Zonotopes in robust drone flight control: an integrated framework for reachability analysis, real-time path planning and model predictive control

DELANSNAY GILLES

In fulfillment of the requirements for the degree of
Docteur en Sciences de l'Ingénieur et Technologie

Supervisor
PROF. VANDE WOUWER ALAIN

Université de Mons

Faculté Polytechnique
SECO lab

Mons, Belgique
2025

II

Members of the Jury

Prof. Verlinden Olivier (UMONS), President
Dr. De Cubber Geert (RMA)
Prof. Langlois Nicolas (ESIGELEC)
Prof. Garone Emanuele (ULB)
Dr. Dewasme Laurent (UMONS)
Prof. Vande Wouwer Alain (UMONS), Supervisor

Pour Jean, Joëlle et Alicja

Courage doesn't always roar. Sometimes it's the quiet voice at the end of the day saying, 'I will try again tomorrow.'
Mary Anne Radmacher

Acknowledgments

This thesis has been an incredible journey—an opportunity I could not refuse. It has allowed me to explore a field I am passionate about while opening doors to new perspectives and teaching experiences.

This research was funded by the University of Mons, where I had the privilege of serving as a teaching assistant in control theory. It has been a long and enriching journey, and I would like to express my gratitude to all those who contributed to this work and helped me grow both academically and personally.

First, I thank my supervisor, Professor Alain Vande Wouwer, for offering me this unique opportunity and for his trust. Throughout this thesis, he has actively contributing to our publications.

I would also like to thank Laurent Dewasme, who played a crucial role in one of the main topics of this thesis. His dedication, knowledge sharing, and technical insights have greatly influenced my work.

My deepest appreciation goes to Vincent Mœyart, a dedicated technical member of the lab. His companionship and unwavering support helped me navigate the toughest parts of this journey, keeping us on the right track with his encouragement.

A special mention to Camilo García Tenorio, my first research collaborator, who introduced me to control theory and academic research. He was the one who sparked my passion for the field, and our discussions on technical topics were always inspiring.

This thesis would not have been possible without the love and support of those closest to me, who provided a nurturing and encouraging environment throughout this journey.

A big shoutout to my amazing friends: Florent, Antoine, Julien, Lorenzo, Margaux, Serge, and Adeline. You guys rock, thanks for being there for me, whether you see these words or not!

I am deeply grateful to my mother, Joëlle, for her unwavering support and the love she has given me every day. Even when challenges arose, she stood by my side, pushing me forward and dedicating her time to me—an incredible gift.

I also want to thank my life partner and love, Alicja, who has been there for me long before this thesis began. Her joy, warmth, and unwavering belief in me have been a constant source of motivation. She has always been a breath of fresh air, reminding me of my goals and giving me the strength to persevere.

Finally, I dedicate a part of this work to my late father, Jean. Though the worst happened during this journey, I know he always, loved me, believed in me, supported me, and took pride in my achievements. I am certain he would have been proud of this milestone, and I am proud to honor his memory through this work.

Abstract

Over the past decade, multirotor aircraft have gained significant attention across various industries due to their ability to hover and maneuver in three-dimensional space. These capabilities make them invaluable for numerous applications, driving substantial advancements in unmanned aerial vehicles (UAVs) and contributing to their widespread adoption.

This work focuses on enhancing the flight robustness of small and medium UAVs, an aspect that is well-established in conventional aerospace operations but often overlooked in low-computation platforms. To address this challenge, we develop a multi-layered control structure that integrates low-level motor control with high-level path planning and collision avoidance.

At the core of our approach, we employ analytical feedback linearization to introduce critical linearity to the system, enabling the use of computationally efficient control techniques. A key contribution of this work is the utilization of zonotopes, a set-based representation, to perform reachability analysis. This method allows us to study system dynamics and determine a safe flight envelope, ensuring that any maneuver can be reversed to maintain operational safety.

To integrate this analysis into the control framework, we explore two upper-layer strategies: reference governors and their extension into tube model predictive control. These approaches enhance robustness against disturbances, such as wind perturbations, ensuring stable flight under challenging conditions.

Furthermore, we enhance existing path-planning algorithms by leveraging zonotopes to efficiently navigate cluttered environments. By constructing an expansive search tree over the available space while accounting for obstacles, our method enables reliable motion planning. Finally, we extend our framework to multi-agent UAV systems by partitioning the

fleet into smaller groups that communicate through distributed model predictive control.

Through experiments, we validate the effectiveness of our multi-layered approach across various real-world scenarios.

Résumé

Au cours de la dernière décennie, les drones multirotors ont connu un essor considérable dans de nombreux secteurs grâce à leur capacité à stationner en vol et à évoluer avec précision dans un espace tridimensionnel. Ces atouts en font des outils incontournables pour diverses applications, favorisant des avancées majeures dans le domaine des véhicules aériens autonomes (UAV) et accélérant leur adoption à grande échelle.

Ce travail vise à renforcer la robustesse en vol des UAV de petite et moyenne taille, un enjeu bien maîtrisé dans l'aéronautique traditionnelle, mais souvent négligé sur des plateformes à faible puissance de calcul. Pour répondre à ce défi, nous proposons une architecture de contrôle multicouche, allant de la gestion des moteurs à bas niveau jusqu'à la planification de trajectoire avancée et un système d'évitement des collisions.

Au cœur de notre approche, nous utilisons une technique de linéarisation analytique pour simplifier le comportement dynamique du drone et permettre l'application de méthodes de contrôle efficaces sur le plan computationnel. Une contribution clé de ce travail réside dans l'exploitation des zonotopes, une représentation par ensembles, afin de réaliser une analyse d'atteignabilité (reachability analysis). Cette approche permet de modéliser la dynamique du système et de définir une enveloppe de vol sécurisée, garantissant que toute manœuvre entreprise peut être corrigée afin de préserver la sécurité opérationnelle.

Pour intégrer cette analyse au sein du contrôle, nous explorons deux stratégies : le reference governor et son extension, le contrôle prédictif par tube (tube model predictive control). Ces méthodes renforcent la robustesse du drone face aux perturbations extérieures, comme les rafales de vent, et assurent un vol stable même dans des conditions difficiles.

Par ailleurs, nous améliorons les algorithmes de planification de trajectoire existants en utilisant les zonotopes pour naviguer plus efficace-

ment dans des environnements encombrés. En construisant un arbre de recherche optimisé tenant compte des obstacles, notre méthode permet une planification de mouvement fiable et réactive. Enfin, nous étendons ce cadre aux systèmes multi-drones en divisant la flotte en sous-groupes capables de collaborer via un contrôle prédictif distribué.

L'efficacité de cette approche multicouche est validée à travers une série d'expériences en conditions réelles, démontrant son potentiel pour améliorer la sécurité et les performances des UAV autonomes.

Contents

List of Figures

List of Tables

Nomenclature

$\cdot_x$	First component of a vector
$\cdot_y$	Second component of a vector
$\cdot_z$	Third component of a vector
α	Normalized weight in optimization problem
β_k	Zonotope factor
$\bigcap$	Intersection
$\bigcup$	Union
$\boldsymbol{\nu}$	Nominal input vector
$\boldsymbol{\omega}_b$	Vector of angular velocities in the body-frame **(rad/s)**
$\boldsymbol{\tau}_i$	Lever arm moment produced by the i^{th} motor **(N·m)**
$\boldsymbol{\xi}$	Nominal state vector
$\mathbf{A_c}$	Linear constraint matrix
$\mathbf{A_T}$	Transition matrix
$\mathbf{A}_d$	Adjacency matrix
$\mathbf{B}$	Input transition matrix
$\mathbf{b}$	Constraint offset
$\mathbf{c}$	Zonotope center

DL	Distribution between agent matrix
DP	Distance between agent matrix
D	Quadcopter geometry matrix
$\mathbf{e_{zb}}$	Unit vector along the the third axis in the body-frame
E	Exponent matrix
$\mathbf{E}_c$	Constraint exponent matrix
$\mathbf{E}_p$	Perturbations on state matrix
$\mathbf{e}_z$	Unit vector along the the third axis in the earth-frame
$\mathbf{f_i}$	Force produced by the i^{th} motor **(N)**
F	End of horizon state weight matrix
$\mathbf{f}_d$	Force disturbances **(N)**
$\mathbf{F}_p$	Perturbations on output matrix
G	Matrix of generator
g	Nonlinear function (in context)
$\mathbf{g}^(k)$	k^{th} generator
h	Nonlinear function
$\mathbf{H}_a$	Ancillary final horizon weight matrix
$\mathbf{H}_a$	Nominal final horizon weight matrix
I	Inertial matrix **(kg·m^2)**
J	Matrix link between the Euler angles and body-frame angular velocities
K	Controller/estimator gain
L	Agent distribution matrix

$\mathbf{m_i}$	Moment produced by the i^{th} motor **(N·m)**
$\mathbf{m}_d$	Moment disturbances **(N·m)**
$\mathbf{p}$	Vector of position in the earth-frame **(m)**
$\mathbf{Q}$	State weight matrix
$\mathbf{Q}_a$	Ancillary state weight matrix
$\mathbf{Q}_n$	Nominal state weight matrix
$\mathbf{R}$	Exponent matrix
$\mathbf{R}$	Input weight matrix
$\mathbf{R}$	Rotation matrix
$\mathbf{R}_a$	Ancillary input weight matrix
$\mathbf{R}_a$	Nominal input weight matrix
$\mathbf{SK}$	Skew matrix
$\mathbf{u}$	Input vector
$\mathbf{v}$	Vector of velocities in the earth-frame **(m/s)**
$\mathbf{V}_{wind}$	Vector of wind velocities in the earth-frame **(m/s)**
$\mathbf{w}$	Disturbances vector
$\mathbf{x}$	State vector
$\mathbf{y}$	Output vector/measurement vector
$\cdot$	Matrix-Zonotope multiplication - generator mapping
δ_{ij}	Binary variables designing membership of agent in a partition
δ_{ij}	Set of partition indices
γ	Angular velocity along the third axis **(rad/s)**
ι	Angular velocity along the second axis **(rad/s)**

χ	Vector of Euler angles **(rad)**
$\mathcal{CPZ}$	Constrained polynomial zonotope
$\mathcal{CZ}$	Constrained zonotope
$\mathcal{L}$	Invariant set
$\mathcal{PZ}$	Polynomial zonotope
$\mathcal{R}$	Reachable set
$\mathcal{R}_b$	Backward reachable set
$\mathcal{R}_f$	Forward reachable set
$\mathcal{SFE}$	Safe flight envelope
$\mathcal{U}$	Input constraint set
$\mathcal{V}$	Nominal input constraint set
$\mathcal{W}$	Disturbance set
$\mathcal{X}$	State constraint set
$\mathcal{Y}$	Output constraint set
$\mathcal{Z}$	Zonotope
ω_i	Rotor rotationnal speed **(rad/s)**
$\ominus$	Pontryagin difference
$\oplus$	Minkowski sum
$\otimes$	Linear mapping
ϕ	Pitch angle **(rad)**
ψ	Yaw angle **(rad)**
ρ	Air density **(kg/m^3)**
θ	Roll angle **(rad)**

$\times$	Cartesian product
Ξ	Nominal state constraint set
ζ	Angular velocity along the first axis **(rad/s)**
a	Acceleration/virtual input **($\mathbf{m/s^2}$)**
A_F	Drag force reference area **($\mathbf{m^2}$)**
A_M	Drag moment reference area **($\mathbf{m^2}$)**
$c(.)$	Cosinus function
C_q	Non-dimensional aerodynamic moment coefficient
C_t	Non-dimensional aerodynamic force coefficient
C_{Fd}	Non-dimensional drag force coefficient
C_{Md}	Non-dimensional drag moment coefficient
d	Number of disturbances
e	Number of outputs
g	Gravitational acceleration **($\mathbf{m/s^2}$)**
h_r	Rotor height **(m)**
k	Discrete increment value
l_a	Arm length **(m)**
m	Mass / Number of partitions **(kg)**
M_d	Disturbing moments **(N·m)**
N	Horizon
n	Number of states/system dimensions
n_c	Number of constraint
n_g	Number of zonotope generators

n_p	Number of polynomial factors in a zonotope
n_q	Number of constraint polynomial factors in a zonotope
p_x	Position along the first axis **(m)**
p_y	Position along the second axis **(m)**
p_z	Position along the third axis **(m)**
r	Number of inputs
r_r	Rotor radius **(m)**
$s(.)$	Sinus function
s_b	Area covered by the blades of a rotor **($\mathbf{m}^2$)**
t	Time **(s)**
t_h	Thrust **(N)**
v_x	Velocity along the first axis **(m/s)**
v_y	Velocity along the second axis **(m/s)**
v_z	Velocity along the third axis **(m/s)**
we	Weight in optimization problem

Chapter 1

Motivation and Contribution

1.1 Introduction

Unmanned Aerial Vehicles (UAVs), particularly multirotor aircraft like quadcopters, have emerged as transformative technologies over the past decade. Their inherent ability to hover, take off and land vertically, and navigate with high agility in three-dimensional space has unlocked a wide array of applications across numerous domains. These range from critical infrastructure inspection, aerial photography and videography, and precision agriculture to search and rescue operations, package delivery, and surveillance. This rapid proliferation is a testament to the versatility and increasing reliability of these platforms.

However, despite their widespread adoption and remarkable capabilities, the control of quadcopters presents significant challenges. These challenges stem primarily from their underactuated and highly nonlinear dynamics, susceptibility to external disturbances such as wind gusts, and the inherent need for safe and reliable operation, especially when operating in complex or uncertain environments. Ensuring robust performance, guaranteed safety, and efficient navigation while operating with limited computational resources, often found on small and medium-sized platforms, remains a critical area of research.

Traditional control approaches for quadcopters often rely on simplified linear models or require significant computational power for nonlinear model predictive control. While effective in ideal conditions, these methods can struggle to provide strong guarantees on system behavior when

faced with uncertainties or disturbances. Furthermore, ensuring that the vehicle always operates within safe limits and can avoid collisions, particularly in dynamic or cluttered spaces, requires sophisticated techniques that can reason about the set of possible future states the system might occupy.

This thesis addresses these challenges by proposing a comprehensive, multi-layered control framework for quadcopters that leverages the power of set-based methods, specifically zonotopes, to enhance robustness and planning capabilities. Zonotopes, as convex hulls of a finite number of vectors, offer a computationally efficient way to represent and propagate sets of states, making them particularly well-suited for real-time applications on platforms with limited processing power. By integrating zonotope-based reachability analysis throughout the control architecture, this work aims to bridge the gap between theoretical guarantees provided by set-theoretic methods and their practical implementation in complex robotic systems.

The proposed control structure is designed as a hierarchy, starting from a fundamental layer responsible for stabilizing the nonlinear dynamics and extending to higher layers that handle robust trajectory tracking, collision avoidance, and cooperative control for multi-agent systems. At the core of the low-level stabilization, we employ feedback linearization. This technique transforms the nonlinear quadcopter dynamics into an equivalent linear system through a change of variables and input transformation. This linearization simplifies the subsequent controller design significantly, allowing the use of well-established linear control techniques and reducing the computational burden compared to directly controlling the nonlinear system. Critically, we demonstrate how zonotopic representations can be integrated even at this foundational level to reason about the effects of uncertainties or unmodeled dynamics on the linearized system.

Building upon the stabilized dynamics, the thesis develops strategies for robust trajectory tracking and constraint satisfaction. A key contribution is the integration of zonotope-based reachability analysis into the control loop. Reachability analysis computes the set of all possible states that a system can reach from a given initial state set, considering bounded disturbances and uncertainties. By representing state and input uncertainties as zonotopes, we can efficiently compute reachable sets. This analysis provides crucial safety guarantees by allowing us to check for potential constraint violations or collisions before they occur.

We explore two specific methodologies for incorporating this reach-

ability information into a robust control framework. Firstly, we investigate the use of Reference Governors (RGs). An RG is a supervisory control scheme that modifies the reference signal sent to a base controller to ensure that state and input constraints are never violated, even in the presence of disturbances. By using zonotopes to represent the predicted future states of the system and its safe operating region, the RG can quickly determine if the current reference will lead to a constraint violation and, if so, compute the necessary modification to the reference.

Secondly, we extend this concept to the framework of Tube Model Predictive Control (Tube-MPC). Tube-MPC is a powerful technique for robust control where the controller aims to keep the actual system trajectory within a predefined tube around a nominal, disturbance-free trajectory. The size and shape of this tube are determined by the reachable sets of the system under disturbances. By using zonotopes to compute these reachable tubes, the Tube-MPC can optimize control inputs over a receding horizon while guaranteeing that the actual system state remains within a tube around the undisturbed trajectory, thus ensuring robustness against disturbances and strict adherence to constraints.In the linear case, this tube can be quantified, but its design remains complex for nonlinear systems.

Beyond robust tracking, safe navigation in environments containing obstacles is paramount for practical UAV deployment. This thesis enhances existing path planning algorithms by integrating zonotope-based collision checking. Specifically, we develop a Rapidly-exploring Random Tree (RRT) algorithm that utilizes zonotopes to represent the vehicle's swept volume or its predicted future occupied space. By growing the RRT using zonotopic state representations and performing collision checks between these zonotopes and obstacle representations, the planner can efficiently find collision-free paths in cluttered environments while implicitly accounting for the vehicle's size and potential uncertainties. This approach offers more robust collision avoidance guarantees compared to point-based planning methods.

Finally, the thesis extends the developed framework to the challenging problem of coordinating multiple UAVs. By partitioning the fleet into smaller groups, we explore strategies for distributed control where each group or agent computes its control actions based on local information and limited communication with neighboring agents.

In summary, this thesis presents a novel and comprehensive approach

to quadcopter control that integrates set-based methods, particularly zonotopes and reachability analysis, into a multi-layered control architecture. By combining feedback linearization for base stability with zonotope enhanced robust control (Reference Governors and Tube MPC) for disturbance rejection and constraint satisfaction, and a zonotope-based RRT for safe path planning, this work provides a powerful framework for robust and safe autonomous flight of quadcopters in complex and uncertain environments. The effectiveness of the developed techniques is validated through simulations and experiments.

1.2 A review of the literature

1.2.1 Introduction to unmanned aerial vehicles

Unmanned Aerial Vehicles (UAVs), and particularly multi-copters, have experienced an unprecedented surge in popularity and utilization across diverse sectors in recent years. This rapid development can be attributed to a confluence of technological advancements, economic factors, and evolving societal needs.

One key element has been the significant improvement in electronics. Sophisticated sensors (e.g., Inertial Measurement Units (IMUs), GPS modules), powerful microprocessors, and reliable communication systems have become significantly smaller, lighter, and more affordable. These advances have enabled the construction of compact and capable flight controllers and navigation systems suitable for small-scale UAVs [1]. Moreover, improvements in battery energy density and power output—especially with Lithium Polymer (LiPo) batteries—have drastically increased the flight endurance and operational capabilities of multi-copters, which is crucial for enabling longer missions and carrying heavier payloads.

The accessibility of open-source software and hardware platforms (e.g., ArduPilot, PX4) has further democratized access to advanced flight control technologies. This openness has fostered rapid innovation and prototyping within the community, significantly reducing acquisition and operational costs compared to manned aircraft or fixed-wing drones [2]. In particular, the vertical take-off and landing (VTOL) capability of multi-copters eliminates the need for runways or specialized launch infrastructure, while their maneuverability and ability to hover make them ideally

suited for operations in confined or complex environments.

UAVs benefit not only from these core design advancements but also from their versatility across various domains. For instance, in agriculture, multi-copters equipped with multispectral or infrared cameras play a crucial role in crop monitoring, disease detection, and irrigation optimization [3]. In the construction industry, UAVs facilitate the inspection of sites, industries, and power lines—minimizing manual inspection needs and ensuring compliance with design models [4]. In cinematography, they provide unique aerial perspectives, and emerging drone-based delivery systems are being explored for last-mile logistics in urban and remote areas [5].

However, while these technological improvements have transformed UAV design and expanded their applications, they also pose new challenges. The development of potent algorithms is essential to effectively manage the diverse applications and uncertainties inherent in UAV operations. Simplistic models may overlook critical dynamics, whereas more intricate models require detailed aerodynamic knowledge and come at a higher cost. Moreover, UAVs must operate reliably under varying environmental conditions—such as wind, rain, and extreme temperatures—which can significantly affect their behavior. These factors underscore the need for robust control methodologies to ensure flight stability and robustness.

Overall, the evolution in UAV technology not only enables a broader range of applications but also sets the stage for advanced research in robust control systems—a key focus of this thesis.

1.2.2 Nonlinear control in multi-rotor aircraft

In the last decade, a wide range of control techniques have been employed for the control of multicopters. For instance, [6] demonstrates a backstepping control method for enabling accurate trajectory tracking in quadcopters, while [7] implements feedback linearization by creating a linearized closed-loop model. Lyapunov-based control approaches are also explored, as highlighted in [8], and model predictive control is readily applied to multi-rotor aircraft in [9]. This list is not exhaustive; numerous other control methods have been reviewed in [10], which provides a comprehensive overview of the various techniques applied to multicopters.

These control methods can broadly be categorized into two groups. The first group comprises analytical techniques, which, despite their math-

ematical complexity in deriving input equations, offer the advantage of fast computation during flight. In contrast, the second group consists of optimization-based controllers that systematically accommodate various constraints, albeit with higher computational demands.

In addition to these conventional approaches, robust control techniques have been extensively employed to manage the inherent disturbances encountered by multicopters. For example, [11] implements an H_∞ control strategy on a load-carrying quadcopter, aiming to minimize the worst-case impact of uncertainties on the system's performance. Similarly, [12] utilizes Linear Parameter-Varying (LPV) control to accommodate multiple dynamic behaviors, while [13] employs sliding mode control to steer the system toward desired behaviors. Handling disturbances through a different paradigm, [14] adopts a neuro-fuzzy control approach. Furthermore, [15] demonstrates a tube model predictive control that confines the multirotor's trajectory within a predefined tube to ensure robust performance.

These methods have been validated experimentally in both simulation and real-flight tests, confirming their effectiveness in enhancing the stability and performance of multicopters under varying conditions.

1.2.3 Reachability

The concept of reachability analysis aims to ascertain whether a system state can be reached within a finite time, given specific initial conditions. This concept is highly useful in formal verification, controller synthesis, and related applications. In the comprehensive review presented in [16], a state-of-the-art examination within the set propagation framework is established. In this framework, initial conditions are embedded in a set representation, which is then used in propagation algorithms to derive the solution of the dynamical equation. The resulting solution encompasses all states that can be reached from the specified initial conditions.

For example, in [17], reachability analysis is applied to evaluate the operational performance of autonomous cars, demonstrating its relevance in aerospace systems as well, as further illustrated in [18], where reachability plays a critical role in controller synthesis.

In addition to propagation methods, the choice of set representation directly influences the accuracy of the computed reachable set. Basic representations, such as intervals [19], while straightforward, tend to yield conservative results. Another widely recognized family of representations

involves ellipsoids [20], which offer the advantage of a convex structure that facilitates rapid optimization processes, albeit at the cost of conservatism in complex systems.

Convex polytopes—whether through H-representation (using linear inequalities) or V-representation (using vertices) [21, 22]—also introduce conservatism, similar to ellipsoids. A recently introduced representation, known as constraint zonotopes, adopts a generator-based approach (Z-representation) inspired by zonotopes, a category of symmetric convex polytopes. This approach, introduced in [23], enables set-theory operations such as Minkowski addition, linear mapping, and intersection, thereby significantly enhancing reachability algorithms and reducing their computational complexity to polynomial time.

Notably, recent extensions to nonconvex polytopes, namely polynomial zonotopes and constrained polynomial zonotopes, have been introduced in [24, 25]. These extensions allow for more accurate reachability computations for nonlinear systems, showcasing the ongoing evolution of reachability analysis methodologies. For instance, in [26], reachable sets are computed to ensure that the states of nonlinear systems under control inputs remain within acceptable operational bounds and avoid critical phenomena.

Overall, these advancements in reachability analysis contribute significantly to the development of robust control strategies for multicopters by providing comprehensive insights into the system dynamics and performance limits.

1.2.4 Operational Reliability in Aerospace

One of the foremost concerns in aviation is the potential for any aerial vehicle to pose significant risks to itself, its potential occupants, and surrounding structures. Consequently, extensive research on possible failure modes, threats, and operational challenges has been undertaken within the aircraft domain [27, 28, 29]. While UAVs currently do not transport people and typically operate with lighter payloads, their flights at lower altitudes—coupled with an increasing number of operations and beyond-line-of-sight missions—introduce notable risk factors.

It is crucial to recognize that risk mitigation measures must also be developed for UAVs, given their expanding presence in the airspace. Although UAVs may not carry passengers at present, their operations con-

tribute to airspace congestion and potential hazards. Drawing inspiration from the protocols established in conventional aviation becomes imperative in managing the unique challenges posed by unmanned aerial vehicles. This proactive approach ensures that operational standards are not only maintained but also adapted to address these challenges.

For instance, innovative approaches have been explored to enhance protective measures in UAV operations. In studies such as [30, 31], tools known as safe flight envelopes are meticulously computed using Hamilton-Jacobi methods. These safe flight envelopes play a critical role in ensuring the revertibility of input actions and are seamlessly integrated into control strategies to ensure that a UAV operates within prescribed boundaries. Another avenue for determining safe flight envelopes involves the application of Monte Carlo techniques, as demonstrated in [32] for quadcopters. Although this technique proves accurate with a large sample size, it demands substantial computational power.

Moreover, advancements in reachability analysis using set propagation techniques based on zonotopes have been employed for risk management considerations. In [33], this method is applied to Kalman filtering, ensuring a robust tube of trajectory. Additionally, [34] explores the impact of icing conditions on system performance by evaluating reachable sets under specific environmental constraints. These diverse methodologies contribute to the proactive establishment of measures that address a spectrum of potential challenges in UAV operations.

1.3 Objectives

The primary objectives of this research are to develop and validate a robust control architecture for unmanned aerial vehicles, with a strong focus on integrating zonotopic tools and real-time implementation, from high-level planing to low-level motor speeds control. To this end, the research aims to achieve the following:

1. **Design a Robust Control Architecture:** Develop a control framework for UAVs that integrates a incremental nonlinear dynamic inversion for feedback linearization. This cascaded control approach will facilitate the transition from high-level position control down to low-level motor speed commands, ensuring robustness against uncertainties.

2. **Perform Zonotopic Reachability Analysis:** Utilize zonotopic set representations to perform reachability analysis on multi-rotors. This objective is critical for developing a systematic approach to handling uncertainties.

3. **Develop Robust-Constrained Control Methods:** Implement a strategy for enforcing constraints through optimization to ensure that the UAV's trajectory remains within defined operational limits.

4. **Path planning and collision Avoidance:** Develop a path planning algorithm enhanced with zonotope sampling for effective region exploration and collision avoidance with an extension to the approach for multi-agent scenarios.

5. **Prototyping and Experimental validations:** To validate the proposed control strategies through real-world experiments on a suite of hardware platforms, each chosen to rigorously test different layers of the control stack, from the lab to the field.

   (a) **Initial Low-Level Validation (Parrot Mambo):** Our initial exploration into low-level control, such as the Incremental Nonlinear Dynamic Inversion (INDI) strategy, was conducted on the Parrot Mambo minidrone. Its accessible firmware provided a straightforward entry point for testing custom controllers that require direct access to the motors.

   (b) **High-Level Outdoor Demonstration (Hexacopter):** Our custom built hexacopter, equipped with a Pixhawk 4 autopilot running the PX4 flight stack, was used as a high-level autonomy demonstrator. For this complex platform, we intentionally retained the underlying closed-loop controllers. This platform was crucial for validating our supervisory algorithms like Tube-MPC in a real-world scale, where the goal is to provide robust guidance to a competent, off-the-shelf autopilot.

