backsteping
X Position with Algorithms-Enhanced Beckstepping
X Position
Roll Angle (ϕ) with Algorithms-Enhanced Beckstepping
Roll Angle (ϕ)
Y Position with Algorithms-Enhanced Beckstepping
Y Position
Pitch Angle (θ) with Algorithms-Enhanced Beckstepping
Pitch Angle (θ)
Z Position with Algorithms-Enhanced Beckstepping
Z Position
Yaw Angle (ψ) with Algorithms-Enhanced Beckstepping
Yaw Angle (ψ)
Time (s)


References
Backsteping Control
POA-Enhanced Backsteping
X Position with Time -POA-
Roll Angle (ϕ) with Time -POA-
Y Position with Time -POA-
Pitch Angle (θ) with Time -POA-
Z Position with Time -POA-
Yaw Angle (ψ) with Time -POA-
X Position
Roll Angle (ϕ)
Y Position
Pitch Angle (θ)
Z Position
Yaw Angle (ψ)
Time (s)


References
Backsteping Control
POA-Enhanced Backsteping
X Position with Time -POA-
Roll Angle (ϕ) with Time -POA-
Y Position with Time -POA-
Pitch Angle (θ) with Time -POA-
Z Position with Time -POA-
Yaw Angle (ψ) with Time -POA-
X Position
Roll Angle (ϕ)
Y Position
Pitch Angle (θ)
Z Position
Yaw Angle (ψ)
Time (s)