   (c) **High-Fidelity Indoor Validation (Crazyflie):** While the Mambo was suitable for initial proofs-of-concept, its limited robustness led us to adopt the Crazyflie 2.1 platform for more rigorous indoor testing. Integrated within a motion capture system providing high-precision, ground-truth state feedback. This setup was essential for easier and faster validation.

These objectives collectively aim to bridge the gap between advanced theoretical control design and its practical deployment in UAV systems, thereby contributing to the broader adoption of robust, and efficient autonomous aerial vehicles.

1.4 Contributions

The primary contributions of this doctoral thesis lie in the development and experimental validation of advanced robust control technoiques for Unmanned Aerial Vehicles (UAVs). These contributions are presented in a series of peer-reviewed publications and a book chapter, each building upon the previous one. Specifically, this work introduces novel applications of zonotopic reachability analysis to address key challenges in UAV autonomy, including robust control, safe navigation, and real-time path planning.

The main contributions can be summarized as follows:

Original Contributions

- **A Framework for Safe Control Using Zonotopic Reachability:** This thesis introduces a method for ensuring robust control of UAVs by integrating a high-fidelity feedback linearization scheme (INDI) with a constrained reference governor. The key innovation is the use of computationally efficient zonotopic reachability analysis to determine the maximum admissible set of states, thereby guaranteeing that the UAV always operates within safe constraints.

- **Experimental Validation of Zonotopic Tube-MPC:** We present experimental validation of a tube-based Model Predictive Control (MPC) strategy that uses zonotopic reachability for a multirotor aircraft. This contribution demonstrates the practical feasibility this approach in handling real-world disturbances and uncertainties.

- **Path Planning Algorithm:** This research proposes a novel extension of the Rapidly-exploring Random Trees (RRT) algorithm, known as Zonotopic RRT*. This new approach integrates zonotopic representations to provide robust, collision-free paths in uncertain environments. Unlike traditional RRT methods, this algorithm accounts

for the system's dynamics and uncertainties, ensuring that the planned trajectory is not only collision-free but also dynamically feasible and safe to execute.

Conference Papers

- **Zonotopic Reachability Analysis of Multirotor Aircraft**
  Delansnay Gilles; Vande Wouwer Alain; Harno G. Hendra et al. (2021, 40th Benelux Meeting on Systems and Control; 25th International Conference on System Theory, Control and Computing): This work introduces the novel use of zonotopic reachability analysis for multirotor systems, demonstrating its computational efficiency and accuracy for future state prediction compared to traditional Monte Carlo simulations.

- **Implementation and Tests of an INDI Control Strategy with the Parrot Mambo Minidrone**
  Delansnay Gilles; Vande Wouwer Alain (2022, 41st Benelux Meeting on Systems and Control; International Conference on Unmanned Aircraft Systems): This paper presents the first implementation of a robust feedback linearization method for the Parrot Mambo minidrone, validating its effectiveness in a practical, cascade control structure. It sets the foundation for integrating reachability-based constraints.

- **Reference Governor in the Zonotopic Framework Applied to a Quadrotor under an INDI Control Strategy**
  Delansnay Gilles; Vande Wouwer Alain (2023, 42nd Benelux Meeting on Systems and Control): This work integrates a reference governor with our zonotopic reachability framework, ensuring the safety of the quadrotor by keeping it within a predefined admissible set.

- **Experimental Validation of Zonotopic Tube-MPC Applied to a Hexacopter**
  Delansnay Gilles, Dewasme Laurent, Vande Wouwer Alain (2025, European Control Conference) : This paper presents an experimental validation of the proposed zonotopic tube-MPC method on a hexacopter, showcasing its ability to ensure robust, safe flight in real-world scenarios.

Journal Papers

- **Design of a Reference Governor in a Zonotopic Framework Applied to a Quadrotor under Feedback Linearization Control Strategies**
  Delansnay Gilles; Vande Wouwer Alain (2023, Journal of Intelligent and Robotic Systems) : This journal paper extends our conference work by providing a detailed theoretical and practical design of a zonotopic reference governor, with a specific focus on computing the maximum admissible sets for safe control.

- **Partitioning and Distributed Tube-MPC for UAV Swarms**
  Delansnay Gilles, Ocampo-Martinez Carlos, Vande Wouwer Alain (2025, Journal of Intelligent and Robotic Systems) *[Submitted]*: This submitted work introduces a method for handling multi-agent systems by combining graph partitioning with distributed tube-MPC, paving the way for scalable swarm control.

- **Real-Time Path Planning Using Zonotopic Extensions to Rapidly-Exploring Random Trees**
  Gilles Delansnay, Vande Wouwer Alain (2025, IEEE/CAA Journal of Automatica Sinica) *[Submitted]* : This paper presents a generalization and improvement of the RRT* algorithm, using zonotopes to guarantee robust and dynamically feasible paths for UAVs in real-time, uncertain environments.

Book Chapter

- **Tube-Based Nonlinear Model Predictive Control of Multirotor Aircraft Using a Polynomial Zonotopic Framework**
  Delansnay Gilles, Dewasme Laurent, Vande Wouwer Alain (2025, Nonlinear and Constrained Control - Applications, Synergies, Challenges and Opportunities) : This chapter provides a comprehensive overview of our simulation-based work on a complex nonlinear tube-MPC, detailing the use of constrained polynomial zonotopes to ensure safety and robust control.

1.5 Thesis Structure

This thesis is organized as follows:

- **Chapter 2:** Details the modeling of multirotor aircraft, with a focus on quadcopters.
- **Chapter 3:** Introduces fundamental concepts of control and state estimation for dynamical systems.
- **Chapter 4:** Discusses a nonlinear control strategy : feedback linearization application for multirotors.
- **Chapter 5:** Explores set representations—particularly zonotopes—and their application in reachability analysis.
- **Chapters 6 and 7:** Investigate the integration of zonotopic methods within optimization-based control frameworks to enhance robustness.
- **Chapter 8:** Presents high-level planning and collision avoidance strategies leveraging zonotopic constructs.
- **Chapter 9:** Examines multi-agent systems, focusing on decentralized control schemes and communication strategies.
- **Chapter 10:** Concludes with a summary of contributions and outlines future research directions.

Chapter 2

Multicopter Modeling

2.1 Introduction

In this chapter, we seek to understand how a multicopter works. We ask ourselves: how can such devices fly and maneuver? We introduce the states and the primary equations governing the system, based on Newton's laws, to model the dynamics of a multicopter. Although additional side effects exist in real systems, we will not consider them here.

2.2 Mechanistic Nonlinear Model

Most popular UAVs are multicopters which have the ability to move in any direction and rotate independently around each of its axes. It has six degrees of freedom and is equipped with an even number of rotors at the end of each arm (Fig. 2.1) in a symmetric disposition. The multicopter's physical parameters include the arm length, denoted as l_a, and the rotor height, h_r. Each rotor i (where $i = 1, 2, 3, 4$) operates at a speed w_i, which generates a force $\mathbf{f}_i$ and a drag moment $\mathbf{m}_i$. Figure 2.2 illustrates the two reference frames used to describe the multicopter's position and velocities. The inertial frame is attached to the Earth and is used to describe the drone's movement from an external perspective, whereas the body frame is attached to the multicopter's center of gravity and describes the measurements collected by the on-board sensors [35]. The multicopter's orientation in the inertial frame is given by the Euler angles $\boldsymbol{\chi} = (\phi, \theta, \psi)^T$, while its angular velocities in the body frame are denoted by $\boldsymbol{\omega}_b = (\zeta, \iota, \gamma)^T$.

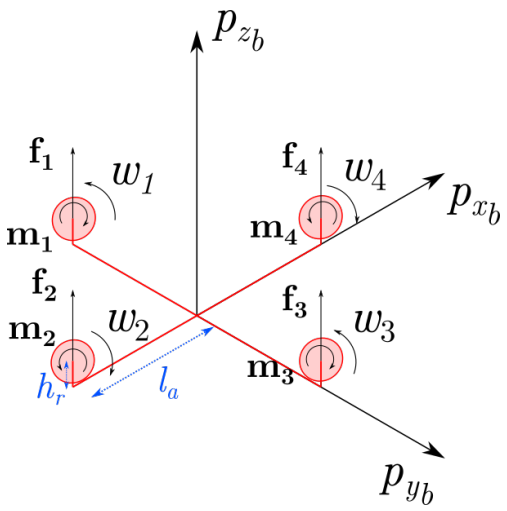


Figure 2.1: A multicopter with four rotors.

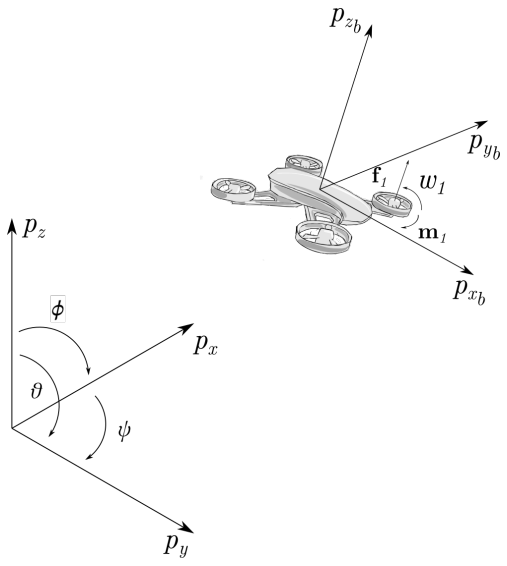


Figure 2.2: Inertial and body frames.

The forces and moments produced by each rotor can be computed in the body frame using aerodynamic models [36, 37]. The rotation and shape of the blades induce airflow through the rotor area, generating a force in the opposite direction. The force is directly proportional to the square of the blade velocity, while the resulting moment is caused by the drag on the blade. In addition, each rotor rotates in the opposite direction of their adjacent rotors to cancel out the natural drag:

$$\mathbf{f}_i = C_t \rho s_b r_r^2 w_i^2 \mathbf{e}_{z_b} \tag{2.1}$$

$$\mathbf{m}_i = C_q \rho s_b r_r^3 w_i |w_i| \mathbf{e}_{z_b}, \quad i = 1, 2, 3, \ldots, n \tag{2.2}$$

In these expressions, $\mathbf{e}_{z_b}$ is a unit vector along the p_{zb} axis of the body frame. C_t and C_q are non-dimensional aerodynamic coefficients for rotor thrust and torque, r_r is the rotor radius, ρ is the air density, s_b is the area covered by the blades, and $\mathbf{D}$ is a matrix that defines the positions of each rotor relative to the multicopter's center of mass, n is the even number of rotors.

Each rotor is placed away from the center of gravity, creating a moment due to the lever arm effect. Given the rotor position (see Fig. 2.1), its height h_r, and arm length l_a, the torque is given by:

$$\boldsymbol{\tau}_i = \mathbf{f}_i \times \mathbf{D} \tag{2.3}$$

where $\mathbf{D}$ encompass the position of all the rotors. The multicopter orientation is described by the Euler angles $\boldsymbol{\chi} = (\phi, \theta, \psi)^T$, and the angular velocities in the body frame (Fig. 2.2) are given by $\boldsymbol{\omega}_b = (\zeta, \iota, \gamma)^T$. The equations of motion are derived from Newton's law, and the relationship between the Euler angular rates and the body-frame angular velocities is obtained as in [35, 38, 39]:

$$\dot{\boldsymbol{\chi}} = \mathbf{J}(\boldsymbol{\chi}) \cdot \boldsymbol{\omega}_b \tag{2.4}$$

$$\dot{\boldsymbol{\omega}}_b = \mathbf{I}^{-1} (-\boldsymbol{\omega}_b \times \mathbf{I}\boldsymbol{\omega}_b) + \mathbf{I}^{-1} \sum_{i=1}^{n} [\mathbf{m}_i + \boldsymbol{\tau}_i] \tag{2.5}$$

where $\mathbf{I}$ is the multicopter's inertial matrix and $\mathbf{J}(\boldsymbol{\chi})$ is the projection matrix that relates the angular velocities in the body frame to the Euler angular rates:

$$\mathbf{J}(\boldsymbol{\chi}) = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix} \tag{2.6}$$

The positions and velocities of the multicopter are expressed in the inertial frame as $\mathbf{p} = (p_x, p_y, p_z)^T$ and $\mathbf{v} = (v_x, v_y, v_z)^T$. They are obtained directly from Newton's principle [35, 38, 39]:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{2.7}$$

$$\dot{\mathbf{v}} = -g \cdot \mathbf{e}_z + \mathbf{R}(\boldsymbol{\chi}) \frac{1}{m} \sum_{i=1}^{n} \mathbf{f}_i \tag{2.8}$$

where m is the mass of the multicopter and g is the gravitational acceleration. The forces $\mathbf{f}_i$, expressed in the body frame, must be projected into the inertial frame. The matrix $\mathbf{R}(\boldsymbol{\chi})$ is a rotation matrix that links both frames by rotating the body frame into the inertial frame [39]:

$$\begin{aligned} \mathbf{R}(\boldsymbol{\chi}) &= \mathbf{R}_z(\psi)\mathbf{R}_y(\phi)\mathbf{R}_x(\theta) \\ &= \begin{bmatrix} c(\theta)c(\psi) & c(\psi)s(\theta)s(\phi) - c(\phi)s(\psi) & s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta) \\ c(\theta)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix}. \end{aligned} \tag{2.9}$$

It is important to note that this representation exhibits a singularity at $\theta = \pi/2$. To avoid the singularity, an alternative coordinate system such as quaternions can be used. However, this work does not consider acrobatic maneuvers within the angular range of the singularity.

2.3 Modeling Simplifications and Justifications

The quadcopter dynamic model presented in this thesis is a deliberate simplification of the true physical system, tailored for the design and implementation of a real-time control system. While this model captures the dominant rigid-body dynamics, several higher-order aerodynamic and mechanical effects are consciously neglected. This section details these simplifications and provides the rationale for their omission.

2.3.1 Complex Aerodynamic Effects

The aerodynamic forces and moments generated by the propellers are highly complex and nonlinear. For our control-oriented model, we simplify these relationships, with the main assumptions being [40, 41]:

Thrust and Inflow Velocity The generated thrust of a propeller is not solely a function of its angular velocity (ω_i^2). In reality, it is also significantly affected by the axial airflow through the rotor disc, which includes the vehicle's own velocity v_z and the propeller's induced velocity (v_i). A more complete expression for thrust (T) is of the form $T = K(\omega - (v_i + v_z) \cdot k)^2$, where k is a constant related to the propeller pitch. This effect becomes prominent during fast vertical or forward flight. However, for near-hover conditions and maneuvers at low to moderate speeds, which constitute the primary operational envelope for this work, the simpler quadratic relationship provides a sufficiently accurate approximation.

Asymmetrical Blade Loading During forward flight, the advancing blade (moving in the direction of flight) and the retreating blade experience different relative airspeeds. Due to the nonlinear relationship between airspeed and drag, the resulting forces are not perfectly compensated between the two sides of the rotor disc. This asymmetry can induce flapping and generate moments on the airframe. These effects are most significant at high speeds and are therefore considered negligible for the scope of this research, which focuses on control strategies robust to such unmodeled dynamics.

Blade Flapping Beyond the asymmetrical loading, blade flexibility leads to flapping, particularly during translational movements. This phenomenon, as described by [36], can be modeled as:

$$\mathbf{f}_i = C_t \rho A r^2 w_i^2 \begin{bmatrix} -\sin(a_{is}) \\ \cos(a_{si})\sin(b_{si}) \\ \cos(b_{si})\sin(a_{si}) \end{bmatrix} \tag{2.10}$$

where a_{si} and b_{si} are the small longitudinal and lateral flapping angles. Although present, this effect is not directly considered in the control design. The flapping angles are small in magnitude and are themselves dependent

on vehicle velocities and additional aerodynamic parameters. Incorporating these dynamics would introduce substantial complexity for a marginal improvement in model fidelity within our flight envelope.

Motor Time Constant Each motor-propeller combination possesses a mechanical time constant, originating from the rotational inertia of the motor and propeller, which dictates the time required to change rotational speed. While these actuator dynamics can be modeled as a first-order system, they are neglected in our model. This simplification is justified by the very fast response of modern brushless DC motors, whose time constants are typically an order of magnitude smaller than the dominant rigid-body dynamics of the quadcopter. Omitting these fast actuator states is a common practice that reduces the overall order of the system, which is highly beneficial for the design of the controller and its implementation on an embedded system with limited computational resources.

2.4 Case Study: The Experimental DJI F550 Hexacopter

The theoretical model is parameterized using our custom-built experimental platform, shown in Figure 2.3. The system is built on a DJI F550 hexacopter frame and serves as the primary testbed for the control strategies developed in this thesis.

The core components include a Pixhawk 4 flight controller, a Raspberry Pi 5 for high-level processing, and a suite of sensors for navigation and perception. The physical parameters of the drone, which are critical for an accurate simulation, were measured and are listed in Table 2.1. These values for mass (m), arm length (l_a, derived from wingspan), and inertia ($\mathbf{I}$, which can be estimated or identified experimentally) are used to instantiate the dynamic model described in the previous section.

Figure 2.3: Experimental setup - Hexacopter equipped with a Pixhawk 4, GPS, batteries, camera, LiDAR, and a Raspberry Pi 5. The main frame used is a DJI F550.

Table 2.1: Multicopter Specifications

Parameter	**Value**
Type	Hexarotor
Mass (m)	3.37 kg
Height	37.5 cm
Wingspan	55 cm
Rotor radius	11.68 cm (Assuming 9.2 x 4.5 inch props)
Motor's speed constant (KV)	1000 KV

2.5 The Role and Strategy of Numerical Simulation

While physical experimentation provides the ultimate validation, numerical simulation was an indispensable tool throughout this research. It of-

fered a safe, repeatable and cost-effective environment for developing, testing and refining our control algorithms. Our simulation strategy was tailored to the specific goals of this thesis, leveraging state-of-the-art toolboxes to accelerate development while focusing on novel algorithmic applications.

Our primary simulation environment was MATLAB/Simulink, chosen for its powerful capabilities in control systems design and analysis. The specific tools we used were key to our progress:

- For low-level controller development, we utilized the official Simulink Support Package for Parrot Minidrones. This toolbox provided a seamless interface between the Simulink environment and the Parrot Mambo, enabling the direct deployment and testing of our control laws on the hardware from the design environment. This greatly accelerated our initial validation cycles.

- For the core of our set-based analysis, we leveraged the CORA (COntinuous Reachability Analyzer) toolbox for MATLAB. CORA is a state-of-the-art tool for formal verification and reachability analysis. Using this well-validated toolbox allowed us to focus on the novel application of zonotopes to drone control rather than spending effort re-implementing the complex underlying computational geometry and set-based arithmetic.

The typical research workflow involved designing and validating a concept in the MATLAB/Simulink/CORA ecosystem, followed by implementing the final algorithm in Python for integration with our other hardware platforms (the Crazyflie and the Pixhawk hexacopter). This transition required careful verification to ensure the Python code was a faithful representation of the validated simulation models.

A fair question is why a high-fidelity physics simulator like Gazebo was not a central tool. Gazebo excels at simulating the entire robotic system, including realistic sensors and environmental interactions. However, the primary focus of this thesis was on the mathematical guarantees of the control and planning algorithms themselves. The MATLAB/CORA environment was the most suitable for this purpose. Therefore, our use of Gazebo was confined to initial setup and code validation, rather than the rigorous, closed-loop dynamic simulations where the formal properties of our algorithms were the main subject of investigation. This deliberate

choice of tools allowed us to concentrate our efforts on the novel control and planning aspects of this research.

Chapter 3

Basics of Control

3.1 Introduction

Once a model of the system is established, we explore the fundamentals of controlling it. In this chapter, we first present how to control systems, then we show a general overview of autopilot architectures—the standard structures before focusing on a central example: the Parrot Mambo mini-drone. We discuss methods for control design [42, 43, 44] and state estimation [45, 46], and we briefly illustrate quantitative aspects such closed-loop dynamics, robustness, stability, and linearization [47].

3.2 Fundamental Concepts

Every real-world system can be described by a set of state variables, $\mathbf{x}$, whose evolution is governed by a dynamical equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}). \tag{3.1}$$

Often, the system also includes actuators that can be triggered by external actions. These control inputs are denoted by $\mathbf{u}$, and the dynamics become

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{3.2}$$

A primary goal in control engineering is not only to understand the system dynamics but also to influence them to achieve a desired objective. As

operators, we wish to manipulate the inputs so that the states reach specified target values (for example, a driver accelerates a car to reach a desired speed). However, manually adjusting inputs is both time-consuming and impractical for high-dimensional or highly nonlinear systems. To enable automatic control, we develop controllers and closed-loop control architectures. In doing so, we define a reference signal $\mathbf{r}$, representing the desired state, and design a control law, denoted by $\boldsymbol{\varphi}$, such that

$$\dot{\mathbf{u}} = \boldsymbol{\varphi}(\mathbf{x}, \mathbf{r}). \tag{3.3}$$

In practice, more common formulations directly specify $\mathbf{u}$ as a function of $\mathbf{x}$ and $\mathbf{r}$; however, dynamic controllers with internal states (or integrators) are also widely used in autopilot design.

3.3 Control of a Single-Input Single-Output System

Controlling a nonlinear system can be complex and may require advanced mathematical tools. Nevertheless, a decent control strategy can often be achieved using simple methods. Consider a single-input, single-output (SISO) system described by

$$\dot{x} = f(x, u). \tag{3.4}$$

A straightforward approach is to select an operating point (x_0, u_0)—the nominal condition around which the system is expected to operate—and then linearize the dynamics around this point. The nonlinear function is approximated by its first-order Taylor expansion:

$$f(x, u) \approx f(x_0, u_0) + \frac{\partial f}{\partial x}\Big|_0 \Delta x + \frac{\partial f}{\partial u}\Big|_0 \Delta u, \tag{3.5}$$

$$\Delta f \approx \frac{\partial f}{\partial x}\Big|_0 \Delta x + \frac{\partial f}{\partial u}\Big|_0 \Delta u, \tag{3.6}$$

where $\Delta x = x - x_0$, $\Delta u = u - u_0$, and the derivatives are evaluated at the operating point. This yields the linearized dynamics:

$$\Delta \dot{x} = A_T \Delta x + B \Delta u, \tag{3.7}$$

with

$$A_T = \frac{\partial f}{\partial x}\Big|_0, \quad B = \frac{\partial f}{\partial u}\Big|_0.$$

The eigenvalue of A_T determine the stability of the system. In continuous time, if the eigenvalue has a negative real parts, the system is stable; if any have positive real parts, the system is unstable. For simplicity, we omit the Δ notation in subsequent discussions while remembering that the linear model is valid near the operating point.

A well-known and widely used controller for SISO systems is the Proportional Integral Derivative (PID) controller:

$$u = K_p e + K_i \int_0^t e \, \mathrm{d}t + K_d \frac{\mathrm{d}e}{\mathrm{d}t}, \tag{3.8}$$

where the error is defined as $e = r - x$. The proportional term drives the error toward zero, the integral term eliminates any steady-state error, and the derivative term improves the transient response by damping the system. The design of the gains is a key factor influencing the entire dynamics of the closed-loop system. Various techniques can be used for tuning, ranging from trial-and-error and empirical methods such as Ziegler-Nichols to analytical approaches like Bode or Black-Nichols analysis, as well as optimization-based methods.

Such a control approach is effective for linear systems; however, in nonlinear systems, it requires the system to remain close to the operating point. If this condition is not met, performance degradation occurs, robustness decreases, and the risk of instability arises, making control guarantees difficult to uphold.

3.4 Estimation of a Single-Input Single-Output System

Although state estimation is not the primary focus of this thesis, it is an essential component of a complete control scheme. In many practical systems, not all state variables are directly measurable. Instead, sensors provide measurements y that are functions of the state:

$$y = h(x). \tag{3.9}$$

For example, accelerometers and gyrometers provide information about a drone's motion but not the full state.

A standard approach to estimating the state is to use an observer such as a Kalman filter. For a linearized system, the dynamics and measurement model are

$$\dot{x} = A_T x + Bu, \tag{3.10}$$
$$y = Cx, \tag{3.11}$$

and the corresponding observer is given by

$$\dot{\hat{x}} = A_T \hat{x} + Bu + K_o (y - C\hat{x}), \tag{3.12}$$

where $\hat{x}$ denotes the estimated state and K_o is the observer gain, typically computed from a Riccati equation. With an appropriate choice of K_o, the estimate $\hat{x}$ converges to the true state x.

As with control, linearization affects the performance of state estimation, with accuracy deteriorating as the system deviates further from the operating point. This can lead to major issues, as incorrect state estimates result in incorrect control inputs, further degrading the overall control performance. For further reading on dynamical systems, control, and state estimation, we refer the reader to [48, 49, 50].

3.5 General Structure of Autopilots

Autopilots are widely used to achieve safe and reliable control of complex systems such as drones. In typical autopilot architectures, the overall control structure is divided into several layers:

- **Inner loop:** Often dedicated to stabilizing the vehicle's attitude (roll, pitch, yaw) by quickly rejecting disturbances. A common inner-loop design uses a high-bandwidth controller based on a linearized model of the attitude dynamics.

- **Outer loop:** Controls the overall position or velocity of the drone. The outer loop typically has lower bandwidth and relies on the inner-loop controller to provide a stable platform. The outer loop is often design by neglecting the inner loop control assuming some order of magnitude in time constant differences between the two loops.

- **State Estimation:** Since not all states are directly measurable, sensor fusion (often via a Kalman filter) is employed to estimate the state vector $\mathbf{x}$ from available measurements $\mathbf{y}$. These estimated states are then used in the control laws. The bandwidth of the state estimation algorithm is a critical design parameter, as it dictates how quickly the filter can track changes in the system state, which in turn affects the performance of the control loops. As a general rule, the state estimator must be faster than the control loops' bandwidth.

The design of these control layers involves questions of robustness, how the system behaves under disturbances, and stability, ensuring the closed-loop system converges to desired values. Stability and robustness can be ensured through two main approaches. The first, more rigorous but complex, involves analytical study by formally proving stability and robustness conditions through mathematical analysis. However, this approach is often overlooked in open-source applications due to the advanced mathematical knowledge required and the significant gap between theoretical models and real-world implementation. The second, more practical but less rigorous, relies on numerical simulations, where various scenarios, system limits, and behaviors are tested to evaluate performance.

In many applications the system is designed to operate close to a nominal equilibrium, making such linear approximations effective.

3.6 Application: The Parrot Mambo Minidrone

We now apply these basic principles of control and state estimation to a more complex, multi-input multi-output (MIMO) system: the Parrot Mambo minidrone. Although the drone has multiple inputs and outputs, its control design can be simplified by decoupling it into several single-input single-output (SISO) subsystems. This is made possible by the inherent structure of the vehicle dynamics.

3.6.1 System Overview

The Parrot Mambo minidrone is a small UAV (see Fig. 3.1) that weighs 63 g and measures 18×18 cm. A complete list of the Parrot Mambo parameters is provided in Table 3.1. It embeds a microprocessor, an iner-

tial measurement unit (IMU) that provides linear acceleration and angular speed measurements, an ultrasonic sensor for obstacle detection, and a pressure sensor for altitude evaluation. A ground-looking camera is used to estimate the relative coordinates via optical flow. The Parrot Mambo minidrone can be directly controlled using a dedicated smartphone application. Additionally, Mathworks, in collaboration with Parrot, has developed a dynamic simulator that incorporates a six-degree-of-freedom model, the blade flapping effect, and simulated sensors. The simulator comes with a cascade PID flight controller, a state estimator (including the bias), and a visualization tool (see Fig. 3.2). This user-friendly tool can be used for educational purposes and rapid controller prototyping, and it will be employed in the following to design and test an INDI flight control structure. In this context, it is assumed that the states are provided; a full explanation of how the Parrot Mambo obtains these states through Kalman filtering is provided.

Parameter (Symbol)	**Value (Units)**
Mass (m)	0.063 kg
Inertia along x (I_x)	5.829×10^{-5} kgm^2
Inertia along y (I_y)	7.169×10^{-5} kgm^2
Inertia along z (I_z)	1.0×10^{-4} kgm^2
Aerodynamic coefficient (Thrust) (C_t)	0.0107 (dimensionless)
Aerodynamic coefficient (Torque) (C_q)	7.8264×10^{-4} (dimensionless)
Air density (ρ)	1.293 kg/m^3
Surface covered by the blades (s_b)	0.0034 m^2
Rotor radius (r_r)	0.0330 m
Arm length (l)	0.0624 m
Rotor height (h_r)	0.0159 m
PSD of white noise (IMU x-axis)	0.2183 m/s^2
PSD of white noise (IMU y-axis)	0.1864 m/s^2
PSD of white noise (IMU z-axis)	0.3725 m/s^2

Table 3.1: Parameters of the MathWorks simulator for the Parrot Mambo.

Figure 3.1: Parrot Mambo minidrone

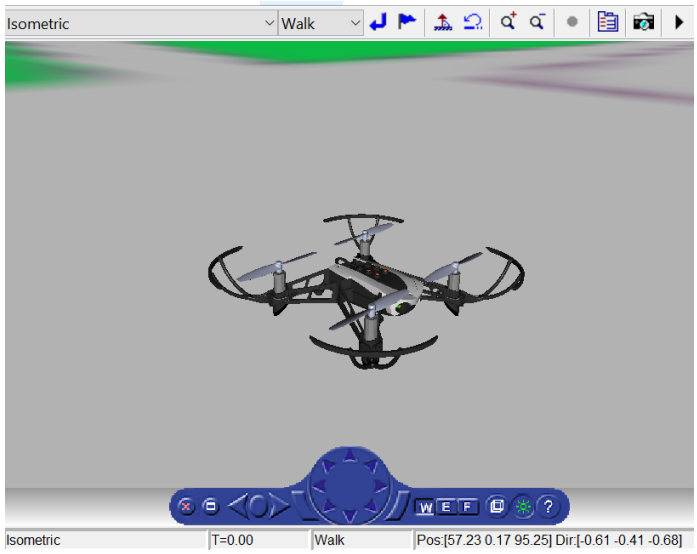


Figure 3.2: The Parrot Mambo minidrone visualization box allows the user to observe the simulated flight in real time.

3.6.2 Cascade PID Control Structure

To control the Parrot Mambo minidrone, a cascade PID control structure is implemented. The control inputs are redefined as

$$\mathbf{u} = [t_h, \tau_x, \tau_y, \tau_z]^T, \tag{3.13}$$

with:

$$t_h = \sum_{i=1}^{4} f_{i,z}, \tag{3.14}$$

$$\tau_x = \sum_{i=1}^{4} \left(m_{i,x} + \tau_{i,x}\right), \tag{3.15}$$

$$\tau_y = \sum_{i=1}^{4} \left(m_{i,y} + \tau_{i,y}\right), \tag{3.16}$$

$$\tau_z = \sum_{i=1}^{4} \left(m_{i,z} + \tau_{i,z}\right). \tag{3.17}$$

The system is linearized around an equilibrium point defined by:

$$\mathbf{x}_{ep} = \begin{bmatrix} \mathbf{p}_e \\ \mathbf{0}_{1\times 2} \\ \psi_e \\ \mathbf{0}_{1\times 6} \end{bmatrix}, \tag{3.18}$$

$$\mathbf{u}_{ep} = \begin{bmatrix} mg \\ \mathbf{0}_{1\times 3} \end{bmatrix}, \tag{3.19}$$

$$\mathbf{A}_t = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{ep},\mathbf{u}_{ep}}, \tag{3.20}$$

$$\mathbf{B} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{x}_{ep},\mathbf{u}_{ep}}. \tag{3.21}$$

An important property of the linearized model is that its eigenvalues determine the system's stability. In our case, the eigenvalues are all at zero, indicating that, in the absence of additional control, the system behaves like an inertial body—moving with constant velocity upon any input. Moreover, the linearized dynamics decouple into independent subsystems: the

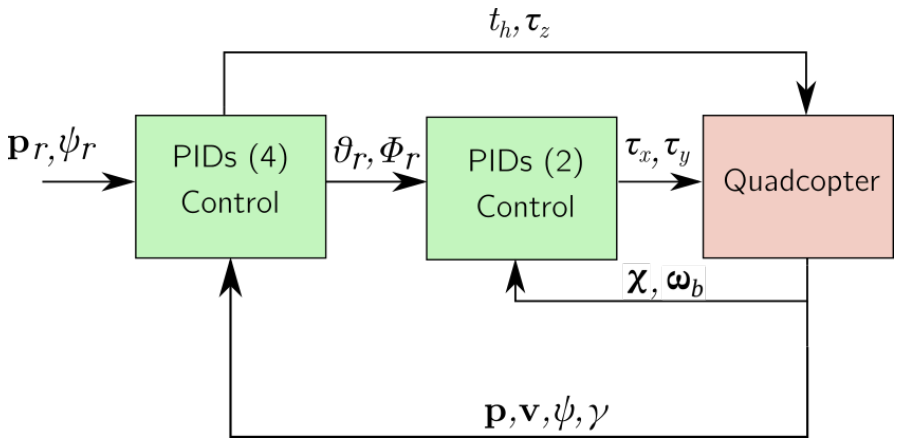


Figure 3.3: Cascade PID Structure in the Parrot Mambo drone.

roll and pitch as inputs., yaw (ψ), and height (p_z) depend solely on τ_x, τ_y, τ_z, and t_h, respectively. This decoupling allows for a single-input single-output (SISO) design for each state. Once designed, an outer loop is created to control positions p_x and p_y using the roll and pitch as inputs, see Fig. 3.3.

As the control is computed on a discrete device, we must express the control law in discrete form, the PID controller is then expressed as:

$$u(k) = K_p e(k) + K_d \frac{e(k) - e(k-1)}{\Delta t} + K_i \sum_{i=0}^{k} e(i)\Delta t, \tag{3.22}$$

where the error is $e(k) = r(k) - x(k)$. The proportional term drives the state toward the reference, the integral term eliminates steady-state error, and the derivative term improves transient behavior by damping the response. PID parameters are tuned using optimization techniques (in our case, a genetic algorithm) to minimize a cost function such as

$$\min_{K_p, K_d, K_i} \quad \sqrt{\sum_{k=0}^{N} (\mathbf{x}(k) - \mathbf{r}(k))^2}, \tag{3.23}$$

where $\mathbf{x}(k)$ and $\mathbf{r}(k)$ are the state and reference at time step k, and N is the horizon.

The cascade PID structure is widely adopted in UAV autopilots due to its simplicity and modularity. The decoupling of dynamics into separate SISO channels for altitude, yaw, roll, and pitch not only simplifies the controller design but also enables independent tuning for each subsystem. Under nominal conditions, this structure offers robust performance in the presence of moderate modeling uncertainties and sensor noise. However, its performance is highly dependent on the accurate tuning of the PID gains. Suboptimal tuning may lead to issues such as excessive overshoot, oscillatory behavior, or sluggish transient responses.

Moreover, the discrete implementation of the PID controller requires careful handling of the derivative term, especially when sensor noise is significant. In practice, filtering techniques are often employed to mitigate the amplification of noise, ensuring that the derivative action contributes positively to system damping without inducing jitter.

While the cascade PID controller provides a robust and easily implementable baseline, it may not fully address the challenges posed by the highly nonlinear and coupled dynamics of UAVs under external perturbations or rapid maneuvering. For instance, the drone will often operate at an inclined attitude, which deviates from the linearized equilibrium point, leading to performance degradation in the control and can lead to instability. This limitation motivates the exploration of advanced control strategies, such as feedback linearization, which are discussed in subsequent chapters.

Feedback linearization explicitly cancels the non-linearities in the system dynamics to achieve a linear input-output relationship, allowing the application of linear control design methods to a broader operational regime. This approach offers potential improvements in terms of response speed, precision, and robustness against large disturbances and model uncertainties. In this context, the cascade PID controller serves as a benchmark, highlighting the trade-offs between simplicity and optimal performance. The insights gained from the analysis of the PID cascade control provide a solid foundation for understanding and appreciating the benefits of the more advanced control methods developed later in this thesis.

3.6.3 Attitude Estimation

The estimation process fuses data from several sensors to obtain accurate state estimates [51]. For instance, the roll and pitch angles are computed

from the accelerometer measurements as follows:

$$\phi_m(k) = \arctan \frac{a_{y,m}(k)}{a_{z,m}(k)}, \quad \theta_m(k) = \arcsin \frac{a_{x,m}(k)}{g}, \tag{3.24}$$

where the subscripts x, y, and z refer to the measurement components and the index m denotes a measured value.

These measurements are combined with rotational speeds data to predict the attitude:

$$\phi(k) = \hat{\phi}(k-1) + \Delta t \zeta_m(k) - \Delta t Bias_\phi(k), \tag{3.25}$$

$$\theta(k) = \hat{\theta}(k-1) + \Delta t \iota_m(k) - \Delta t Bias_\theta(k), \tag{3.26}$$

where Δt represents the time interval and $Bias$ is the inherent bias in the gyroscopic measurement. The update step is computed as:

$$\begin{bmatrix} \hat{\phi}(k) \\ Bias_\phi(k) \end{bmatrix} = \begin{bmatrix} \phi(k) \\ Bias_\phi(k) \end{bmatrix} + K_1(k)(\phi_m(k) - \phi(k)), \tag{3.27}$$

$$\begin{bmatrix} \hat{\theta}(k) \\ Bias_\theta(k) \end{bmatrix} = \begin{bmatrix} \theta(k) \\ Bias_\theta(k) \end{bmatrix} + K_2(k)(\theta_m(k) - \theta(k)), \tag{3.28}$$

where $K_1(k)$ and $K_2(k)$ are the Kalman gains. The yaw angle ψ is obtained by integrating the rotationnal rate:

$$\hat{\psi}(k) = \hat{\psi}(k-1) + \Delta t \gamma_m(k). \tag{3.29}$$

Similarly, the body angular rates are estimated as:

$$\hat{\zeta}(k) = \zeta_m(k) - Bias_\phi(k), \tag{3.30}$$

$$\hat{\iota}(k) = \iota_m(k) - Bias_\theta(k), \tag{3.31}$$

$$\hat{\gamma}(k) = \gamma_m(k). \tag{3.32}$$

3.6.4 Height and Position Estimation

Height estimation relies on the sonar sensor measurement, $p_{z,m}(k)$, and acceleration data transformed from the body frame to the inertial frame using a rotation matrix $\mathbf{R}(\hat{\chi})$:

$$a_e = \mathbf{R}(\hat{\chi}) a_b. \tag{3.33}$$

A simple kinematic model is then used:

$$\begin{bmatrix} p_z(k) \\ v_z(k) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{p}_z(k-1) \\ \hat{v}_z(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} a_{z,m}(k), \tag{3.34}$$

with an update step:

$$\begin{bmatrix} \hat{p}_z(k) \\ \hat{v}_z(k) \end{bmatrix} = \begin{bmatrix} p_z(k) \\ v_z(k) \end{bmatrix} + K_3(k)(p_{z,m}(k) - p_z(k)). \tag{3.35}$$

Position estimation employs optical flow from a ground-facing camera. Since the field of view is proportional to the altitude, the initial measurements $v_{x,m}$ and $v_{y,m}$ are calibrated:

$$v_{x,fc}(k) = 1.15\hat{p}_z(k)v_{x,m}(k), \tag{3.36}$$
$$v_{y,fc}(k) = 1.15\hat{p}_z(k)v_{y,m}(k), \tag{3.37}$$

and further corrected for the drone's rotation:

$$v_{x,sc}(k) = v_{x,fc}(k) - \hat{p}_z(k)\hat{\zeta}(k), \tag{3.38}$$
$$v_{y,sc}(k) = v_{y,fc}(k) - \hat{p}_z(k)\hat{\imath}(k). \tag{3.39}$$

The corrected measurements are then used in Kalman filtering. The prediction step is given by:

$$\begin{bmatrix} v_x(k) \\ v_y(k) \\ a_x(k) \\ a_y(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{v}_x(k-1) \\ \hat{v}_y(k-1) \\ \hat{a}_x(k-1) \\ \hat{a}_y(k-1) \end{bmatrix} + \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a_{x,m}(k) \\ a_{y,m}(k) \end{bmatrix}, \tag{3.40}$$

and the update step is:

$$\begin{bmatrix} \hat{v}_x(k) \\ \hat{v}_y(k) \\ \hat{a}_x(k) \\ \hat{a}_y(k) \end{bmatrix} = \begin{bmatrix} v_x(k) \\ v_y(k) \\ a_x(k) \\ a_y(k) \end{bmatrix} + K(k) \begin{bmatrix} v_{x,sc}(k) - v_x(k) \\ v_{y,sc}(k) - v_y(k) \end{bmatrix}. \tag{3.41}$$

Once the velocities are filtered, the position can be computed by simple integration:

$$\hat{p}_x(k) = \hat{p}_x(k-1) + \Delta t \hat{v}_x(k), \tag{3.42}$$

$$\hat{p}_y(k) = \hat{p}_y(k-1) + \Delta t \hat{v}_y(k). \tag{3.43}$$

In the Mathworks simulator, the sensors and estimators are assumed to be perfectly designed so that the estimates match the actual states. This assumption does not hold in reality and is the major source of discrepancy between simulation tests and real-life experiments.

Chapter 4

Feedback Linearization

4.1 Introduction

Once the fundamental system dynamics have been highlighted, we can focus on advanced control strategies for quadcopters. A key challenge for small and medium-sized Unmanned Aerial Vehicles (UAVs) is the limited computational power of embedded systems, which necessitates the use of fast and efficient control methods. A natural approach is to rely on either linear methods or analytical computations. The Parrot Mambo minidrone, for instance, has gained significant popularity in both educational settings and as a versatile platform for testing various control paradigms. Previous research on similar platforms has explored a range of control approaches, including classic PID and fuzzy PID controllers [52, 53, 54], nonlinear adaptive control techniques [55, 56], and various model-based control strategies [57].

In general, a wide spectrum of control strategies can be employed for developing quadcopter flight controllers. Classical linear control techniques, such as PID and Linear Quadratic Regulator (LQR) control [58], are often considered. These methods can effectively manage quadcopter dynamics under specific assumptions, particularly near hovering conditions. However, many contemporary applications, including complex trajectory tracking, real-time collision avoidance, and robust wind perturbation rejection, demand control performance across a wider envelope of flight conditions. This necessity has driven the development and application of advanced nonlinear control techniques, such as gain scheduling control [59],

backstepping methods [60], feedback linearization [61], and nonlinear model predictive control (NMPC) [62].

Among these, Nonlinear Dynamic Inversion (NDI) stands out as a prominent feedback linearization method. Its primary advantage lies in its ability to yield partially or entirely linearized closed-loop dynamics, thereby allowing the application of well-established linear control techniques for controller design. A particularly robust variant, INDI, further enhances robustness by significantly reducing model dependency. The INDI controller achieves this by solving the incremental form of the equations of motion using acceleration feedback, thus circumventing the need for detailed and often uncertain models, particularly those describing complex aerodynamic phenomena [63, 64].

In this chapter, we present and develop two feedback linearization strategies: the classic approach and the Incremental Nonlinear Dynamic Inversion (INDI) method. These techniques are specifically chosen for their computational efficiency and their ability to provide a linearized closed-loop system, which is a highly desirable property for subsequent analysis and embedded implementation on resource-constrained platforms like the Parrot Mambo minidrone. We then compare these methods through comprehensive simulations and demonstrate their practical performance with an experimental validation against a classic PID controller on the Parrot Mambo drone. This work extends to discussing the specific implementation and validation steps, both in simulation using the Matlab/Simulink platform and through real-life tests, thereby providing an educational context for the INDI control technique.

4.2 Classic Approach

The underlying principle of feedback linearization [65] is to design a nonlinear control law such that the closed-loop system becomes linear (see Fig. 4.1).

If we consider a nonlinear system:

$$\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, \mathbf{u}) \tag{4.1}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{4.2}$$

where $\mathbf{x}$ represents the state vector, $\mathbf{u}$ the input vector, and $\mathbf{y}$ the output vector (the measurements), with $\mathbf{g}$ and $\mathbf{h}$ being nonlinear vector functions

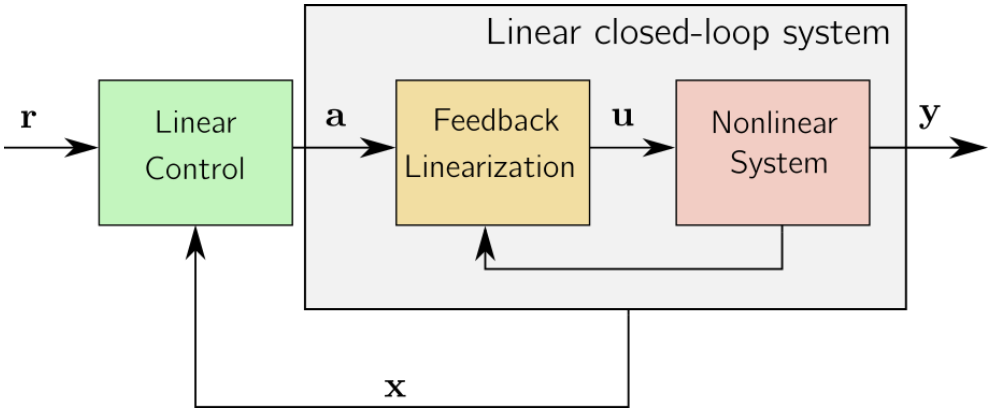


Figure 4.1: Feedback linearization principle.

describing the dynamics, the idea is to differentiate the output with respect to time:

$$\frac{d\mathbf{y}}{dt} = \frac{d\mathbf{h}(\mathbf{x})}{dt} = \frac{d\mathbf{h}(\mathbf{x})}{d\mathbf{x}}\frac{d\mathbf{x}}{dt} = \nabla\mathbf{h}\,\mathbf{g}(\mathbf{x},\mathbf{u}) = \mathbf{L}_g\mathbf{h}(\mathbf{x}) \tag{4.3}$$

which corresponds to the Lie derivative of $\mathbf{h}$ along $\mathbf{g}$. This process is repeated until the Lie derivative can be expressed in an affine input form:

$$\dot{\mathbf{y}}_r = \frac{d^r\mathbf{h}(\mathbf{x})}{d\mathbf{x}}\frac{d\mathbf{x}}{dt^r} = \mathbf{L}_f^r\mathbf{h}(\mathbf{x}) = \mathbf{L}_1^r\mathbf{h}(\mathbf{x}) + \mathbf{L}_2^r\mathbf{h}(\mathbf{x})\mathbf{u} \tag{4.4}$$

A reference model then imposes the output dynamics through the selection of the coefficients β_j, $j = 1, \ldots, r$:

$$\beta_0(\mathbf{r}_d - \mathbf{y}) + \beta_1\frac{d(\mathbf{r}_d - \mathbf{y})}{dt} + \ldots + \beta_r\frac{d^r(\mathbf{r}_d - \mathbf{y})}{dt^r} = 0 \tag{4.5}$$

Equations (4.4) and (4.5) allow us to define the manipulated input $\mathbf{u}$ that must be applied to achieve the desired linear reference dynamics in closed-loop:

$$\mathbf{u} = (\mathbf{L}_2^r\mathbf{h}(\mathbf{x}))^{-1}\Big[\beta_0(\mathbf{r}_d - \mathbf{h}(\mathbf{x})) + \beta_1\Big(\frac{d\mathbf{r}_d}{dt} - \mathbf{L}_f\mathbf{h}(\mathbf{x})\Big) + \ldots + \beta_r\Big(\frac{d^r\mathbf{r}_d}{dt^r} - \mathbf{L}_1^r\mathbf{h}(\mathbf{x})\Big)\Big] \tag{4.6}$$

The method, however, requires the estimation of an augmented state, as in [66], where yaw acceleration, the three translation accelerations, and

the three translation jerks must be taken into account. Let us derive the method using our model. First, consider the yaw dynamics:

$$\dot{\psi} = \begin{bmatrix} 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} \zeta \\ \iota \\ \gamma \end{bmatrix} = \mathbf{R}_{\psi}\boldsymbol{\omega}_b \tag{4.7}$$

For this equation, the inputs are the angular velocities. Let us differentiate this dynamics with respect to time:

$$\ddot{\psi} = \mathbf{R}_{\psi}\dot{\boldsymbol{\omega}}_b + \dot{\mathbf{R}}_{\psi}\boldsymbol{\omega}_b \tag{4.8}$$

$$= \mathbf{R}_{\psi}\Big(\mathbf{I}^{-1}\big(-\boldsymbol{\omega}_b \times \mathbf{I}\boldsymbol{\omega}_b\big) + \mathbf{I}^{-1}\sum_{i=1}^{4}\Big[\mathbf{m}_i + \boldsymbol{\tau}_i\Big]\Big) + \dot{\mathbf{R}}_{\psi}\boldsymbol{\omega}_b \tag{4.9}$$

This expression is affine in the inputs w_i. A similar procedure is applied for the velocity dynamics, and one obtains the acceleration dynamics:

$$\ddot{\mathbf{v}} = \dot{\mathbf{R}}(\boldsymbol{\chi})\frac{1}{m}\sum_{i=1}^{4}\mathbf{f}_i + \mathbf{R}(\boldsymbol{\chi})\frac{1}{m}\sum_{i=1}^{4}\dot{\mathbf{f}}_i \tag{4.10}$$

Differentiating again to obtain the jerk dynamics, and considering the SO_3 group that gives the dynamics of the rotation matrix via the skew-symmetric matrix $\mathbf{SK}$, where $\dot{\mathbf{R}}(\boldsymbol{\chi}) = \mathbf{R}(\boldsymbol{\chi})\mathbf{SK}(\boldsymbol{\omega}_b)$, we have:

$$\dddot{\mathbf{v}} = \ddot{\mathbf{R}}(\boldsymbol{\chi})\frac{1}{m}\sum_{i=1}^{4}\mathbf{f}_i + 2\dot{\mathbf{R}}(\boldsymbol{\chi})\frac{1}{m}\sum_{i=1}^{4}\dot{\mathbf{f}}_i + \mathbf{R}(\boldsymbol{\chi})\frac{1}{m}\sum_{i=1}^{4}\ddot{\mathbf{f}}_i \tag{4.11}$$

$$= \Big(\mathbf{R}(\boldsymbol{\chi})\mathbf{SK}(\boldsymbol{\omega}_b)\mathbf{SK}(\boldsymbol{\omega}_b) + \mathbf{R}(\boldsymbol{\chi})\mathbf{SK}(\dot{\boldsymbol{\omega}}_b)\Big)\frac{1}{m}\sum_{i=1}^{4}\mathbf{f}_i$$

$$+ 2\mathbf{R}(\boldsymbol{\chi})\mathbf{SK}(\boldsymbol{\omega}_b)\frac{1}{m}\sum_{i=1}^{4}\dot{\mathbf{f}}_i + \mathbf{R}(\boldsymbol{\chi})\frac{1}{m}\sum_{i=1}^{4}\ddot{\mathbf{f}}_i$$

The jerk, representing the derivative of acceleration, is formulated as an affine function of a newly defined input, specifically the second derivative of the rotor speed $\ddot{w}_i$. Through these derivations, an input-affine model

emerges, facilitating the application of equation (4.6) to develop the desired closed-loop dynamics. This approach aims to control four outputs: the heading direction ψ and the quadcopter position $\mathbf{p}$. However, a notable drawback is the introduction of additional states, including acceleration, jerk, heading velocity, and input derivatives. Estimating these variables requires more complex observers and additional sensors, adding to the system's intricacy.

4.3 Incremental Nonlinear Dynamic Inversion

The Incremental Nonlinear Dynamic Inversion (INDI) controller builds upon feedback linearization [67, 63, 64]. Instead of differentiating the output multiple times until the Lie derivative can be expressed in an affine input form, the dynamics are expanded using a first-order Taylor series expansion, neglecting higher-order terms:

$$\begin{aligned}\dot{\mathbf{y}} &= \nabla\mathbf{h}\,\mathbf{g}(\mathbf{x},\mathbf{u}) \\ &\approx \nabla\mathbf{h}\Big[\mathbf{g}(0) + \frac{\partial\mathbf{g}(\mathbf{x},\mathbf{u})}{\partial\mathbf{x}}\mathbf{x} + \frac{\partial\mathbf{g}(\mathbf{x},\mathbf{u})}{\partial\mathbf{u}}\mathbf{u}\Big] \\ &= \nabla\mathbf{h}\,\mathbf{g}(0) + \nabla\mathbf{h}\frac{\partial\mathbf{g}(\mathbf{x},\mathbf{u})}{\partial\mathbf{x}}\mathbf{x} + \nabla\mathbf{h}\frac{\partial\mathbf{g}(\mathbf{x},\mathbf{u})}{\partial\mathbf{u}}\mathbf{u} \\ &= \mathbf{L}_0\mathbf{h}(\mathbf{x}) + \mathbf{L}_x\mathbf{h}(\mathbf{x})\mathbf{x} + \mathbf{L}_u\mathbf{h}(\mathbf{x})\mathbf{u} \end{aligned} \tag{4.12}$$

so that an input-affine term readily appears. The output behavior can then be imposed according to a lower-order reference model, i.e., $\dot{\mathbf{y}} = \mathbf{a}$, so that the corresponding actual input is given by

$$\mathbf{u} = (\mathbf{L}_u\mathbf{h}(\mathbf{x}))^{-1}\Big(\mathbf{a} - \mathbf{L}_0\mathbf{h}(\mathbf{x}) - \mathbf{L}_x\mathbf{h}(\mathbf{x})\mathbf{x}\Big). \tag{4.13}$$

Note that the pseudo-inverse of $\mathbf{L}_u$ can be computed if its inverse does not exist. The virtual input $\mathbf{a}$ can be expressed as a linear state feedback law

$$\boldsymbol{\nu} = -\mathbf{K}\mathbf{x}, \tag{4.14}$$

The selection of the gain matrix $\mathbf{K}$ imposes the dynamics of the linear closed-loop system. Equation (4.14) defines the NDI outer loop, while equation (4.13) defines the inner loop.

The main drawback of the feedback linearization methods is that they require accurate knowledge of the model, which is not guaranteed in practical applications, as a detailed and reliable aerodynamic model is often difficult to establish. To alleviate this issue, an incremental version (INDI) has been proposed in [67] based on a time-scale separation between fast and slow phenomena. The fast time scale is related to the actuation of the UAV, whereas the slow time scale is related to the body aerodynamic effects.

Returning to equation (4.12), and considering a small time interval Δt with the corresponding variations in the input and output signals, we assume that

$$\Delta \dot{\mathbf{y}} \approx \mathbf{L}_u \mathbf{h}(\mathbf{x}) \Delta \mathbf{u}, \tag{4.15}$$

neglecting the dynamics that are not directly induced by the actuators. The control move can thus be obtained via

$$\Delta \mathbf{u} = (\mathbf{L}_u \mathbf{h}(\mathbf{x}))^{-1} \Delta \mathbf{a}. \tag{4.16}$$

This formulation lends itself to a discrete-time implementation, which will be adopted for the remainder of this study. The control law only involves the Lie derivative associated with the input and is therefore independent of much of the aerodynamic model and its associated uncertainties [67]. The past virtual input can be evaluated using the measurements:

$$\Delta \mathbf{a} = \mathbf{a} - \dot{\mathbf{y}}. \tag{4.17}$$

This method offers the advantage of avoiding the introduction of augmented states. A critical aspect of the INDI control framework is the nature of its linearization error. The primary source of linearization error in INDI arises from the first-order Taylor series expansion used to approximate the system dynamics over a single discrete time step, Δt. The control law assumes that the incremental change in the state derivative is linearly proportional to the incremental change in the control input, thereby neglecting the higher-order terms of the expansion. The magnitude of this transient error is therefore not dependent on a detailed a priori model of the system's nonlinearities, but is instead contingent upon two key factors: the sampling rate of the controller and the rate of change of the system dynamics. For sufficiently high sampling rates, the time increment Δt is small, ensuring that the system state does not deviate significantly

from the point of expansion. Consequently, the contribution of the neglected higher-order terms becomes negligible, and the linear approximation holds with high fidelity. This characteristic is the principal reason for INDI's acclaimed robustness to model uncertainties and external disturbances, as the error is inherently transient and manageable through hardware selection (i.e., fast sensors and computation) rather than exhaustive system identification. However, it requires the explicit inclusion of inputs in the output expression. This characteristic poses a challenge for our quadcopter, given its cascade structure. In this configuration, the inputs w_i first influence the dynamics of angular velocities, which subsequently drive the Euler angles, and these Euler angles in turn determine the UAV's position. Consequently, the INDI procedure is applied to each loop within this hierarchical structure.

The first loop, illustrated in Fig. 4.2, is dedicated to linearizing the dynamics of angular velocities $\dot{\boldsymbol{\omega}}_b$ and the z-axis velocity $\dot{v}_z$. This process involves considering the four inputs—the motor rotational speeds w_i—and reformulating them into an affine form. Subsequently, a linear control law is developed to ensure tracking of a predefined reference in both angular velocities and z-axis velocity.

A secondary loop, as depicted in Fig. 4.3, is devised to linearize the dynamics of the Euler angles $\dot{\boldsymbol{\chi}}$, which are directly influenced by the body-frame angular velocities $\boldsymbol{\omega}_b$. A control law is then formulated to facilitate the tracking of the referenced Euler angles. The output of this second loop is an angular velocity, which becomes the target for the first loop to track. For this control structure, the inner loop dynamics are neglected to simplify the design of the outer loop. This approach is acceptable when the inner dynamics are significantly faster than the outer loop. The reference for the velocity along the z-axis will be determined in the third loop.

Finally, the third loop, illustrated in Fig. 4.4, is dedicated to linearizing the dynamics of velocities $\dot{\mathbf{v}}$ driven by the Euler angles $\boldsymbol{\chi}$. A linear control law is formulated to supply the desired acceleration to the INDI controller based on the desired position. Simultaneously, a desired heading is directly fed into the second loop.

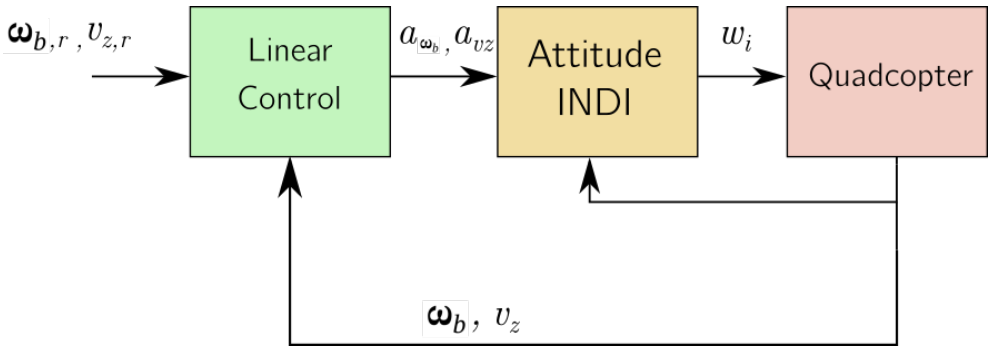


Figure 4.2: First inner loop: attitude linearization.

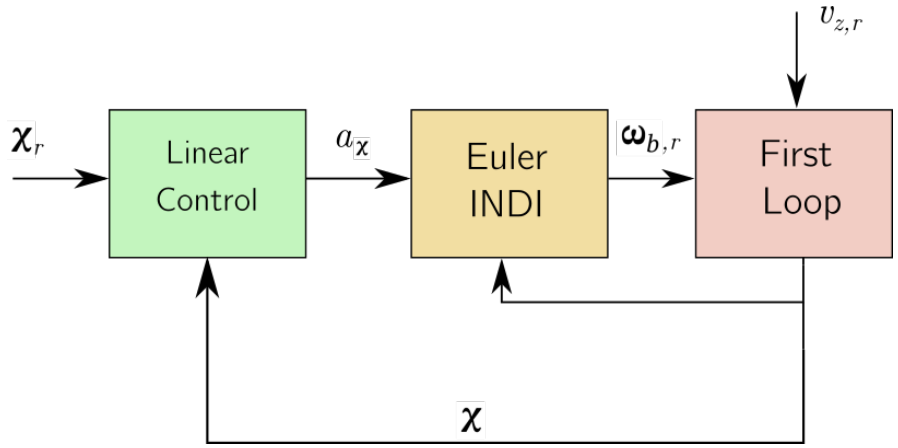


Figure 4.3: Second inner loop: Euler angles linearization.

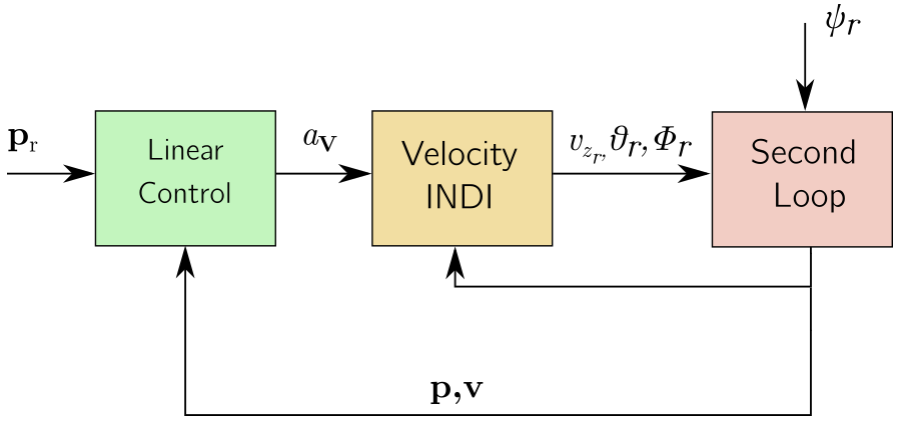


Figure 4.4: Third loop: Position control and Euler angles linearization.

4.4 Results

4.4.1 Simulations on the Parrot Mambo: Classic Feedback Linearization and Incremental Nonlinear Dynamic Inversion under Step Inputs

First, we compare the results of the two feedback linearization methods on the Mathworks simulator of the Parrot Mambo drone. The results are shown in Fig. 4.5. The yaw angle ψ reaches its setpoint in about 2 seconds, while the elevation setpoint is reached in about 3 seconds. The positions p_x and p_y take longer to settle, as they are regulated in a second control layer. Both methods are able to control the UAV as desired; additional tuning can be performed to improve performance and meet specific user needs. In the classic approach, derived states are estimated through a simple differentiation of the filtered (from the Kalman filter) measurements. For the remainder of this work, we will use the incremental structure as it provides greater robustness through the natural cascade structure.

4.4.2 Simulations on the Parrot Mambo: Classic PID Structure versus Incremental Nonlinear Dynamic Inversion under Trajectory Tracking

A circular trajectory is generated, including take-off and landing, to test controller performance under a realistic trajectory tracking scenario.

Cubic B-spline interpolation [68] is used to define the trajectory. Given several control points, the algorithm creates a control polygon within which the trajectory is built. The trajectory is then sampled uniformly (see Fig. 4.6).

The INDI controller is compared against the classic cascade-PID structure for circular trajectory following (see Fig. 4.7). These simulation results show that the PID control scheme allows the specified trajectory to be followed, although it is affected by increasing tracking errors. In contrast, the INDI controller performs significantly better, ensuring a return to the initial point.

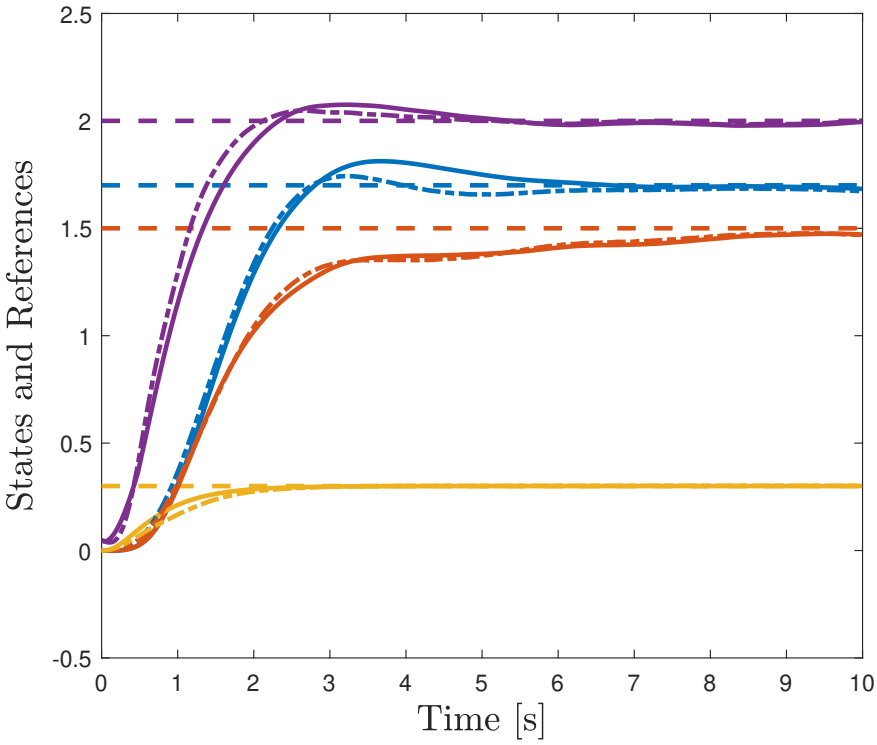


Figure 4.5: Responses of the Mambo minidrone to step setpoints: classic feedback linearization (dotted lines), INDI (continuous lines), references (dashed lines); purple: p_z, blue: p_x, orange: p_y, yellow: ψ.

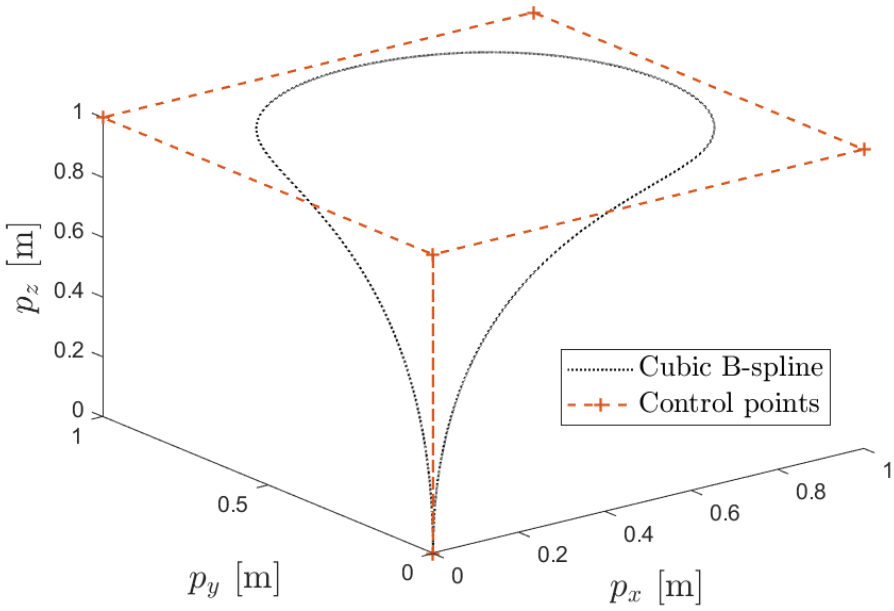


Figure 4.6: Generated cubic B-spline trajectory from given control points.

4.4.3 Real Experimentation on the Parrot Mambo: Classic PID Structure versus Incremental Nonlinear Dynamic Inversion under Trajectory Tracking

To evaluate the performance of the proposed controller in a real-world scenario, we conducted experiments using the Parrot Mambo minidrone. Both the Incremental Nonlinear Dynamic Inversion (INDI) and a classic PID controller were implemented and compiled for the drone's on-board processor.

Our test setup relied on the drone'son-board state estimator, which fuses data from the Inertial Measurement Unit (IMU) and a downward-facing camera using optical flow. This method provides position estimates in a local frame, which are used as feedback for the control loops. While this setup does not use an external motion tracking system, the on-board estimator is representative of a typical consumer-grade drone's capabilities and allows for a direct evaluation of the controllers under realistic conditions.

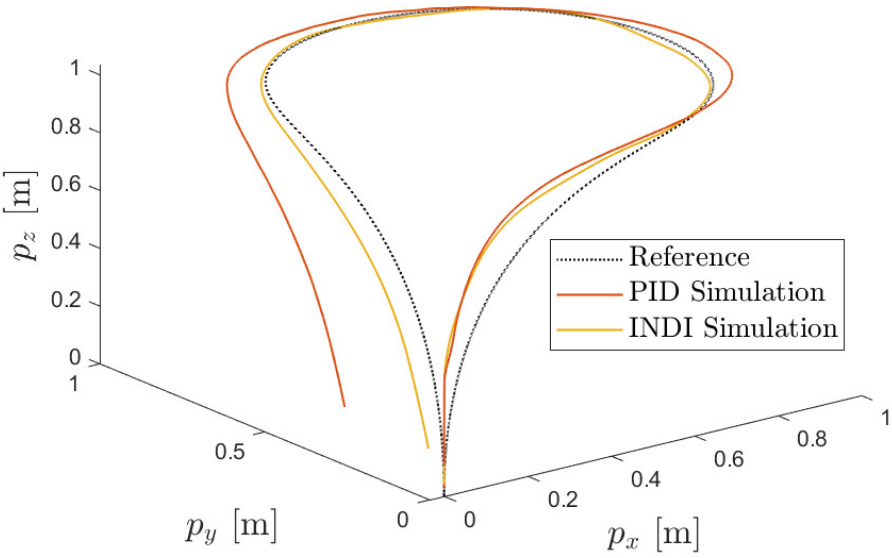


Figure 4.7: Simulation results: Trajectory following comparison between the INDI controller and a classic PID control structure on a 30 seconds time scale.

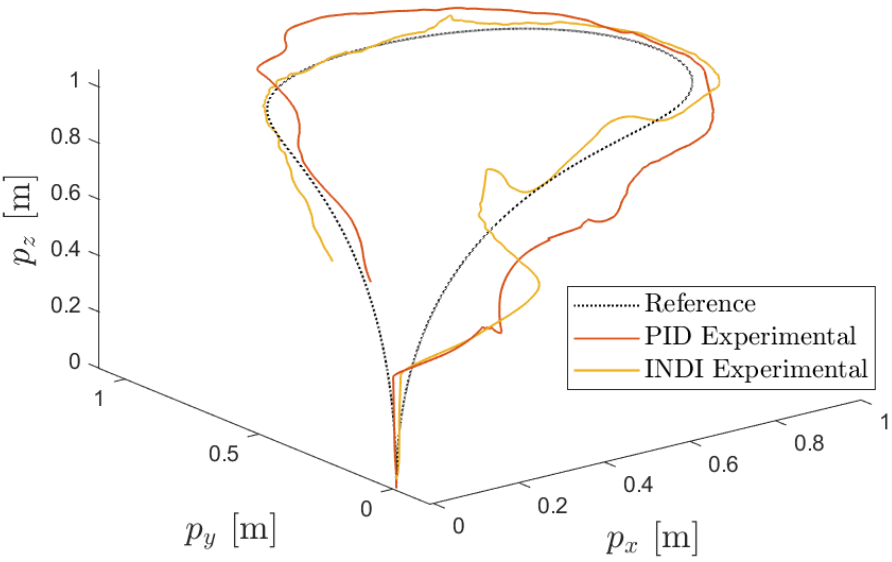


Figure 4.8: Real-life experimental results: Trajectory following comparison between the INDI controller and a classic PID structure on a 30 seconds time scale.

To ensure the statistical relevance of our findings, each experiment was repeated 10 times for both control strategies, and the results presented here are representative of the average performance. Fig. 4.8 shows the experimental results for a single representative trial of the same trajectory tracking problem. A deviation appears in the early stage of both strategies, just after take-off; this is due to the convergence of the optical flow, as the onboard estimator initializes.

We compute the RMS value between the reference and the obtained trajectory for both control strategies:

$$RMS = \sqrt{\frac{1}{N}\sum_{k=0}^{N}\sum_{j=0}^{3}(x_{j,k} - r_{j,k})^2}, \tag{4.18}$$

$$k = 1, 2, \ldots, N, \quad j = 1, 2, 3$$

where x represents the state vector, r the reference, j the vector element,

and k the time step. The INDI controller can track the reference with a lower error ($RMS_{INDI} = 0.091$) compared to the PID structure ($RMS_{PID} = 0.302$), which deviates from the trajectory intermittently. The difference between the two RMS values is explained by the lag of the PID structure relative to the reference. In addition, the INDI controller demonstrates more robust behavior in real-life experiments.

Chapter 5

Set Representation and Reachability Analysis

5.1 Introduction

In our previous chapter, we defined a control strategy for our Unmanned Aerial Vehicles (UAVs) that yields a linear closed-loop model. Building upon this, the current chapter goes deeper into the analysis of this closed-loop behavior to enhance control performance and ensure safe operation. Safety is paramount in aviation, and preventing hazardous events such as Loss of Control in-Flight (LOC-I) remains a crucial concern for both manned and unmanned aircraft [69, 70]. LOC-I can be triggered by various preceding events, highlighting the complexity of mitigating its severe consequences [71]. To properly address this, it is essential not only to focus on recovery from upset flight conditions but also to enhance awareness of situations that may lead to LOC-I.

Enhancing situational awareness, particularly regarding upset flight conditions, is critical for reducing the chance of LOC-I occurrence. A key element in achieving this is the estimation of safe boundaries for the aircraft's maneuvering envelope over a given time horizon, integrating this information into the decision-making process. Such schemes can be realized through various methods, including region-of-attraction analysis [72], path-planning optimization [73], robust-tracking analysis [74], and system-model identification [75, 76]. Other prominent approaches include Monte Carlo simulation [77, 78] and differential-manifold stability analy-

sis [79, 80].

Among these approaches, reachability analysis has emerged as particularly appealing for control system analysis and synthesis [81, 82, 83]. It provides reachable sets that contain all possible state trajectories for given time horizons, initial states, and control inputs [84, 85]. These reachable sets are invaluable as they allow for the prediction of aircraft dynamics without exhaustive simulations and enable the estimation of boundaries within which flight safety is guaranteed. Information derived from reachable sets is therefore essential for both manned and unmanned flights, especially when aircraft encounter upset flight conditions [86, 82, 83].

Reachability analysis can be performed based on optimal control methods [84, 87, 88, 89] or set-propagation methods [90, 91, 92, 93, 94]. While optimal control methods can yield precise estimates in terms of level sets by solving Hamilton-Jacobi-Isaac partial differential equations, their computational cost escalates significantly with the dimension and complexity of the dynamical model [85, 83]. Conversely, set-propagation methods offer a more computationally efficient alternative for models with a larger number of state variables. The efficiency of these methods is largely determined by the choice of set representation. Common representations include intervals [19], ellipsoids [92], and polytopes [21], or, more recently, through generator-based representations like zonotopes [95].

In this chapter, we employ tools from set theory, focusing on zonotopes, which are a core element of this thesis. Zonotopes provide a powerful and reliable set representation in high-dimensional spaces while remaining intuitive and computationally efficient. Their inherent mathematical properties, such as closure under linear transformations and Minkowski summation [96, 91, 97], make them particularly suitable for efficient computation of reachable sets without significant wrapping effects [96]. While conventional methods like Monte Carlo analysis have been traditionally used for safety assessments [98], recent research demonstrates that reachability analysis with set representations like zonotopes often outperforms Monte Carlo techniques in terms of both accuracy and computational efficiency [99]. Moreover, extensions such as polynomial zonotopes and constrained polynomial zonotopes [24] have further enriched the capabilities for representing non-convex sets, offering improved accuracy and broader applicability. These advancements enable the precise computation of a safe flight envelope to ensure secure aircraft maneuvering.

This chapter is organized as follows. First, we present the fundamental

set operations that underpin the zonotopic framework. Next, we introduce zonotopes and their extensions, detailing their properties and representations. Finally, we discuss reachability analysis methods using zonotopes, demonstrating how these techniques can be applied to compute safe flight envelopes for our UAVs, building upon previous studies where zonotopic reachability analysis was applied to linear models of multi-rotor aircraft.

5.2 Set Operations

In this section, we review some common operations, which will be useful in this work, in set theory and provide formal definitions along with 2D illustrations for intuitive comprehension.

5.2.1 Linear Mapping

Linear mapping consists of multiplying matrix to each element of a set.

Definition 5.2.1 (Linear map)**.** Given a set $\mathcal{S} \subset \mathbb{R}^n$ and a matrix $\mathbf{M} \in \mathbb{R}^{v \times n}$, the linear mapping is defined as:

$$\mathbf{M} \otimes \mathcal{S} := \{\mathbf{Ms} \mid \mathbf{s} \in \mathcal{S}\}.$$

This operation is illustrated in Fig. 5.1.

5.2.2 Minkowski Sum

The Minkowski sum adds every element of one set to every element of another set.

Definition 5.2.2 (Minkowski sum)**.** Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, their Minkowski sum is defined as:

$$\mathcal{S}_1 \oplus \mathcal{S}_2 := \{\mathbf{s}_1 + \mathbf{s}_2 \mid \mathbf{s}_1 \in \mathcal{S}_1,\ \mathbf{s}_2 \in \mathcal{S}_2\}.$$

Fig. 5.2 illustrates the Minkowski sum.

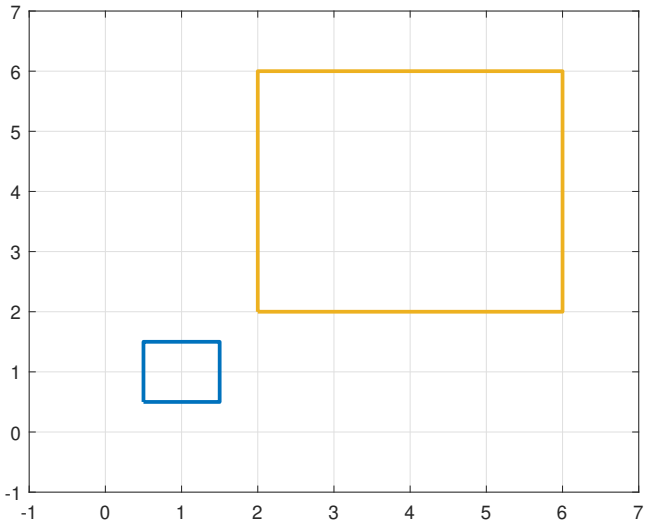


Figure 5.1: Linear mapping (orange) of a initial set (blue) by a matrix $\mathbf{M} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$

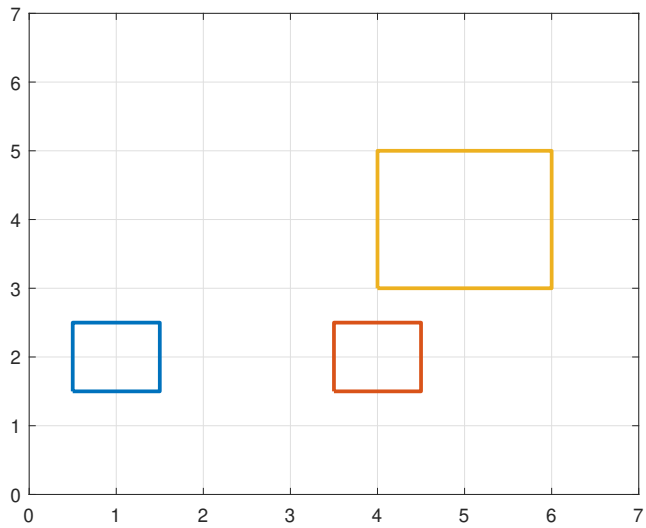


Figure 5.2: Minkowski sum (orange) of two sets (blue and red)

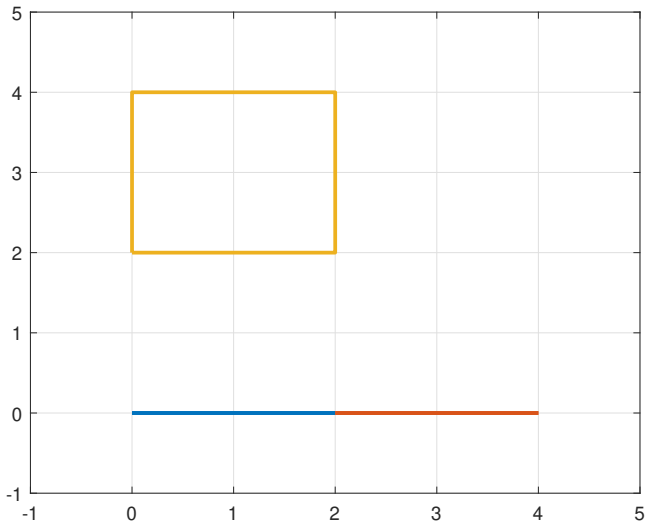


Figure 5.3: Cartesian product (orange) of two one-dimensional sets (blue and red)

5.2.3 Cartesian Product

The Cartesian product combines two sets of given dimensions into a set of higher dimensions.

Definition 5.2.3 (Cartesian product)**.** Given two sets $\mathcal{S}_1 \subset \mathbb{R}^n$ and $\mathcal{S}_2 \subset \mathbb{R}^v$, the Cartesian product is defined as:

$$\mathcal{S}_1 \times \mathcal{S}_2 := \{[\mathbf{s}_1\ \mathbf{s}_2]^T \mid \mathbf{s}_1 \in \mathcal{S}_1,\ \mathbf{s}_2 \in \mathcal{S}_2\}.$$

Fig. 5.3 illustrates the Cartesian product.

5.2.4 Convex Hull

The convex hull of two sets is the set of all convex combinations of points taken from either set. It is analogous to drawing the line segment between two points and generalizes this concept to sets.

Definition 5.2.4 (Convex Hull)**.** Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, their convex hull is defined as:

$$CH(\mathcal{S}_1, \mathcal{S}_2) := \{\lambda\mathbf{s}_1 + (1-\lambda)\mathbf{s}_2 \mid \mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}_1 \cup \mathcal{S}_2,\ \lambda \in [0,1]\}.$$

Fig. 5.4 illustrates the convex hull operation.

5.2.5 Intersection

The intersection operation returns the set of points that are common to both sets, analogous to the logical 'and' operator.

Definition 5.2.5 (Intersection)**.** Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, their intersection is defined as:

$$\mathcal{S}_1 \cap \mathcal{S}_2 := \{\mathbf{s} \mid \mathbf{s} \in \mathcal{S}_1 \text{ and } \mathbf{s} \in \mathcal{S}_2\}.$$

Fig. 5.5 illustrates the intersection operation.

5.2.6 Union

The union of two sets gathers all elements that belong to either set, analogous to the logical 'or' operator.

Definition 5.2.6 (Union)**.** Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, their union is defined as:

$$\mathcal{S}_1 \cup \mathcal{S}_2 := \{\mathbf{s} \mid \mathbf{s} \in \mathcal{S}_1 \text{ or } \mathbf{s} \in \mathcal{S}_2\}.$$

Fig. 5.6 illustrates the union operation.

5.2.7 Pontryagin Difference

The Pontryagin difference is defined based on the Minkowski sum and is used to determine the set of all points which, when added to a given set, remain within another set.

Definition 5.2.7 (Pontryagin Difference)**.** Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, the Pontryagin difference is defined as:

$$\mathcal{S}_1 \ominus \mathcal{S}_2 := \{\mathbf{s} \in \mathbb{R}^n \mid \mathbf{s} \oplus \mathcal{S}_2 \subseteq \mathcal{S}_1\}.$$

Fig. 5.7 illustrates the Pontryagin difference.

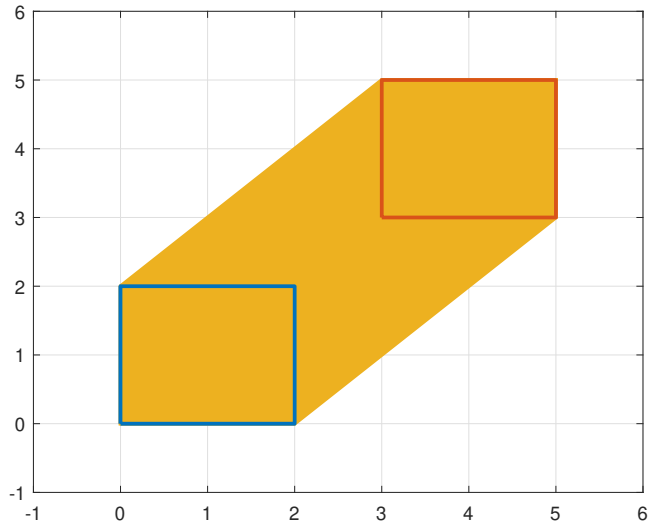


Figure 5.4: Convex hull (orange) of two sets (blue and red)

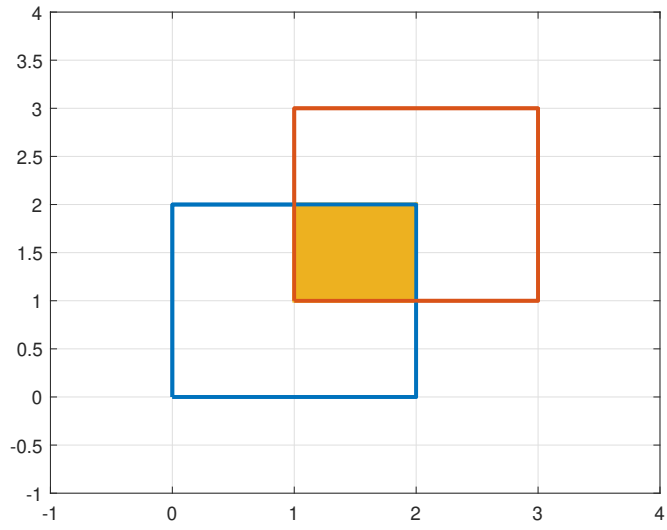


Figure 5.5: Intersection (orange) of two sets (blue and red)

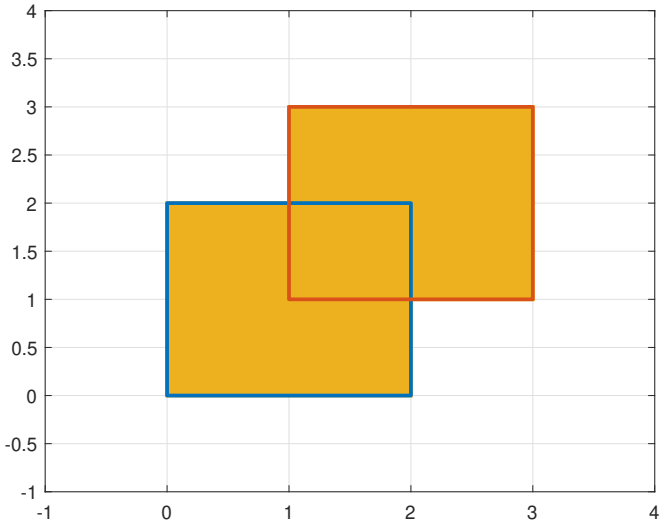


Figure 5.6: Union (orange) of two sets (blue and red)

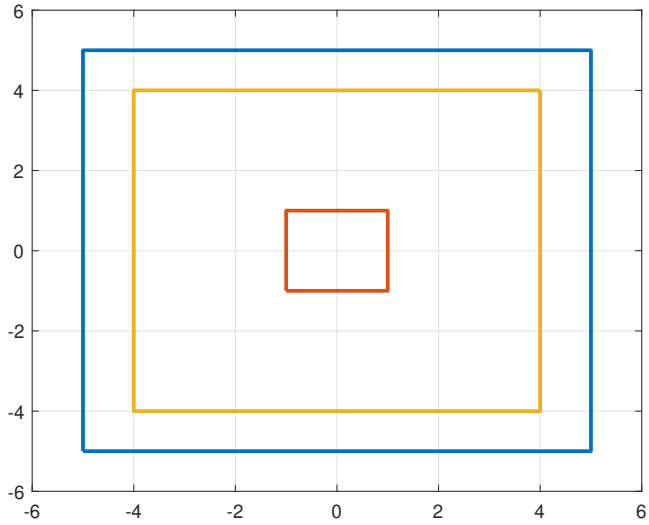


Figure 5.7: Pontryagin difference (orange) of two sets: blue set minus red set

5.3 Set Representation: Zonotopes and Extensions

Having established the fundamental set operations, we now introduce the specific set representations used in this work. Our primary focus is on zonotopes and their extensions, namely constrained zonotopes, polynomial zonotopes, and constrained polynomial zonotopes. These representations support various operations essential for reachability analysis.

5.3.1 Zonotopes

We begin with the definition of a zonotope.

Definition 5.3.1 (Zonotope [25])**.** Provided a center $\mathbf{c} \in \mathbb{R}^n$ and a matrix of generators $\mathbf{G} \in \mathbb{R}^{n \times n_g}$, a zonotope is defined as:

$$\mathcal{Z} := \left\{ \mathbf{c} + \sum_{k=1}^{n_g} \beta_k \mathbf{g}^{(k)} \mid \beta_k \in [-1; 1] \right\} \tag{5.1}$$

Here, $\mathbf{G}$ contains the generators $\mathbf{g}^{(k)}$. A shorthand notation is given by

$$\mathcal{Z} = \langle \mathbf{c}, \mathbf{G} \rangle_Z.$$

A zonotope is constructed as a linear combination of its generators with the scalar coefficients β_k restricted to the interval $[-1; 1]$, resulting in a centrally symmetric polytope. This Z-representation is preferred over half-space or vertex representations because it maintains polynomial complexity in reachability algorithms [100] and is readily extendable. Figure 5.8 illustrates the step-by-step construction of a zonotope; each new generator is incorporated via a Minkowski sum, effectively translating the zonotope along that generator.

5.3.2 Constrained Zonotopes

The first extension is the constrained zonotope.

Definition 5.3.2 (Constrained zonotope [101])**.** Provided a center $\mathbf{c} \in \mathbb{R}^n$, a matrix of generators $\mathbf{G} \in \mathbb{R}^{n \times n_g}$, a constraint generator matrix $\mathbf{A}_c \in$

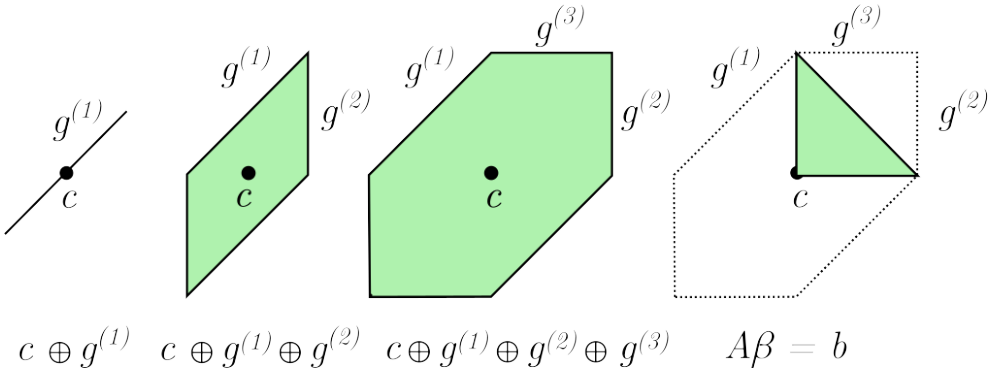


Figure 5.8: Construction of a constrained zonotope. First, the center is defined and translated along the first generator. Each subsequent generator is added through a Minkowski sum, corresponding to a translation along the new generator. A constraint can then be applied to restrict the domain of the generator coefficients.

$\mathbb{R}^{n_c \times p}$, and a constraint offset $\mathbf{b} \in \mathbb{R}^{n_c}$, a constrained zonotope is defined as:

$$\mathcal{CZ} := \left\{ \mathbf{c} + \sum_{k=1}^{n_g} \beta_k \mathbf{g}^{(k)} \mid \sum_{k=1}^{n_c} \beta_k \mathbf{A}_c, (., k) = \mathbf{b}, \;\; \beta_k \in [-1; 1] \right\}$$

A shorthand notation is

$$\mathcal{CZ} = \langle \mathbf{c}, \mathbf{G}, \mathbf{A}_c, \mathbf{b} \rangle_{\mathcal{CZ}}.$$

A constrained zonotope is formed similarly to a zonotope, but with additional linear constraints imposed on the coefficients. Note that a zonotope is a special case of a constrained zonotope with empty constraints:

$$\mathcal{Z} = \langle \mathbf{c}, \mathbf{G} \rangle_{\mathcal{Z}} = \langle \mathbf{c}, \mathbf{G}, [], [] \rangle_{\mathcal{CZ}}.$$

Figure 5.9 shows examples of zonotopes and constrained zonotopes.

5.3.3 Polynomial Zonotopes

The zonotope concept can be further extended to represent more complex shapes by allowing nonlinear combinations of the coefficients with the generators.

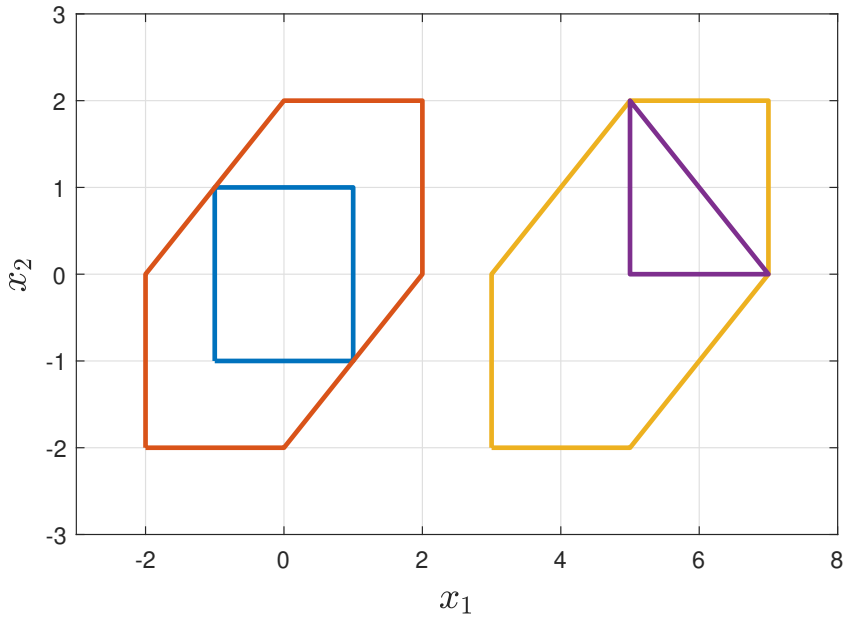


Figure 5.9: Examples of zonotopes and constrained zonotopes.

Blue: $\mathcal{Z} = \langle [0\ 0]^T, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \rangle_z$,

Orange: $\mathcal{Z} = \langle [0\ 0]^T, \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \rangle_z$,

Yellow: $\mathcal{Z} = \langle [5\ 0]^T, \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \rangle_z$,

Purple: $\mathcal{CZ} = \langle [0\ 5]^T, \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, [1\ 1\ 1], 1 \rangle_{cz}$.

Definition 5.3.3 (Polynomial zonotope [24])**.** Provided a center $\mathbf{c} \in \mathbb{R}^n$, a matrix of generators $\mathbf{G} \in \mathbb{R}^{n \times n_g}$, and an exponent matrix $\mathbf{E} \in \mathbb{N}_0^{n_p \times n_g}$, a polynomial zonotope is defined as:

$$\mathcal{PZ} := \left\{ \mathbf{c} + \sum_{i=1}^{n_g} \left(\prod_{k=1}^{n_p} \beta_k^{E_{(k,i)}} \right) \mathbf{g}^{(i)} \mid \beta_k \in [-1; 1] \right\}$$

A shorthand notation is

$$\mathcal{PZ} = \langle \mathbf{c}, \mathbf{G}, \mathbf{E} \rangle_{\mathcal{PZ}}.$$

In this framework, the coefficients undergo polynomial transformations, allowing a more versatile representation. When the exponent matrix is the identity, the polynomial zonotope reduces to a zonotope:

$$\mathcal{Z} = \langle \mathbf{c}, \mathbf{G} \rangle_{\mathcal{Z}} = \langle \mathbf{c}, \mathbf{G}, \mathbf{I} \rangle_{\mathcal{PZ}}.$$

5.3.4 Constrained Polynomial Zonotopes

Finally, polynomial constraints can be applied to the generators, leading to a general nonconvex representation known as a constrained polynomial zonotope.

Definition 5.3.4 (Constrained Polynomial zonotope [24])**.** Provided a center $\mathbf{c} \in \mathbb{R}^n$, a matrix of generators $\mathbf{G} \in \mathbb{R}^{n \times n_g}$, an exponent matrix $\mathbf{E} \in \mathbb{N}_0^{n_p \times n_g}$, a constraint generator matrix $\mathbf{A}_c \in \mathbb{R}^{n_c \times n_q}$, a constraint offset $\mathbf{b} \in \mathbb{R}^{n_c}$, and a constraint exponent matrix $\mathbf{R} \in \mathbb{N}_0^{n_p \times n_q}$, a constrained polynomial zonotope is defined as:

$$\mathcal{CPZ} := \left\{ \mathbf{c} + \sum_{i=1}^{n_g} \left(\prod_{k=1}^{n_p} \beta_k^{\mathbf{E}_{(k,i)}} \right) \mathbf{g}^{(i)} \mid \sum_{j=1}^{n_q} \left(\prod_{k=1}^{n_p} \beta_k^{\mathbf{R}_{(k,i)}} \right) \mathbf{A}_{(.,i)} = \mathbf{b}, \ \beta_k \in [-1; 1] \right\}$$

A reduced notation is given by

$$\mathcal{CPZ} = \langle \mathbf{c}, \mathbf{G}, \mathbf{E}, \mathbf{A}_c, \mathbf{b}, \mathbf{R} \rangle_{\mathcal{CPZ}}.$$

It is easy to observe that a zonotope can be expressed as a constrained polynomial zonotope:

$$\mathcal{Z} = \langle \mathbf{c}, \mathbf{G} \rangle_{\mathcal{Z}} = \langle \mathbf{c}, \mathbf{G}, \mathbf{I}, [], [], [] \rangle_{\mathcal{CPZ}}.$$

While polynomial zonotopes enable complex representations with sparse parameterizations, they still lack certain operations and are difficult to depict graphically since their boundaries are not described by a simple analytical expression. Moreover, when only a few values appear in the different matrices, the resulting shape may become unclear.

5.3.5 Set Operations on Constrained Polynomial Zonotopes

Constrained polynomial zonotopes ($\mathcal{CPZ}$) serve as a versatile tool in computational mathematics, offering a wide range of set operations dedicated to reachability analysis. In this section, we define the operations for $\mathcal{CPZ}$ and explain how the resulting sets are obtained. Let us define specific operations for constrained polynomial zonotopes:

Definition 5.3.5 (Linear map of a $\mathcal{CPZ}$ [25])**.** Given a $\mathcal{CPZ} = \langle \mathbf{c}, \mathbf{G}, \mathbf{E}, \mathbf{A}_c, \mathbf{b}, \mathbf{R} \rangle_{\mathcal{CPZ}} \subset \mathbb{R}^n$ and a matrix $\mathbf{M} \in \mathbb{R}^{v \times n}$, the linear mapping is obtained as:

$$\mathbf{M} \otimes \mathcal{CPZ} := \langle \mathbf{Mc}, \mathbf{MG}, \mathbf{E}, \mathbf{A}_c, \mathbf{b}, \mathbf{R} \rangle_{\mathcal{CPZ}}$$

Definition 5.3.6 (Minkowski sum of two $\mathcal{CPZ}$ [25])**.** Given two $\mathcal{CPZ}$,

$$\mathcal{CPZ}_1 = \langle \mathbf{c}_1, \mathbf{G}_1, \mathbf{E}_1, \mathbf{A}_1, \mathbf{b}_1, \mathbf{R}_1 \rangle_{\mathcal{CPZ}} \subset \mathbb{R}^n$$

and

$$\mathcal{CPZ}_2 = \langle \mathbf{c}_2, \mathbf{G}_2, \mathbf{E}_2, \mathbf{A}_2, \mathbf{b}_2, \mathbf{R}_2 \rangle_{\mathcal{CPZ}} \subset \mathbb{R}^n,$$

their Minkowski sum is defined as:

$$\mathcal{CPZ}_1 \oplus \mathcal{CPZ}_2 := \left\langle \mathbf{c}_1 + \mathbf{c}_2, [\mathbf{G}_1 \ \ \mathbf{G}_2], \begin{bmatrix} \mathbf{E}_1 & 0 \\ 0 & \mathbf{E}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{R}_1 & 0 \\ 0 & \mathbf{R}_2 \end{bmatrix} \right\rangle$$

Definition 5.3.7 (Cartesian product of two $\mathcal{CPZ}$ [25])**.** Given two $\mathcal{CPZ}$,

$$\mathcal{CPZ}_1 = \langle \mathbf{c}_1, \mathbf{G}_1, \mathbf{E}_1, \mathbf{A}_1, \mathbf{b}_1, \mathbf{R}_1 \rangle_{\mathcal{CPZ}} \subset \mathbb{R}^n$$

and

$$\mathcal{CPZ}_2 = \langle \mathbf{c}_2, \mathbf{G}_2, \mathbf{E}_2, \mathbf{A}_2, \mathbf{b}_2, \mathbf{R}_2 \rangle_{\mathcal{CPZ}} \subset \mathbb{R}^v,$$

their Cartesian product is defined as:

$$\mathcal{CPZ}_1 \times \mathcal{CPZ}_2 := \left\langle \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{G}_1 & 0 \\ 0 & \mathbf{G}_2 \end{bmatrix} \begin{bmatrix} \mathbf{E}_1 & 0 \\ 0 & \mathbf{E}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{R}_1 & 0 \\ 0 & \mathbf{R}_2 \end{bmatrix} \right\rangle$$

Definition 5.3.8 (Intersection of two $\mathcal{CPZ}$ [25])**.** Given two $\mathcal{CPZ}$,

$$\mathcal{CPZ}_1 = \langle \mathbf{c}_1, \mathbf{G}_1, \mathbf{E}_1, \mathbf{A}_1, \mathbf{b}_1, \mathbf{R}_1 \rangle_{\mathcal{CPZ}} \subset \mathbb{R}^n$$

and

$$\mathcal{CPZ}_2 = \langle \mathbf{c}_2, \mathbf{G}_2, \mathbf{E}_2, \mathbf{A}_2, \mathbf{b}_2, \mathbf{R}_2 \rangle_{\mathcal{CPZ}} \subset \mathbb{R}^n,$$

their intersection is given by:

$$\mathcal{CPZ}_1 \cap \mathcal{CPZ}_2 := \left\langle \mathbf{c}_1, \mathbf{G}_1, \begin{bmatrix} \mathbf{E}_1 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{A}_1 & 0 & 0 & 0 \\ 0 & \mathbf{A}_2 & 0 & 0 \\ 0 & 0 & \mathbf{G}_1 & -\mathbf{G}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{c}_2 - \mathbf{c}_1 \end{bmatrix}, \begin{bmatrix} \mathbf{R}_1 & 0 & \mathbf{E}_1 & 0 \\ 0 & \mathbf{R}_2 & 0 & \mathbf{E}_2 \end{bmatrix} \right\rangle$$

Definition 5.3.9 (Emptiness of a $\mathcal{CPZ}$ [25])**.** Given a

$$\mathcal{CPZ} = \langle \mathbf{c}, \mathbf{G}, \mathbf{E}, \mathbf{A}, \mathbf{b}, \mathbf{R} \rangle_{\mathcal{CPZ}} \subset \mathbb{R}^n,$$

the set is non-empty if:

$$\mathcal{CPZ} = \emptyset \iff min\{||\beta||_\infty : \sum_{j=1}^{n_q} \left(\prod_{k=1}^{n_p} \beta_k^{\mathbf{R}_{(k,i)}} \right) \mathbf{A}_{(.,i)} = \mathbf{b}\} \leq 1 \tag{5.2}$$

Additionally, $\mathcal{CPZ}$ possesses convex hull and union operations; however, their expressions are considerably more involved. We refer the interested reader to [25] for further details.

Table 5.1 summarizes the operations available for each set representation.

5.4 Reachability Analysis

In this section, we recall the notion of reachability for linear systems and describe how to compute reachable sets using zonotope representations. Consider a set of initial conditions $\mathcal{X}_0$ and an input trajectory $u(\cdot)$ taken from the input set $\mathcal{U}_0$. Let $\mathcal{X}(t_f, x_0, u(\cdot))$ denote the set of solutions of our system at time t_f for an initial condition x_0. The forward reachable set $\mathcal{R}([0, t_f])$ is defined as:

$$\mathcal{R}([0, t_f]) = \{ \mathcal{X}([0, t_f], x_0, u(\cdot)) \in \mathbb{R}^n \mid x_0 \in \mathcal{X}_0, \forall t : u(\cdot) \in \mathcal{U}([0, t_f]) \} \tag{5.3}$$

Table 5.1: Available Operations for Each Representation.

Rep.	Linear Map	Mink. Sum	Cart. Product	Conv. Hull	Intersection	Union	Nonconvex Rep.
$\mathcal{Z}$	✓	✓	✓	o	o	o	✗
$\mathcal{CZ}$	✓	✓	✓	✓	✓	o	✗
$\mathcal{PZ}$	✓	✓	✓	✓	✗	✗	✓
$\mathcal{CPZ}$	✓	✓	✓	✓	✓	✓	✓

Legend: ✓= exact operation, o = over-approximated, ✗= not possible.

The computation of the reachable set can be decomposed into the homogeneous (unforced) solution and the non-homogeneous (forced) solution:

$$\mathcal{R}([0,t_f]) = \mathcal{R}_h([0,t_f]) \oplus \mathcal{R}_p([0,t_f]) \tag{5.4}$$

Each term can be evaluated by subdividing the time interval $[0,t_f]$ into sufficiently short subintervals of duration τ:

$$\mathcal{R}_h([k\tau,(k+1)\tau]) = e^{\mathbf{A}_c\tau}\,\mathcal{R}_h([(k-1)\tau,k\tau]) \tag{5.5}$$

$$\mathcal{R}_p([k\tau,(k+1)\tau]) = e^{\mathbf{A}_c\tau}\,\mathcal{R}_p([(k-1)\tau,k\tau]) \oplus \mathcal{R}_p(\tau) \tag{5.6}$$

where $\mathbf{A}_c$ is the system matrix used to compute the transition matrix $e^{\mathbf{A}_c\tau}$ and k denotes the time step. The final reachable set is obtained as:

$$\begin{aligned}\mathcal{R}([0,t_f]) = \bigcup_{k=1}^{t_f/\tau}\Big(&\mathcal{R}_h([(k-1)\tau,k\tau]) \\ &\oplus \mathcal{R}_p([(k-1)\tau,k\tau]) \oplus \mathcal{R}_p(\tau)\Big)\end{aligned} \tag{5.7}$$

where $\mathcal{R}_p(\tau)$ is evaluated through a Taylor series expansion. For more details about the algorithm, the reader is referred to [102].

We define the *safe flight envelope* as an envelope within which the vehicle is able to exit its nominal condition and return to a hovering state within a finite time horizon [77] (see Fig. 5.10). In other words, the safe flight envelope is the intersection between the forward reachable set and the backward reachable set at a given time instant. For a closed-loop, stabilized system, the safe flight envelope converges to the forward reachable set as time increases.

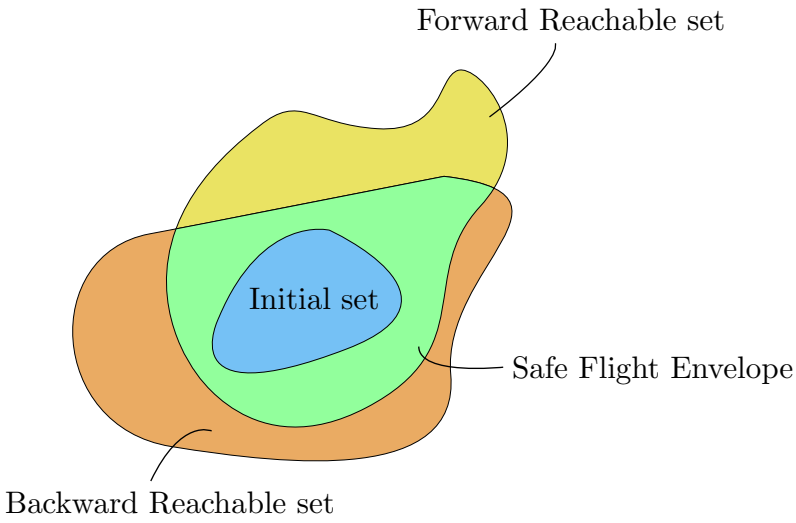


Figure 5.10: Safe Flight Envelope Definition.

It is important to note that the fidelity of the safe flight envelope is directly dependent on the accuracy of the system model, the resulting safe flight envelope is only as reliable as the model used to compute it.

The CORA toolbox [100] offers the possibility to compute a tight over-approximation of the zonotopic forward reachable set from given zonotopes. The algorithm can also compute the zonotopic backward reachable set by simply reversing time (i.e., replacing t by $\tau = -t$ in the system equations).

However, the over-approximation inherent in the algorithm may not fully address our concerns. It is therefore of interest to compute an underestimation of both sets. To this end, we implemented in the CORA toolbox the under-approximation provided by the standard and wrapping-free algorithms proposed in [103]. Figure 5.11 compares the over- and under-approximations.

In the CORA toolbox, the over-approximation of $\mathcal{R}([0, t_f])$ is achieved by over-approximating the initial homogeneous reach set $\mathcal{R}_h([0, \tau])$ through the following three stages:

1. An initial homogeneous solution is computed as:

$$\mathcal{R}_h(\tau) = e^{\mathbf{A}_c \tau} \mathcal{X}_0, \tag{5.8}$$

after which both the initial condition set $\mathcal{X}_0$ and $\mathcal{R}_h(\tau)$ are enclosed

by a global zonotope.

2. An error term derived from the input set is computed and added to the enclosure [102].

3. The resulting zonotope is over-approximated by a lower-order zonotope to avoid excessive zonotope order and reduce computational time.

Our proposed extension for the under-approximation of $\mathcal{R}([0, t_f])$ also proceeds in three stages:

1. As in stage 1 above, we compute $\mathcal{R}_h(\tau)$ (5.8) and enclose it together with the initial condition set $\mathcal{X}_0$ by a global zonotope.

2. A security factor is applied to shrink the resulting zonotope. A diagonal matrix of security factors $\mathbf{S}$ is defined, which depends on the model confidence and safety levels. The safety matrix is applied only to the generators:

$$\mathbf{S} \cdot \mathcal{Z} = (c,\ \mathbf{s}_1 g^{(1)},\ \ldots,\ s_{n_g} g^{(n_g)}) \tag{5.9}$$

where

$$\mathbf{S} = \begin{bmatrix} s_1 & & \\ & \ddots & \\ & & s_{n_g} \end{bmatrix} \in \mathbb{R}^{n_g \times n_g},$$

with $s_i < 1$ for all i. This step does not increase the order of the zonotope; however, further order reduction may be necessary to maintain tractable computations.

3. The zonotope order is reduced by an under-approximation method [103] (see Chapter 5.1 and the implementation available in the CORA toolbox).

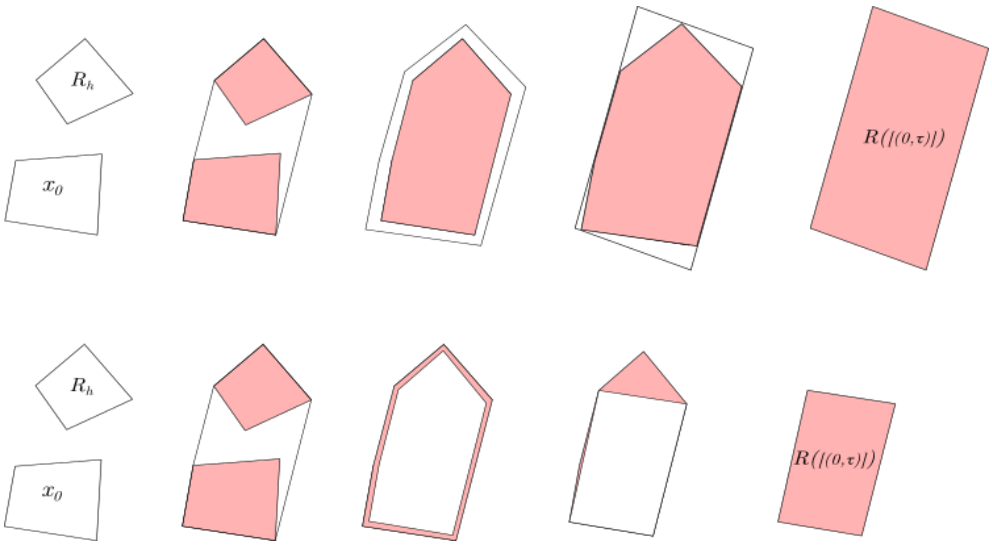


Figure 5.11: Upper graph: computation of the initial time reachable set by the CORA toolbox (over-approximation); lower graph: computation of the initial time interval reachable set by under-approximation.

5.5 Zonotopic Reachability Analysis of the Parrot Mambo

5.5.1 Linear Model Identification

The nonlinear dynamics of the quadcopter are characterized by twelve states (including positions, Euler angles, velocities, and rotational velocities). The incorporation of control and state estimation imparts an almost linear behavior to the closed-loop system under hovering conditions. Thus, we derive a linear model using classical identification techniques (employing the subspace method provided in the Matlab Identification Toolbox). The linear model is reduced to focus on the states for which safety concerns are most critical: the roll angle ϕ, pitch angle θ, and their respective body-frame angular velocities ζ and ι. The considered inputs are the three Euler angle references and the elevation reference.

Training and test data are generated using a multirotor aircraft simulator. Twenty random inputs are applied over 60 seconds for training, and three random inputs are applied over 8 seconds for testing, with a time

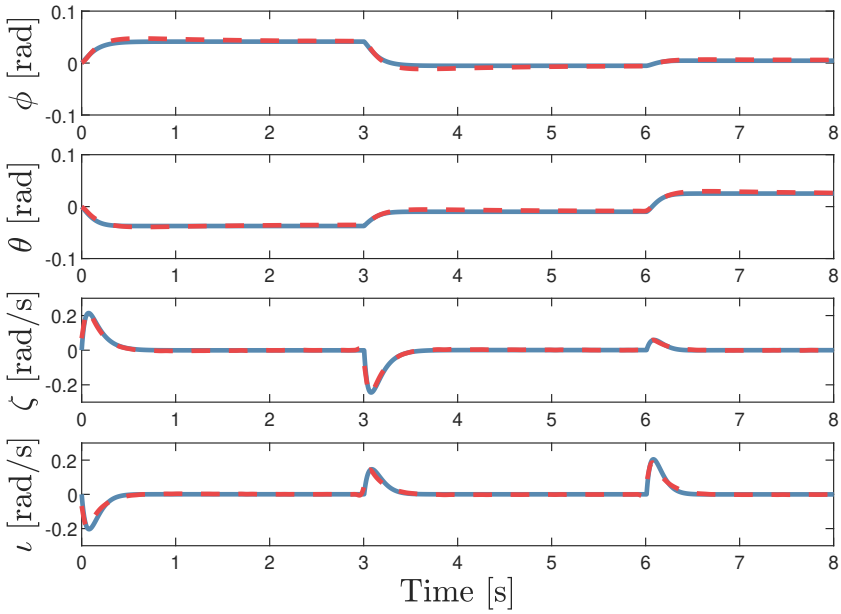


Figure 5.12: Orange dotted line: Test dataset generated using the Simulink Support Package for Parrot Minidrones. Blue line: Prediction of the identified linear model.

step of 0.005 seconds. The identified square system (four inputs and four states) accurately describes the closed-loop multi-rotor aircraft, as shown in Fig. 5.12.

5.5.2 Parrot Mambo Reachable Sets

The under-approximation algorithm is applied to the identified model to obtain the zonotopic forward and backward reachable sets. Figures 5.13 and 5.14 show these sets for a time horizon of 0.01 seconds.

We observe that the safe flight envelope is quite large, as both reachable regions exhibit similar shapes and sizes. In practice, it may also be desirable to account for certain nonlinear constraints (expressed as bounds on the state variables) that are not captured by the linear approximation. When these bounds are violated, the drone is considered to be in a critical state. The final reachable set is obtained by applying these bound con-

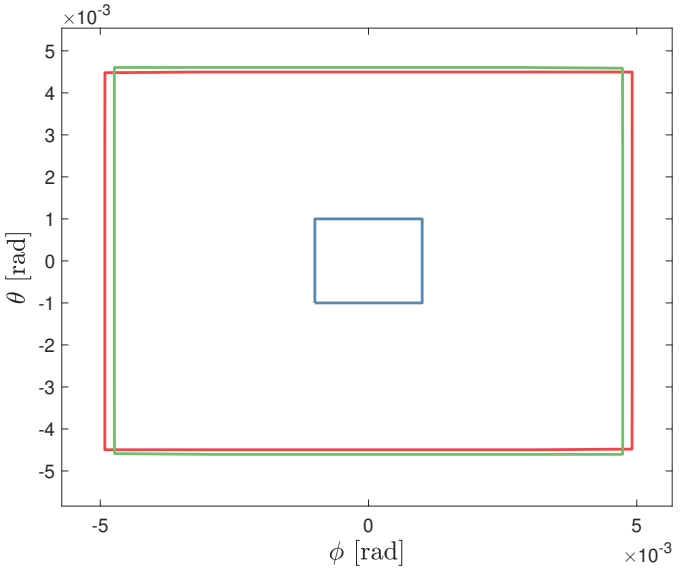


Figure 5.13: Projection of the zonotopes on the 2-dimensional state-space $\phi - \theta$ after 0.01 seconds. Blue: initial zonotope. Red: forward reachable set $\mathcal{R}_f(0.01)$. Green: backward reachable set $\mathcal{R}_b(0.01)$.

straints. Note that zonotopes are not closed under intersection, and so the intersection of the forward and backward reachable sets (which defines the safe flight envelope) should be represented as a constrained zonotope.

Figures 5.15 and 5.16 show the reachable regions under constraints at 0.05 seconds. In these figures, the original backward reachable sets exceed the saturation limits: $|\zeta| < 2.5$ rad/s and $|\iota| < 2.5$ rad/s.

Since a stable linear model approximation is used, the backward system is unstable, and its reachable set grows exponentially. Thus, for longer time periods, the forward reachable set becomes a subset of the backward reachable set, and the flight envelope corresponds to the forward reachable set (see Fig. 5.17). The analysis is focused on short time intervals corresponding to the controller sampling time.

5.5.3 Monte Carlo Approach and Comparison

To validate our approach, the results are compared with a Monte Carlo method. In the Monte Carlo approach, a random initial condition is se-

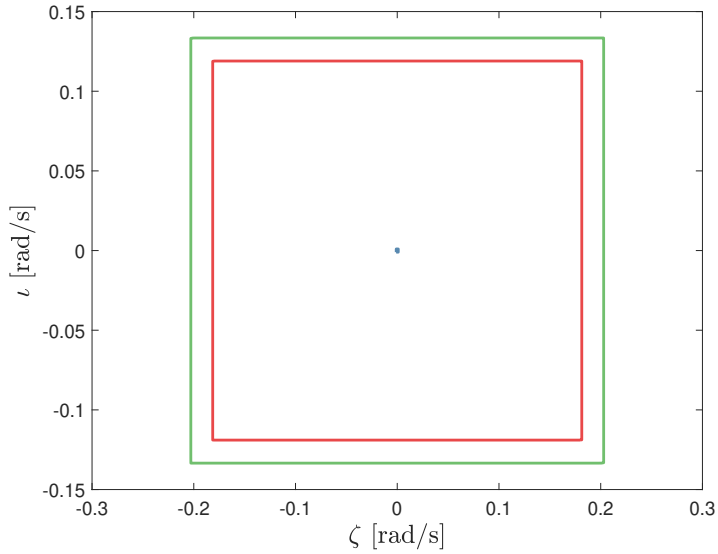


Figure 5.14: Projection of the zonotopes on the 2-dimensional state-space $\zeta - \iota$ after 0.01 seconds. Blue: initial zonotope. Red: forward reachable set $\mathcal{R}_f(0.01)$. Green: backward reachable set $\mathcal{R}_b(0.01)$.

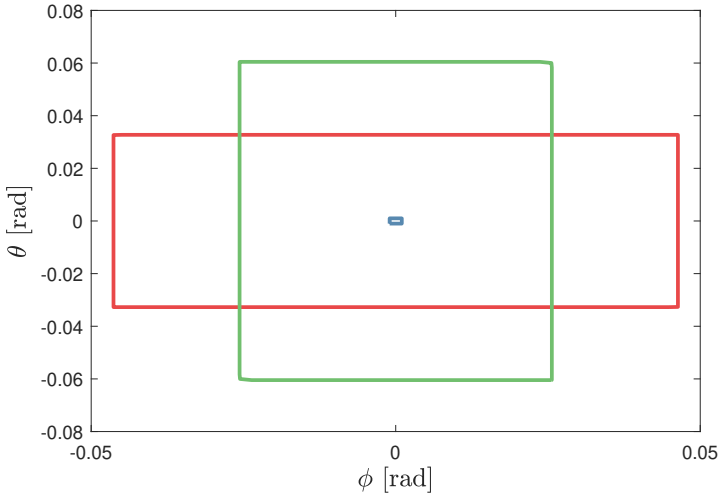


Figure 5.15: Projection of the constrained reachable regions on the 2-dimensional state-space $\phi - \theta$ after 0.05 seconds. Blue: initial zonotope. Red: forward reachable set $\mathcal{R}_f(0.05)$. Green: backward reachable set $\mathcal{R}_b(0.05)$.

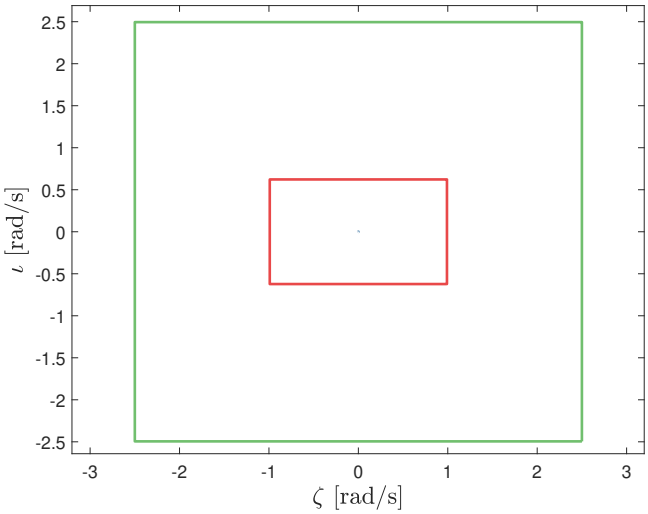


Figure 5.16: Projection of the constrained reachable regions on the 2-dimensional state-space $\zeta - \iota$ after 0.05 seconds. Blue: initial zonotope. Red: forward reachable set $\mathcal{R}_f(0.05)$. Green: backward reachable set $\mathcal{R}_b(0.05)$.

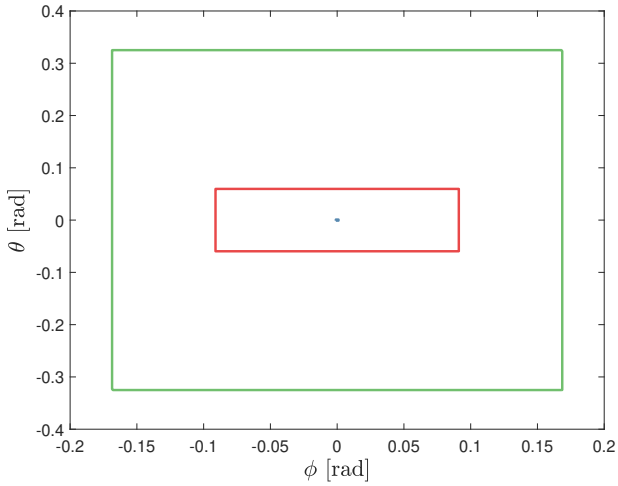


Figure 5.17: Projection of the unsaturated zonotopes on the 2-dimensional state-space $\phi - \theta$ after 0.15 seconds. The flight envelope corresponds to the forward reachable set. Blue: initial zonotope. Red: forward reachable set $\mathcal{R}_f(0.15)$. Green: backward reachable set $\mathcal{R}_b(0.15)$.

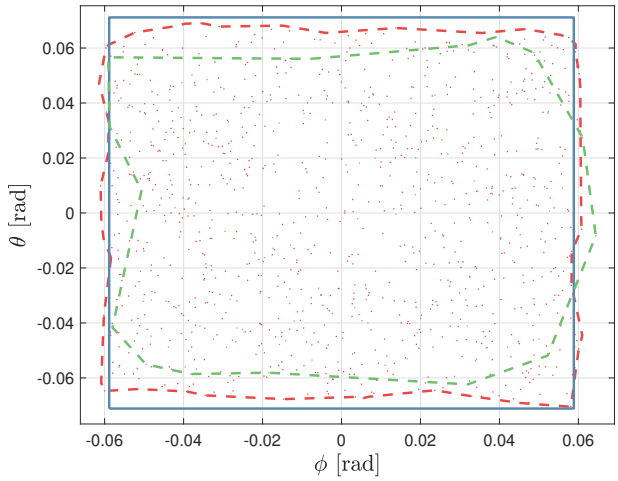


Figure 5.18: Blue line: forward zonotopic reachability analysis. Red dotted line: forward Monte Carlo reachability analysis using the linear system. Red dots: endpoints of linear model trajectories. Green dotted line: forward Monte Carlo reachability using the nonlinear model. Green dots: endpoints of nonlinear model trajectories.

lected and the input space is sampled. Repeated simulations over a specified time span yield trajectories whose endpoints approximate the forward reachable set. An advantage of this method is that it does not require a linear approximation of the system. However, drawbacks include the difficulty of performing backward integration in a complex simulator and the high computational cost required for accurate results.

Figure 5.18 compares the forward reachability analysis based on the zonotopic approach with Monte Carlo simulations using, on one hand, the approximate linear model and, on the other hand, the complete nonlinear model. These results confirm that the implemented techniques yield accurate reachable sets and safe flight envelopes. To compute the zonotopic reachable set, a safety matrix $\mathcal{S}$ with all coefficients equal to 0.95 is used.

The resulting safe flight envelope closely approximates the forward reachable set at any given time and converges to it as the final time increases. This is due to the efficiency of the closed-loop system, which remains stabilized over a wide operational range. The primary interest of the current analysis is in the short time scale for transition performance evaluation.

Chapter 6

Reference Governor

6.1 Introduction

Building on the computation of safe flight envelopes for our system, we now seek to integrate these tools into our control strategies to ensure that the UAV always remains within a defined safe region. Beyond the primary flight controller, various techniques are employed to enhance overall system safety. For instance, envelope protection algorithms are designed to prevent loss of control by ensuring the system remains within its operational limits [104]. Similarly, collision avoidance systems manage interactions with fixed and moving obstacles, facilitating efficient decision-making [105, 106]. Another crucial intermediary layer, particularly relevant for ensuring safe maneuvers and compatibility with closed-loop dynamics, is provided by reference governors [107]. These governors act between a high-level guidance system and the flight controller, modulating commands to prevent hazardous conditions.

Many of these advanced techniques, including reference governors, are set-oriented approaches. The traditional and well-established method for computing admissible sets in this context relies on polyhedral representations, such as half-space (H-representation) or vertex (V-representation) descriptions. While exact for linear systems, polyhedral methods suffer from the "curse of dimensionality": the number of facets or vertices required to describe a set can grow exponentially with the state-space dimension. Consequently, core operations for reachability analysis, such as the Minkowski sum (crucial for propagating uncertainties) and linear

mapping, become computationally intractable for systems of even moderate complexity, hindering their application to complex, high-dimensional problems.

To address this scalability challenge, our approach leverages zonotopes as an alternative set representation [108, 101]. A zonotope is defined compactly by a center vector and a set of generator vectors, offering a significant computational advantage because fundamental operations for reachability are remarkably efficient. For instance, the linear mapping of a zonotope and the Minkowski sum of two zonotopes are computed through simple, low-complexity matrix-vector operations on their generators. The complexity of these operations scales polynomially with the state dimension, in stark contrast to the exponential complexity associated with polyhedra. It enables the tractable analysis of higher-dimensional systems and opens the door to online implementations, which would be infeasible using conventional polyhedral techniques.

In this chapter, we present a reference governor that specifically guarantees the transient response of the closed-loop system remains within bounds and does not violate the pre-defined safe envelope constraints. This work builds upon preliminary results in the thesis. A key novelty of our current approach is the computation of the maximum admissible set using reachability analysis, which enables the construction of a robust safe flight envelope. The proposed reference governor is designed within a linear framework, leveraging the linearized closed-loop dynamics provided by the INDI flight controller (see Figure 6.1). It employs a set-oriented technique with a zonotopic representation (see Figure 6.2) to accelerate computations, a procedure that can be effectively implemented using available software tools such as CORA [100]. Finally, the proposed reference governor strategy, integrated with the cascade INDI flight control, is validated using the MathWorks Parrot Mambo flight simulator.

6.2 Theory

To introduce the concept of a reference governor, a discrete-time linear system representing the closed-loop dynamics is considered:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}_T\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k), \end{aligned} \tag{6.1}$$

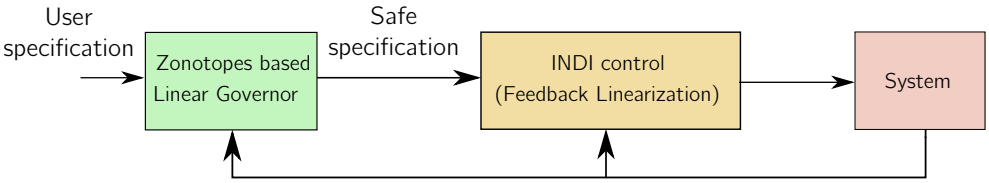


Figure 6.1: Safe flight envelope protection through the use of a linear governor based on the exploitation of zonotopes. The internal controller is a feedback linearizing controller that enables the linear governor.

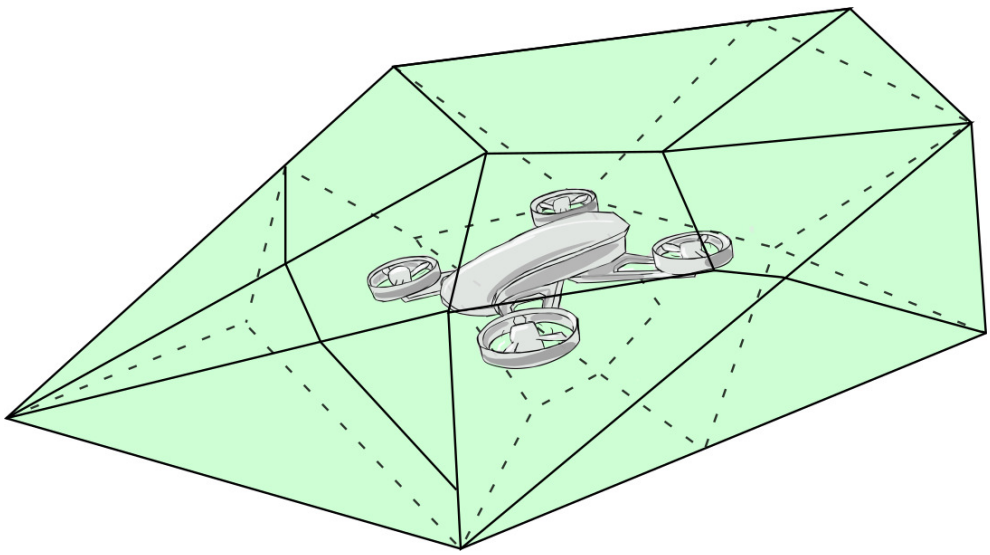


Figure 6.2: UAV in a safe flight envelope with a zonotopic representation.

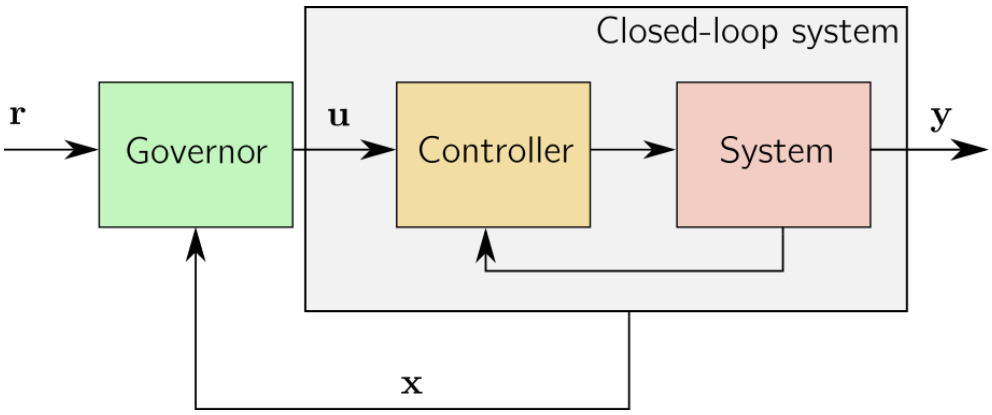


Figure 6.3: Reference governor structure

where $\mathbf{x}(k) \in \mathbb{R}^n$ is the states vector, $\mathbf{u}(k) \in \mathbb{R}^{n_u}$ the input vector of the closed-loop system (the reference), and $\mathbf{y}(k) \in \mathbb{R}^m$ the output. It is assumed that the closed-loop system is stable, i.e., the eigenvalues of $\mathbf{A}_T$ are inside the unit circle. This assumption generally holds as controllers are designed, on the one hand, to ensure system stability, and the other hand, to provide good performance.

The reference governor [109] is an outer layer of control that transforms a reference $\mathbf{r}(k)$ generated by a high-level guidance system into a new reference $\mathbf{u}(k)$ which is more suitable with respect to the dynamics of the closed-loop system and the constraints imposed on the system, as shown in Fig. 6.3.

The governor is based on an optimization problem:

$$\begin{aligned} \min_{\mathbf{u}(k)} \ & \|\mathbf{r}(k) - \mathbf{u}(k)\|_2^{\mathbf{Q}} \\ s.t. \ & (\mathbf{u}(k), \mathbf{y}(k+1)) \in \mathcal{O}_\infty \end{aligned} \tag{6.2}$$

The governor looks for a reference $\mathbf{u}(k)$ which is the closest possible to $r(k) \in \mathbb{R}^m$, the high-level reference, while ensuring that both $\mathbf{u}(k)$ and $y(k)$ belong to the maximum admissible set $\mathcal{O}_\infty$. This set corresponds to all the admissible trajectories originating from given initial conditions $\mathbf{x}(0)$ that lead under constant inputs $\mathbf{g}$ to outputs $\mathbf{y}(k)$ inside a constraint set $\mathcal{Y}$:

$$\mathcal{O}_\infty = \{(\mathbf{u}, \mathbf{x}) : \hat{\mathbf{y}}(k \mid \mathbf{u}, \mathbf{x}(0)) \in \mathcal{Y}, \forall k \in \mathbb{Z}^+\}, \tag{6.3}$$

with $\hat{\mathbf{y}}(k \mid \mathbf{u}, \mathbf{x}(0))$ the prediction of the output using the model in (6.1). In the general case, the constraint set $\mathcal{Y}$ is not convex and is computed using

Monte Carlo techniques [109]. This is why it is proposed to compute a convex tighten sub-set $\tilde{\mathcal{O}}_\infty \subset \mathcal{O}_\infty$ in order to alleviate the computational burden. As the input $\mathbf{u}$ and the initial states $\mathbf{x}(0)$ are bounded and model (6.1) is stable, one can find a value k^* for which $\hat{\mathbf{y}}(k^* \mid \mathbf{u}, \mathbf{x}(0)) = \hat{\mathbf{y}}(\infty \mid \mathbf{u}, \mathbf{x}(0))$.

6.3 Maximum Admissible set

The prediction of the system output can be systematically and efficiently computed using zonotopic reachability. Consider $\mathcal{CZ}_{\mathbf{x}(0)}$ the constrained zonotope of initial conditions and $\mathcal{CZ}_{\mathbf{u}}$ the constrained zonotope of inputs. A reachability algorithm ([100], [110]) can be used to compute the reach set $\mathcal{CZ}_{\mathbf{x}(k^*)}$ and $\mathcal{CZ}_{\mathbf{y}(k^*)}$:

$$\mathcal{CZ}_{\mathbf{x}(k^*)} = \mathbf{A}_T^{k^*} \mathcal{CZ}_{\mathbf{x}(0)} \oplus \sum_{i=0}^{k^*-1}{}^{\oplus} \mathbf{A}_T^i B \mathcal{CZ}_{\mathbf{u}} \tag{6.4}$$

$$\mathcal{CZ}_{\mathbf{y}(k^*)} = \mathbf{C}\mathcal{CZ}_{\mathbf{x}(k^*)} \oplus \mathbf{D}\mathcal{CZ}_{\mathbf{u}}. \tag{6.5}$$

If the predicted set $\mathcal{CZ}_{\mathbf{y}(k^*)}$ is a subset of the constraint set $\mathcal{Y}$, the maximum admissible set is represented by both $\mathcal{CZ}_{\mathbf{y}(k^*)}$ and $\mathcal{CZ}_{\mathbf{u}}$, i.e.,

$$(\mathbf{u}(k), \mathbf{y}(k+1)) \in \mathcal{O}_\infty \iff \begin{cases} \mathbf{u}(k) \in \mathcal{CZ}_{\mathbf{u}} \\ \mathbf{y}(k+1) \in \mathcal{CZ}_{\mathbf{y}(k^*)} \end{cases} \tag{6.6}$$

As the operations performed with the zonotopes tend to increase the number of generators, reducing the order of the resulting zonotopes may be necessary. Reduction techniques are available in the CORA toolbox [100]. This toolbox is a useful library for reachability analysis, which covers different types of set representations for both linear and nonlinear systems, conversion methods, and several reachability techniques.

Thanks to the use of the reachability algorithm, the definition of the admissible set is extended. In the initial definition, the input is considered constant along the trajectory, while in the reachability analysis, the zonotope $\mathcal{CZ}_{\mathbf{u}}$ is constant, so that, for a given trajectory, the inputs contained in this set at each time step can vary.

6.4 Constraint set $\mathcal{Y}$

A linear constraint can be imposed on the output set, reducing the maximum admissible set. The admissible output is given by the intersection of the constraint set $\mathcal{Y}$ and output prediction $\mathcal{CZ}_{\mathbf{y}(*)}$. Constraint sets $\mathcal{Y}$ are generally expressed as linear constraints: $\mathbf{h}^T\mathbf{y} < \mathbf{f}_c$, and in this case, Theorem 1 from [111] allows computing the intersection between a half-space and a constrained zonotope (or a zonotope).

Definition 6.4.1. [111][Constrained zonotope-Half space intersection] Let a constrained zonotope $\mathcal{CZ} = \{\mathbf{c}, \mathbf{G}, \mathbf{A}_c, \mathbf{b}\} \subset \mathbb{R}^n$ intersects a half-space $\mathcal{H} = \{\mathbf{y} \in \mathbb{R}^n \mid \mathbf{h}^T\mathbf{y} < \mathbf{f}_c\}$ then their intersection $\mathcal{CZ}_h = \mathcal{CZ} \cap \mathcal{H}$ is a constrained zonotope of the form:

$$\mathcal{CZ}_h = \left\{\mathbf{c}, \begin{bmatrix}\mathbf{G} & \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{A}_c & \mathbf{0} \\ \mathbf{h}^T\mathbf{G} & \mathbf{dm}/2\end{bmatrix}, \begin{bmatrix}\mathbf{b} \\ \mathbf{f}_c - \mathbf{h}^T\mathbf{c} - \mathbf{dm}/2\end{bmatrix}\right\} \tag{6.7}$$

where $\mathbf{dm} = \mathbf{f}_c - \mathbf{h}^T\mathbf{c} + \sum_{i=1}^{ng} \mid \mathbf{h}^T\mathbf{g}_i \mid$.

Alternatively, the convex set $\mathcal{Y}$ can be expressed in a zonotopic form. The admissible output is easily computed by an intersection i.e.,

$$\mathcal{CZ}_{\mathbf{y}} = \mathcal{CZ}_{\mathbf{y}(k^*)} \cap \mathcal{Y}, \tag{6.8}$$

and

$$(\mathbf{u}(k), \mathbf{y}(k)) \in \mathcal{O}_\infty \iff \begin{cases} \mathbf{u}(k) \in \mathcal{CZ}_{\mathbf{u}} \\ \mathbf{y}(k+1) \in \mathcal{CZ}_{c,\mathbf{y}} \end{cases} \tag{6.9}$$

This definition is however not sufficient. Indeed, the reference governor looks one step ahead in time, and the UAV may arrive near the constraint limit, in a way that no control action in the next time step could avoid a constraint violation. Therefore, it would be interesting to reduce $\mathcal{CZ}_{\mathbf{u}}$ with respect to the constraint set $\mathcal{Y}$.

From expressions (6.4) and (6.5), it is possibly to write:

$$\mathcal{CZ}_{c,\mathbf{y}(k^*)} = \mathbf{CA}_T^{k^*}\mathcal{CZ}_{\mathbf{x}(0)} \oplus \overset{k^*-1}{\underset{i=0}{\oplus}} \mathbf{CA}_T^{i}\mathbf{B}\mathcal{CZ}_{c,\mathbf{u}} \oplus D\mathcal{CZ}_{c,\mathbf{u}}, \tag{6.10}$$

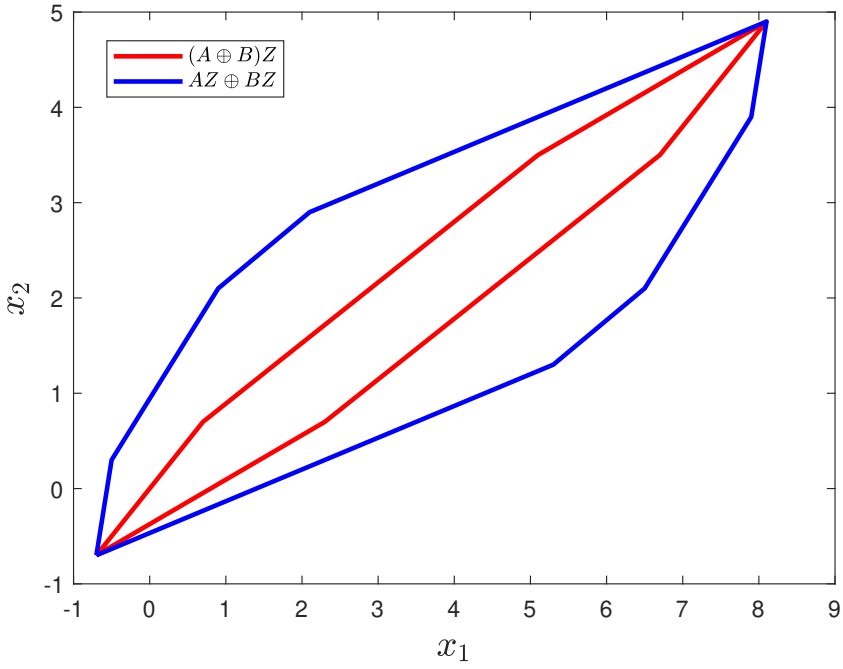


Figure 6.4: Factorization of a zonotope over the Minkowski sum.

where the output set $\mathcal{CZ}_{c,y(k*)}$ and the initial set $\mathcal{CZ}_{\mathbf{x}(0)}$ are known. However, computing the corresponding set $\mathcal{CZ}_{c,v}$ is not an easy task as zonotopes cannot be factorized over the Minkowski sum (see Fig. 6.4).

To overcome this problem, it is proposed to build an approximate input constraint $\mathbf{h}_v^T\mathbf{g} < \mathbf{f}_{c,\mathbf{u}}$ from the constraint on the output $\mathbf{h}^T\mathbf{y} < \mathbf{f}_c$. To this end, the individual impact of each input j on the output is evaluated through a reachability analysis:

$$\mathcal{CZ}_{\mathbf{y},j} = \mathbf{C}\sum_{i=0}^{k^*-1}{}^{\oplus}\, \mathbf{A}_T^i\mathbf{B}\mathcal{CZ}_{\mathbf{u},j} \oplus \mathbf{D}\mathcal{CZ}_{\mathbf{u},j} = \{\mathbf{c}_{\mathbf{y},j}, [\mathbf{g}_{\mathbf{y},j}^{(1)} \dots \mathbf{g}_{\mathbf{y},j}^{(n_{gj})}], [], []\} \quad (6.11)$$

where $\mathcal{CZ}_{\mathbf{g},j}$ has its center at the origin and a generator of the form $\mathbf{g}_{u,j} = [0 \dots 1 \dots 0]^T$ where only the j^{th} element is non zero. The resulting matrix of generators of the output set gives information about the maximum output value reached by the system for a unit input. This information can be compiled into a matrix $\mathbf{P}_j$:

$$\mathbf{P}_j = \sum_{i=1}^{n_{g,j}} \|\mathbf{g}_{\mathbf{y},j}^{(i)}\| \quad (6.12)$$

This matrix links the outputs to the j^{th} input. In first approximation, it is possible to write

$$\Delta\mathbf{y} \approx \mathbf{P}\Delta\mathbf{u}, \;\; \mathbf{P} = [\mathbf{P}_1 \dots \mathbf{P}_r] \quad (6.13)$$

The constraint on the input is computed as:

$$\begin{aligned} \mathbf{h}^T\mathbf{y} < \mathbf{f}_c &\iff \mathbf{h}^T\Delta\mathbf{y} < \mathbf{f}_c - \mathbf{h}^T\mathbf{y}_0 \\ &\iff \mathbf{h}^T\mathbf{P}\Delta\mathbf{u} < \mathbf{f}_c - \mathbf{h}^T\mathbf{y}_0 \\ &\iff \mathbf{h}^T\mathbf{P}\mathbf{u} < \mathbf{f}_c - \mathbf{h}^T\mathbf{y}_0 + \mathbf{h}^T\mathbf{P}\mathbf{u}_0, \end{aligned} \quad (6.14)$$

with $\mathbf{y}_0$ and $\mathbf{u}_0$ a steady-state point of the system.

If the constraint set $\mathcal{Y}$ is given in a zonotopic form, the same matrix $\mathbf{P}$ can be used to compute a zonotope which will intersect $\mathcal{CZ}_{\mathbf{u}}$.

Consider a numerical example to illustrate the concepts, e.g.,

$$\mathbf{A}_T = \begin{bmatrix} -2 & 1 \\ 3 & -2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 1 \\ 2 & 6 \end{bmatrix},$$
$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

which represents a discrete-time system with a sampling period of 1 second. The zonotopic reachability is computed up to 3 seconds, according to equation (6.4) with zero initial condition and the input set $\mathcal{CZ}_{\mathbf{u}}$ shown in Fig. 6.5b (blue zonotope). The resulting zonotope $\mathcal{CZ}_{\mathbf{y}}$ is shown in Fig. 6.5a (blue zonotope). A constraint $\mathbf{h}$ on the output is considered:

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}^T \mathbf{y} < 5.$$

This constraint corresponds to the hyperplane $\mathcal{Y}$ (red in subfigure a) and intersects $\mathcal{CZ}_{\mathbf{y}}$ to produce the admissible set $\mathcal{CZ}_{c,\mathbf{y}}$ (green constrained zonotope in subfigure a). Applying the above-mentioned procedure, a projection matrix $\mathbf{P}$ is constructed, and a constraint on the input $\mathbf{u}$:

$$\mathbf{P} = \begin{bmatrix} 10 & 22 \\ 14 & 45 \end{bmatrix}, \quad \begin{bmatrix} 4 \\ 23 \end{bmatrix}^T \mathbf{u} < 5.$$

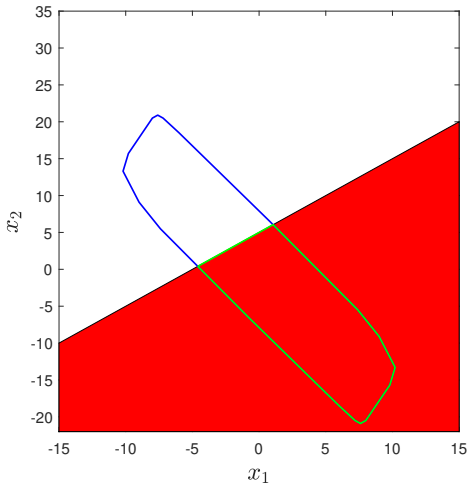
The hyperplane V intersects the initial zonotope $\mathcal{CZ}_{\mathbf{u}}$ and produces the constrained input zonotope $\mathcal{CZ}_{c,\mathbf{u}}$ (green in Fig. 6.5b).

6.5 Disturbances

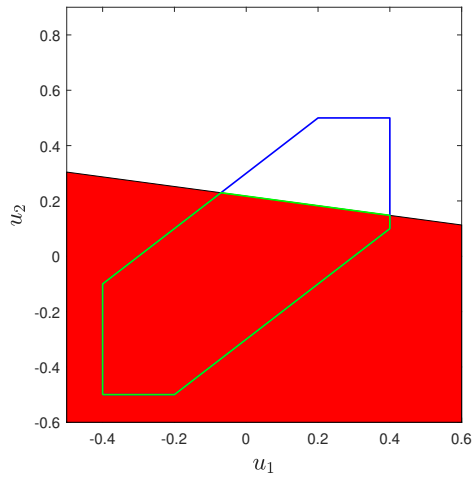
The reference governor allows introducing bounded disturbances in the strategy quite easily:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}_T\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{E}_p\mathbf{w}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) + \mathbf{F}_p\mathbf{w}(k), \end{aligned} \tag{6.15}$$

with unknown bounded disturbance $\mathbf{w}(k)$. The maximum admissible set accepts the same definition but includes disturbances in the predicted output:



(a) Intersection $\mathcal{CZ}_{c,y}$ (green line) of a constrained zonotope $\mathcal{CZ}_{\mathbf{y}}$ (blue line) with a hyper-plan $\mathcal{Y}$ (red).



(b) Initial input constrained zonotope $\mathcal{CZ}_{\mathbf{u}}$ (blue line) and constraint input $\mathcal{CZ}_{c,\mathbf{u}}$ (green line).

Figure 6.5: Numerical example

$$\mathcal{O}_\infty = \{(\mathbf{u}, \mathbf{x}) : \hat{\mathbf{y}}(k \mid \mathbf{u}, \mathbf{x}(0), \mathbf{w}) \in \mathcal{Y}, \forall k \in \mathcal{CZ}^+\}, \tag{6.16}$$

and the zonotopic reachability is extended as

$$\mathcal{CZ}_{\mathbf{x}(k^*)} = \mathbf{A}_T^{k^*} \mathcal{CZ}_{\mathbf{x}(0)} \oplus \sum_{i=0}^{k^*-1}{}^{\oplus} \mathbf{A}_T^i \mathbf{B} \mathcal{CZ}_{\mathbf{u}} \oplus \sum_{i=0}^{k^*-1}{}^{\oplus} \mathbf{A}_T^i \mathbf{E}_p \mathcal{CZ}_{\mathbf{w}} \tag{6.17}$$

$$\mathcal{CZ}_{\mathbf{y}(k^*)} = \mathbf{C} \mathcal{CZ}_{\mathbf{x}(k^*)} \oplus \mathbf{D} \mathcal{CZ}_{\mathbf{u}} \oplus \mathbf{F}_p \mathcal{CZ}_w, \tag{6.18}$$

with $\mathcal{CZ}_w$ the zonotope set of bounded disturbances.

6.6 Safety margins

To account for the modeling simplifications and uncertainties inherent in the system model, the pre-computed maximum admissible set is reduced using safety factors. The selection of these factors is a critical step to bridge the gap between the idealized model and the physical system's behavior.

A practical and effective method for tuning these margins is to analyze the closed-loop system's performance empirically.

Specifically, the maximum observed overshoot in the output and input signals during operation near the original constraint boundaries serves as a direct indicator of the potential for constraint violation. This overshoot encapsulates the effects of unmodeled dynamics, delays, and disturbances not captured during the set's computation. By choosing safety factors that create a margin slightly larger than the maximum anticipated overshoot, a robust buffer is established:

$$\mathcal{CZ}_{c,\mathbf{y}} = \{\mathbf{c}_\mathbf{y}, \mathbf{S}_\mathbf{y}, \mathbf{G}_\mathbf{y}, \mathbf{A}_\mathbf{y}, \mathbf{b}_\mathbf{y}\} \tag{6.19}$$

$$\mathcal{CZ}_{c,\mathbf{u}} = \{\mathbf{c}_\mathbf{u}, \mathbf{S}_\mathbf{u}\mathbf{G}_\mathbf{u}, \mathbf{A}_\mathbf{u}, \mathbf{b}_\mathbf{u}\} \tag{6.20}$$

where $\mathbf{S}_\mathbf{y} \in \mathbb{R}^{n\times n}$ and $\mathbf{S}_\mathbf{u} \in \mathbb{R}^{n_u\times n_u}$ are diagonal with each element comprised between 0 and 1, i.e., $0 < s_{y,i,i} < 1$, $0 < s_{g,j,j} < 1, \forall i = 1, \dots, n,\ \forall j = 1, \dots, n_u$.

6.7 Reference governor design for the Mambo

Feedback linearization controllers provide a linear closed-loop structure (basically a set of integrators) that decouples the dynamics of each state of the drone. In this section, we show how the design of the reference governor can be achieved when the INDI strategy is considered for the baseline control. The model used for the reachability analysis is restricted to references specified by the user and their associated states.

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}_T\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{E}_p\mathbf{w}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) + \mathbf{F}_p\mathbf{w}(k) \end{aligned} \tag{6.21}$$

$$\mathbf{u}(k) = \begin{bmatrix} p_{\mathbf{x},ref,k} \\ p_{y,ref,k} \\ p_{z,ref,k} \\ \psi_{ref,k} \end{bmatrix},\ \mathbf{x}(k) = \begin{bmatrix} p_{\mathbf{x},k} \\ p_{y,k} \\ p_{z,k} \\ \psi_k \\ v_{\mathbf{x},k} \\ v_{y,k} \\ v_{z,k} \\ \gamma_k \end{bmatrix},\ \mathbf{w}_k = \begin{bmatrix} f_{\mathbf{x},k} \\ f_{y,k} \\ f_{z,k} \\ m_{z,k} \end{bmatrix} \tag{6.22}$$

Moreover, it is assumed that all the states are available through measurements or observers. Therefore, the **C** matrix is an identity matrix and **D** is a zero matrix.

In UAVs, the primary disturbances originate from wind and aerodynamic effects. Wind can be seen as a set of forces and torques that apply to the drone structure, while aerodynamic effects are linked to the high-speed rotation of rotor blades. As shown in [112], those effects are nonlinear and impact each rotor-produced trust. Accordingly, the disturbances are defined as follows

$$\mathbf{E}_p = \begin{bmatrix} \mathbf{0}_{4\times1} & \mathbf{0}_{4\times1} & \mathbf{0}_{4\times1} & \mathbf{0}_{4\times1} \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}}, \end{bmatrix} \tag{6.23}$$

and F is a zero-matrix. For computer implementation, these equations are expressed in discrete time.

To compute the admissible set, it is required to define bounds on the initial conditions, the inputs, and the disturbances. In our implementation, the initial conditions are restricted to the origin. Indeed the closed-loop system is stable, and the effect of the initial conditions vanishes over time. The inputs, i.e., the references for the spatial location and the orientation along the z-axis, have no physical limits in an open environment. However, the controller is designed for responses to unitary step inputs, and the reference variations are limited accordingly:

$$\begin{bmatrix} p_x - 1 \\ p_y - 1 \\ p_z - 1 \\ \psi - 1 \end{bmatrix} < \begin{bmatrix} p_{\mathbf{x},ref} \\ p_{y,ref} \\ p_{z,ref} \\ \psi_{ref} \end{bmatrix} < \begin{bmatrix} p_x + 1 \\ p_y + 1 \\ p_z + 1 \\ \psi + 1 \end{bmatrix} \tag{6.24}$$

This procedure allows the input of the INDI controller to stay within the design range, and in turn, avoid input saturation. The zonotope of input is expressed in variations with respect to the actual states, i.e.,

$$\mathcal{CZ}_{\Delta\mathbf{u}} = \left\langle \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, [], [] \right\rangle_{\mathcal{CZ}} \tag{6.25}$$

Knowing the initial set and the set of inputs, equations (6.4) and (6.5) can be used to compute the output zonotope $\mathcal{CZ}_{\Delta y}$.

The maximum admissible set is updated online:

$$\mathcal{CZ}_{\mathbf{u}} = \mathcal{CZ}_{\Delta\mathbf{u}} + \begin{bmatrix} p_{\mathbf{x},k} \\ p_{y,k} \\ p_{z,k} \\ \psi_k \end{bmatrix}, \quad \mathcal{CZ}_{\mathbf{y}} = \mathcal{CZ}_{\Delta\mathbf{y}} + \begin{bmatrix} p_{\mathbf{x},k} \\ p_{y,k} \\ p_{z,k} \\ \psi_k \\ v_{\mathbf{x},k} \\ v_{y,k} \\ v_{z,k} \\ \gamma_k \end{bmatrix} \tag{6.26}$$

Finally, a zonotope can be built for the disturbances and since the drone is flying indoors, only small perturbations can occur.

The projection of the maximum admissible set on several 2D phase planes is shown in Fig. 6.6.

A similar approach can be followed in the case of classic feedback linearization. The only differences are the matrices of the closed-loop model.

Optimization problem

The optimization problem 6.2 is now expressed in matrix form, considering the use of zonotopes. First, the input $\mathbf{u}(k)$ must stay inside a constrained zonotope $\mathcal{CZ}_{c,\mathbf{u}} = \{c_{\mathbf{u}}, G_{\mathbf{u}}, A_{c,\mathbf{u}}, b_{\mathbf{u}}\}$

$$\mathbf{u}(k) = \mathbf{c}_{\mathbf{u}} + \mathbf{G}_{\mathbf{u}}\boldsymbol{\beta}_{\mathbf{u}(k)} \in \mathcal{CZ}_{c,\mathbf{u}} \iff \begin{cases} \|\boldsymbol{\beta}_{\mathbf{u}(k)}\|_\infty < 1 \\ \mathbf{A}_{c,\mathbf{u}}\boldsymbol{\beta}_{\mathbf{u}(k)} = \mathbf{b}_{\mathbf{u}} \end{cases} \tag{6.27}$$

If one considers $\boldsymbol{\beta}_{\mathbf{u}(k)}$ as a new optimization variable, then linear equality constraints and box constraints are added to the quadratic optimization

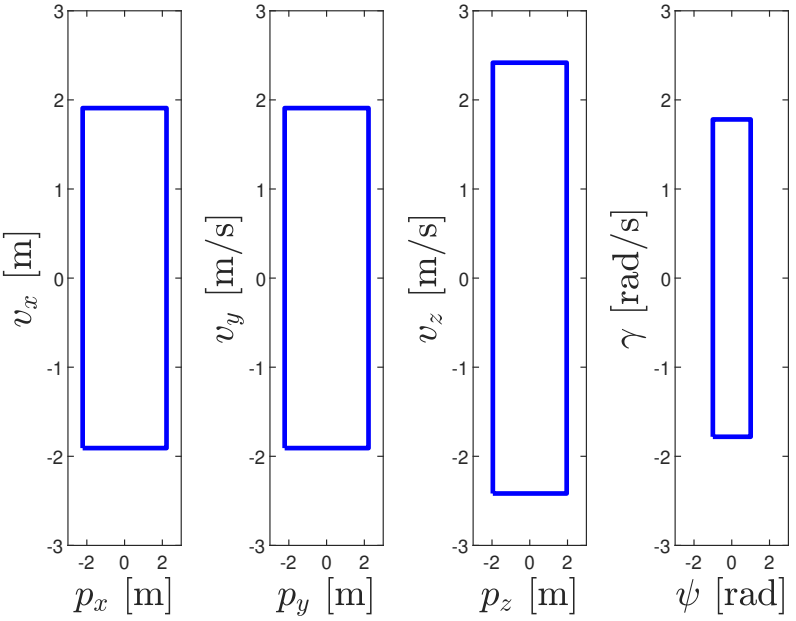


Figure 6.6: Projection of the maximum admissible zonotope on several 2D phase planes - INDI case

problem:

$$\begin{bmatrix} \mathbf{I}_{n_u \times n_u} & -\mathbf{G}_\mathbf{u} \\ \mathbf{0}_{n_{c,u} \times n_u} & \mathbf{A}_{c,\mathbf{u}} \end{bmatrix} \begin{bmatrix} \mathbf{u}(k) \\ \boldsymbol{\beta}_{\mathbf{u}(k)} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_\mathbf{u} \\ \mathbf{b}_\mathbf{u} \end{bmatrix}, \tag{6.28}$$

$$-\mathbf{1}_{n_{g,c} \times 1} < \boldsymbol{\beta}_{\mathbf{u}(k)} < \mathbf{1}_{n_{g,c} \times 1} \tag{6.29}$$

Second, the output prediction at the next time step $\mathbf{y}(k+1)$ must lie inside a constrained zonotope $\mathcal{CZ}_{c,\mathbf{y}} = \{\mathbf{c}_\mathbf{y}, \mathbf{G}_\mathbf{y}, A_{c,\mathbf{y}}, \mathbf{b}_\mathbf{y}\}$

$$\mathbf{y}(k+1) = \mathbf{C}\mathbf{A}_T\mathbf{x}(k) + \mathbf{C}\mathbf{B}\mathbf{u}(k) + \mathbf{D}\mathbf{u}(k) \tag{6.30}$$

$$= \mathbf{c}_\mathbf{y} + \mathbf{G}_\mathbf{y}\boldsymbol{\beta}_{\mathbf{y}(k+1)} \in \mathcal{CZ}_{c,y} \iff \begin{cases} \|\boldsymbol{\beta}_{\mathbf{y}(k+1)}\|_\infty < 1 \\ \mathbf{A}_{c,\mathbf{y}}\boldsymbol{\beta}_{\mathbf{y}(k+1)} = \mathbf{b}_\mathbf{y} \end{cases} \tag{6.31}$$

$\mathbf{x}(k)$ is known as it is provided by measurements or state observers. Again, $\boldsymbol{\beta}_{\mathbf{y}(k+1)}$ is considered as a new optimization variable and

$$\begin{bmatrix} (\mathbf{C}\mathbf{B} + \mathbf{D}) & -\mathbf{G}_\mathbf{y} \\ \mathbf{0}_{n_{c,y} \times n_u} & \mathbf{A}_{c,\mathbf{y}} \end{bmatrix} \begin{bmatrix} \mathbf{u}(k) \\ \boldsymbol{\beta}_{\mathbf{y}(k+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_\mathbf{y} - \mathbf{C}\mathbf{A}_T\mathbf{x}(k) \\ \mathbf{b}_\mathbf{y} \end{bmatrix}, \tag{6.32}$$

$$-\mathbf{1}_{n_{g,y} \times 1} < \boldsymbol{\beta}_{\mathbf{y}(k+1)} < \mathbf{1}_{n_{g,y} \times 1} \tag{6.33}$$

Introducing new variables in the optimization problem makes it computationally more expensive. It is therefore essential to consider their number, which is given by the generators of the resulting constrained zonotopes, and the number of constraints applied to those generators. Techniques for reducing the order or the number of constraints of such representations are available in [113], [101]. Moreover, the weight given to those new variables in the optimization problem can modify the reference governor dynamics.

Eventually, we can discuss the feasibility of the optimization problem by asking whether a solution always exists and what the requirements are. In the case of the reference governor, it is clear that the necessary condition is the non-emptiness of the two zonotopes; that is, there must exist a non-empty set from which the optimization variables can be selected. This non-emptiness condition can be easily verified through (5.2).

6.8 Tests of the control structure

In this section, the zonotopic reference governor is tested using the Mathworks simulator for the Mambo mini drone in indoor conditions with noisy measurements. Both feedback linearization strategies are considered and compared. Fig. 6.7 recalls the system response under reference steps without reference governor for both feedback linearization methods. The yaw angle ψ reaches the setpoint in about 2 seconds while the elevation setpoint is reached in about 3 seconds. The positions p_x, and p_y take a longer time to settle, as they are regulated in a second control layer. Fig. 6.8 shows the system responses with the reference governor. Note that the reference governor acts on $p_{\mathbf{x},ref}$, $p_{y,ref}$ and $p_{z,ref}$ in order to comply with the admissible set, whereas ψ_{ref} is not affected. As the quadcopter moves the references are adapted to eventually reach the initially targeted position. This results into a smooth, somewhat slower transient. Note that both methods provide equivalent results, as performances are mostly determined by the tuning of the linear controllers in cascade with feedback linearization.

Considering the same reference trajectory, constraints are now introduced. First, a constraint on the p_x axis is considered:

$$p_x < 0.5 \tag{6.34}$$

The 3D trajectory is shown in Fig. 6.9a while a projection in the $p_x - p_y$ plane is shown in Fig. 6.9b. The reference governor adapts the reference so as to ensure the respect of the constraint considering the system dynamics. The new reference is close to the constraint but not equal to it, as overshooting or disturbances could push the system over the admissible bound.

Velocity constraints can also be considered:

$$v_x < 0.05 \tag{6.35}$$

Fig. 6.10 shows the time evolution of the velocities (for the same trajectory tracking problem) in three cases: (a) no constraint and the velocity evolves freely, (b) a constraint is applied limiting the velocity amplitude in the INDI case, (c) a constraint is applied limiting the velocity amplitude in the classic feedback linearization case. The latter case displays a more oscillating velocity, due to the use of the jerk in the linear control.

The main contribution of this work is the development of a reference governor scheme using zonotopic sets. This strategy is applied to a mini

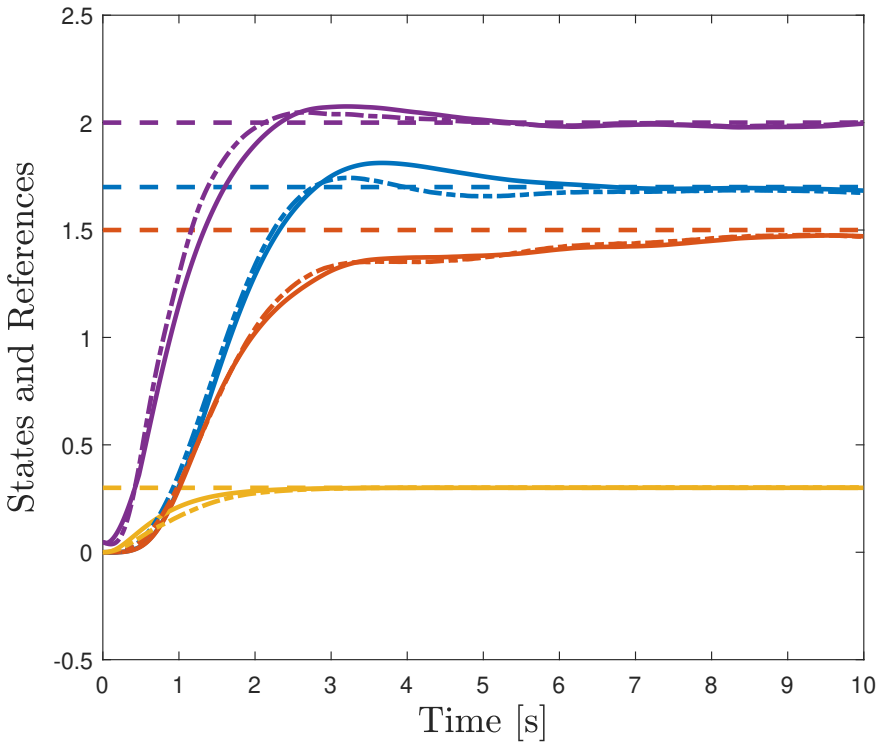


Figure 6.7: Unconstrained responses of the Mambo mini drone to setpoint steps without the reference governor - classic feedback linearization: dotted lines - INDI: continuous lines - references: dashed lines - purple: p_z, blue: p_x, orange: p_y, yellow: ψ.

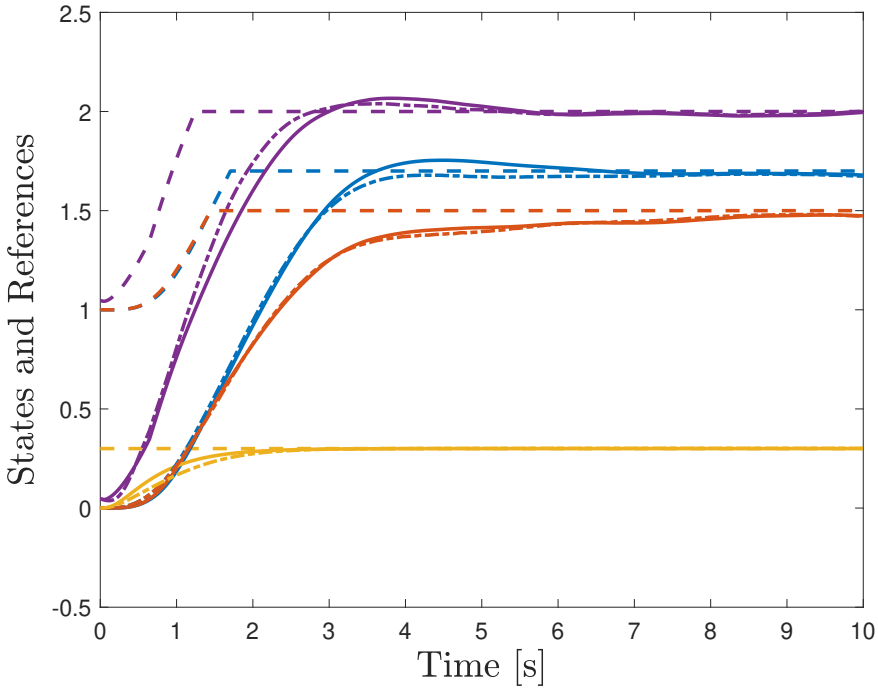
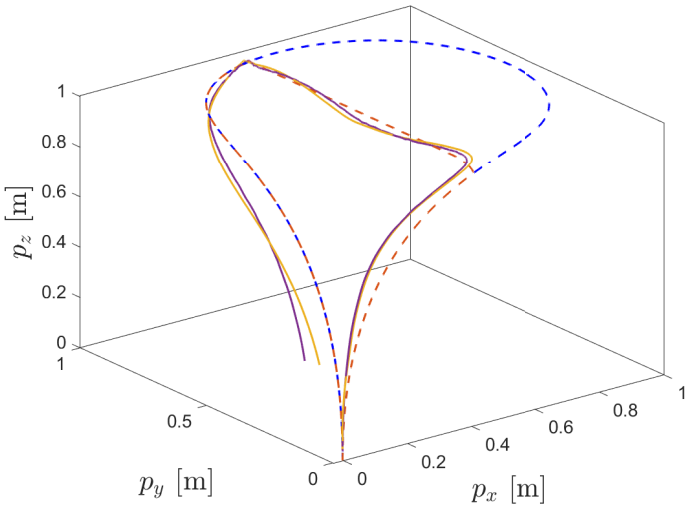
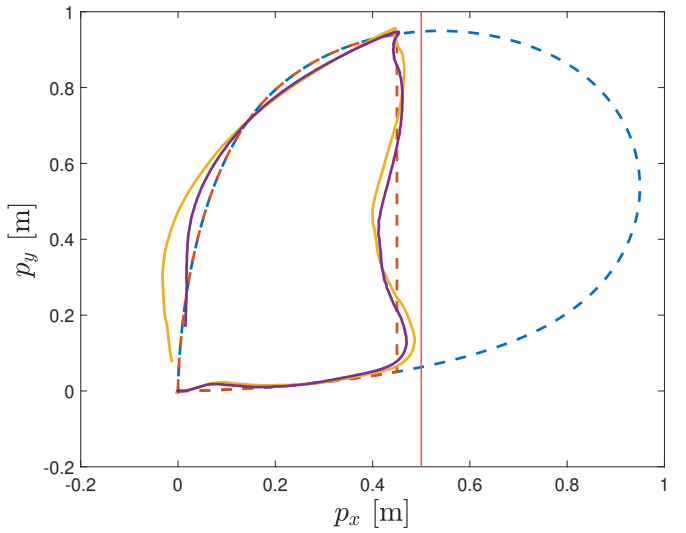


Figure 6.8: Unconstrained responses of the Mambo mini drone to setpoint steps with the reference governor - classic feedback linearization: dotted lines - INDI: continuous lines - references: dashed lines - purple: p_z, blue: p_x, orange: p_y, yellow: ψ.



(a) 3D trajectory.



(b) Projection in the $\mathbf{x} - y$ plane.

Figure 6.9: Constrained response (constraint $\mathbf{x} < 0.5$) of the Mambo mini drone - Trajectory (INDI - yellow line, classic Feedback linearization - purple line) of the quadcopter with the initial reference (blue dotted line) and the updated reference (orange dotted line).

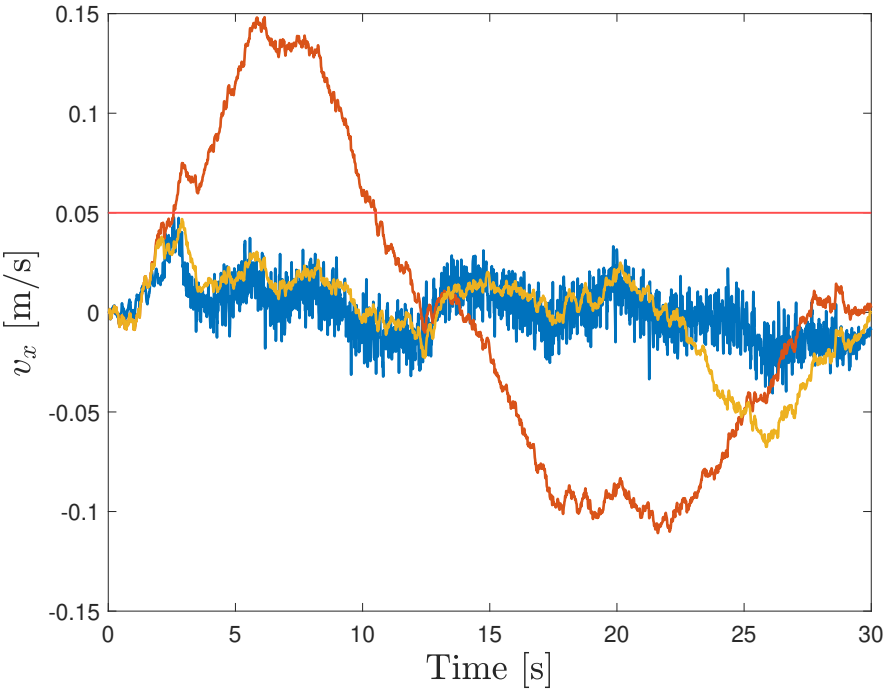


Figure 6.10: Velocities in the **x** direction without the constraint (red line) and with the constraint $v_x < 0.05$ (INDI: orange line and classic feedback linearization: blue line.

drone whose flight controller is designed using two feedback linearization techniques, e.g., a classic approach, leading to an augmented system that requires the estimation of additional variables (jerks), and an incremental nonlinear dynamic inversion technique that requires high sampling rate to achieve robustness. Both provide a linear closed-loop system and the reference governor can therefore be developed in a linear framework.

Simulation tests demonstrate the feasibility of the approach and promising results, with the satisfaction of the imposed constraints.

In terms of performance, both linearization techniques come in a close match, but our preference goes to the INDI controller as it avoids the burden of considering the jerks. The INDI controller provides partial linearization, depending on the quality of the underlying dynamic model and the sampling rate of the measurement units. Possible synchronization issues have been reported in [114] but we have not observed such issues in the Parrot Mambo evaluation.

The definition of the reference governor could be extended to multiple prediction steps, and, as a further extension, linear robust model predictive control will be considered, to enable multistep prediction and enhance robustness linked to partial linearization in the inner loop.

Chapter 7

Tube-Based Model Predictive Control

7.1 Introduction

In this chapter, we aim to develop a robust control methodology for multi-rotor aircraft, specifically addressing the challenges posed by external disturbances and model uncertainties. After the application of robust feedback linearization to create an approximately linearized model, which inherently introduces model uncertainties, the focus shifts to designing a resilient outer-loop control strategy.

To robustly address the pivotal issue caused by disturbances and model uncertainties, research has increasingly turned toward robust control methodologies. In the realm of MPC, three primary deterministic formulations have been developed to handle bounded uncertainties: minimax MPC, multi-stage MPC, and tube-based MPC.

Minimax MPC is a worst-case approach that aims to minimize the performance cost under the most unfavorable realization of uncertainty over the prediction horizon [115, 116]. This formulation guarantees robust stability and constraint satisfaction but often leads to highly conservative and computationally intractable problems, as the optimization needs to account for all possible disturbance sequences.

Multi-stage MPC (also known as scenario-based MPC) handles uncertainties by modeling their evolution as a discrete tree of scenarios [117, 118]. At each stage, the control action is optimized for a set of possible

future uncertainty realizations. While this approach can be less conservative than minimax MPC, its computational complexity grows exponentially with the length of the prediction horizon and the number of uncertain parameters, making it challenging for real-time applications with continuous disturbances.

Tube-based MPC offers a practical alternative to these methods by decoupling the complex robust control problem into two computationally efficient components. It achieves this by defining a tube of trajectories around a nominal, disturbance-free trajectory. This tube encapsulates all possible real system trajectories under the influence of bounded disturbances. This approach [119, 120, 121], effectively balances robustness with computational tractability, making it well-suited for systems with fast dynamics like UAVs.

Tube MPC has found diverse and growing applications across various domains. For instance, it has been successfully employed in the design of automotive cruise control systems [122], for the robust control of spacecraft [123], and in managing electrical vehicle safety systems [124]. Within the realm of Unmanned Aerial Vehicles (UAVs), Tube MPC has been utilized to handle uncertain linear dynamics [125] and to manage UAVs carrying payloads, which can be modeled as disturbances [126]. Beyond robotics and aerospace, recent original studies have also explored Tube MPC applications in fields such as pharmaceutical production, particularly for bioprocess models exhibiting large uncertainties [127].

The implementation of these robust MPC techniques often relies on the concept of invariant sets. In this chapter, a tube-based control strategy is implemented as an outer loop controller, utilizing constrained polynomial zonotopes as state and input constraint sets. The reachability approach, using this same advanced representation, is exploited to determine the safe flight envelope, which serves as a primary constraint within the tube-based control framework. This approach specifically addresses the application of Tube MPC to an approximately linearized model that accounts for internal uncertainties and disturbances, while effectively integrating computationally efficient zonotopic reachable sets.

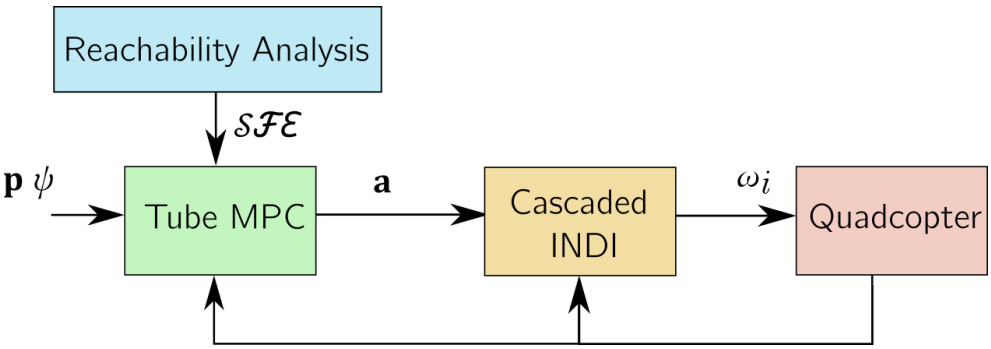


Figure 7.1: Embedded control of the multirotor aircraft: A feedback linearization method in cascade with Tube-MPC - the references for feedback linearization are the accelerations $\mathbf{a}$ and the references of Tube MPC are the positions $\mathbf{p}$ and the yaw angle ψ.

7.2 Principle

The control scheme should optimally track a predefined aircraft trajectory under constraints while being robust to disturbances. To achieve these objectives, adapted to the constraints of an embedded multirotor aircraft system, a multi-layered control approach is adopted, as shown in Fig. 7.1. The procedure is divided into two parts. A robust feedback linearization aims to establish an almost linear closed-loop system, as shown in previous chapter, onto which a Tube MPC is applied to face various challenges, including constraint handling and robustness to external disturbances, model uncertainties, and partial linearization.

7.3 Theory

The tube paradigm is the conceptualization of the robust MPC objective: finding an invariant set of the error system defined by the difference between the nominal system and its true disturbed realization. Tube-based MPC therefore aims to maintain the real system subject to disturbances in a bounded vicinity of an assumed optimal nominal trajectory, as illustrated in Fig. 7.2, [119, 121].

The system to be controlled is modeled by the following discrete-time

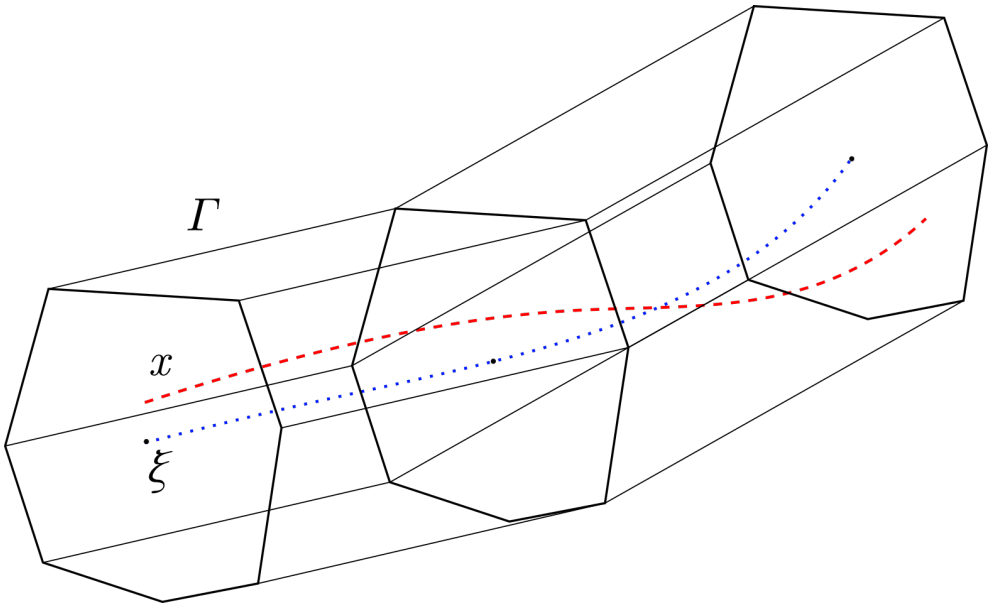


Figure 7.2: Tube-based robust model predictive control principle: the computed nominal trajectory (blue line) is the reference of an ancillary controller that drives and encompasses the real system trajectory (red line) in a tube defined by zonotopic regions.

nonlinear time-invariant dynamics subject to disturbances:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k)) \tag{7.1}$$

where $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R}^q \to \mathbb{R}^n$ is a continuous function representing the system dynamics, $\mathbf{x}(k) \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state vector belonging to the corresponding closed set $\mathcal{X}$, $\mathbf{u}(k) \in \mathcal{U} \subseteq \mathbb{R}^r$ is the input vector belonging to the closed set $\mathcal{U}$ and $\mathbf{w}(k) \in \mathcal{W} \subseteq \mathbb{R}^q$ is the model unknown disturbance vector, assumed to be bounded by the vertices of the closed set $\mathcal{W}$. The sets $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{W}$ are compact and $\mathcal{W}$ contains zero.

A nominal, disturbance-unaffected, model is associated with system (7.1):

$$\boldsymbol{\xi}(k+1) = \mathbf{f}(\boldsymbol{\xi}(k), \boldsymbol{\nu}(k), \mathbf{0}) \tag{7.2}$$

where $\boldsymbol{\xi}(k) \in \mathbb{R}^n$ is the nominal state vector and $\boldsymbol{\nu}(k) \in \mathbb{R}^r$ the nominal input vector. The ancillary controller computes the closed-loop control law $\mathbf{u}(k)$ using the optimal state $\boldsymbol{\xi}(k)$ and input trajectories $\boldsymbol{\nu}(k)$ as references and considering the available disturbed state $\mathbf{x}(k)$, e.g.,

$$\mathbf{u}(k) = \boldsymbol{\varphi}(\mathbf{x}(k), \boldsymbol{\xi}(k), \boldsymbol{\nu}(k)) \tag{7.3}$$

in such a way that the real system trajectory remains in a sufficiently close neighborhood of the nominal system trajectory, i.e. the following error system remains in a set $\mathcal{L}$:

$$\begin{aligned}
\mathbf{e}(t+1) &= \mathbf{x}(t+1) - \boldsymbol{\xi}(t+1) && (7.4)\\
&= \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k)) - \mathbf{f}(\boldsymbol{\xi}(k), \boldsymbol{\nu}(k), \mathbf{0}) && (7.5)\\
&= \mathbf{f}(\mathbf{x}(k), \boldsymbol{\varphi}(\mathbf{x}(k), \boldsymbol{\xi}(k), \boldsymbol{\nu}(k)), \mathbf{w}(k)) - \mathbf{f}(\boldsymbol{\xi}(k), \boldsymbol{\nu}(k), 0) && (7.6)\\
&\in \mathcal{L} && (7.7)
\end{aligned}$$

where $\mathcal{L}$ is a robust controllable invariant set [121].

Additional constraint sets Ξ and $\mathcal{V}$ for the nominal system can be considered to enforce and modulate the size of the resulting tube around the nominal trajectory.

Figure 7.3 sketches the overall structure with the nominal controller working as an open-loop optimal controller, which computes ideal trajectories feeding the ancillary controller which drives the real system close to the ideal path.

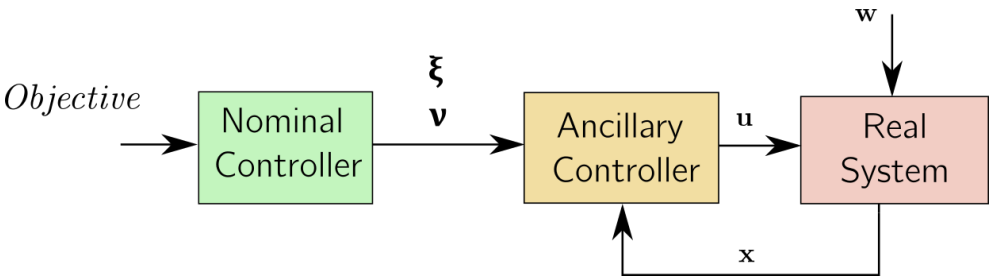


Figure 7.3: Tube model predictive control with the nominal and ancillary controllers

Application to the aircraft system

To implement Tube MPC, the discrete-time representation of the linear system resulting from the application of feedback linearization to the aircraft dynamics

$$\boldsymbol{\xi}(k+1) = \mathbf{A}_T \boldsymbol{\xi}(k) + \mathbf{B}\boldsymbol{\nu}(k) \tag{7.8}$$

with $\boldsymbol{\xi} = [p_x \ p_y \ p_z \ \psi \ v_x \ v_y \ v_z]^T$ and $\boldsymbol{\nu} = [a_x \ a_y \ a_z \ a_\psi]^T$ is used by the nominal controller. It is important to note that this model considers only 7 state variables instead of the 12 variables in the original model as the application of cascaded INDI makes the angular velocities ($\boldsymbol{\omega}_b$) and the two Euler angles (ϕ, θ) fast variables, whose dynamics can be neglected in the design of the outer loops. We solve the following nonlinear programming problem:

$$\begin{aligned}
\underset{\boldsymbol{\nu},\boldsymbol{\xi}(0)}{min} \ & \sum_{i=1}^{N-1} (\boldsymbol{\xi}(i) - \boldsymbol{\xi}_e(i))^T \mathbf{Q}_n (\boldsymbol{\xi}(i) - \boldsymbol{\xi}_e(i)) + (\boldsymbol{\nu}(i) - \boldsymbol{\nu}_e(i))^T \mathbf{R}_n (\boldsymbol{\nu}(i) - \boldsymbol{\nu}_e(i)) \\
& + (\boldsymbol{\xi}(N) - \boldsymbol{\xi}_e(N))^T \mathbf{H}_n (\boldsymbol{\xi}(N) - \boldsymbol{\xi}_e(N)) \qquad (7.9) \\
s.t. \ & \boldsymbol{\xi}(i+1) = \mathbf{A}_T \boldsymbol{\xi}(i) + \mathbf{B}\boldsymbol{\nu}(i), \\
& \boldsymbol{\xi}(i) \in \Xi, \ \ \boldsymbol{\nu}(i) \in \mathcal{V}, \\
& \boldsymbol{\xi}(0) \in \mathbf{x}_0 \oplus \Xi
\end{aligned}$$

where $\boldsymbol{\xi}_e$ and $\boldsymbol{\nu}_e$ are respectively the state and input reference trajectories. $\mathbf{Q}_n$, $\mathbf{R}_n$, $\mathbf{H}_n$ are the weight matrices and Ξ and $\mathcal{V}$ are respectively the nominal state and input compact sets.

The state and input sequences resulting from the nominal optimal problem resolution read:

$$\boldsymbol{\xi}^* := \{\boldsymbol{\xi}^*(0), \boldsymbol{\xi}^*(1)..., \boldsymbol{\xi}^*(N)\}, \quad \boldsymbol{\nu}^* := \{\boldsymbol{\nu}^*(0), \boldsymbol{\nu}^*(1)..., \boldsymbol{\nu}^*(N)\}. \tag{7.10}$$

The ancillary MPC aims to maintain the real system states and inputs as close as possible to the nominal optimal trajectories $\boldsymbol{\xi}^*$ and $\boldsymbol{\nu}^*$. The corresponding cost criterion is minimized in the following nonlinear programming problem:

$$\begin{aligned} \underset{\mathbf{u}}{min} \quad & \sum_{i=1}^{N-1} (\mathbf{x}(i) - \boldsymbol{\xi}^*(i))^T \mathbf{Q}_a (\mathbf{x}(i) - \boldsymbol{\xi}^*(i)) + (\mathbf{u}(i) - \boldsymbol{\nu}^*(i))^T \mathbf{R}_a (\mathbf{u}(i) - \boldsymbol{\nu}^*(i)) \\ & + (\mathbf{x}(N) - \boldsymbol{\xi}^*(N))^T \mathbf{H}_a (\mathbf{x}(N) - \boldsymbol{\xi}^*(N)) \\ s.t. \quad & \mathbf{x}(i+1) = \mathbf{A}_T \mathbf{x}(i) + \mathbf{B}\mathbf{u}(i), \\ & \mathbf{x}(i) \in \mathcal{X}, \ \ \mathbf{u}(i) \in \mathcal{U}, \\ & \mathbf{x}(0) = \mathbf{x}_0 \end{aligned} \tag{7.11}$$

with $\mathbf{x}$ and $\mathbf{u}$ being the real system state and input vectors. $\mathbf{Q}_a$, $\mathbf{R}_a$, $\mathbf{H}_a$ are the weight matrices, which may differ from the one used in the nominal control problem, enabling to restrain the tube around specific state/input variables (or conversely, to widen the tube). The compact sets $\mathcal{X}$ and $\mathcal{U}$ are assumed to respectively include the sets Ξ and $\mathcal{V}$. [119] proposes the following simple procedure to define the tightened constraint sets:

$$\Xi = \lambda_1 \mathcal{X} \tag{7.12a}$$

$$\mathcal{V} = \lambda_2 \mathcal{U} \tag{7.12b}$$

where λ_1 and λ_2 are contained in [0 1].

The resulting tube set $\mathcal{L}$ is not defined explicitly in the MPC problem because its determination is unfortunately difficult in the case of nonlinear control problems [119]. The size of the tube set therefore remains undetermined but can be approximated, for adequately chosen sets Ξ and $\mathcal{V}$, following a Monte-Carlo analysis. It is also important to mention that a wrong choice of these sets, i.e. the violation of these soft constraints, does not prevent Tube MPC from converging to a specific tube set even if it may however affect the ancillary controller performance. The following section provides a better insight into how these state and input sets can be judiciously designed in the specific case of aircraft path tracking.

7.4 Safe flight envelope as a constraint set

Interestingly, in the context of aircraft path tracking, $\mathcal{SFE}$s may be part of the state variable constraints of the Tube-MPC problem. In the following, we proceed with the computation of the safe flight envelope of the multirotor aircraft under feedback linearization. Considering the full nonlinear model defined previously, the constraint set on the inputs $\mathbf{u} = [a_x,\ a_y,\ a_z,\ a_\psi]^T$ reads:

$$\mathcal{U} = \langle \mathbf{0}_{4\times1}, \mathbf{I}_{4\times4}, \mathbf{I}_{4\times4}, [], [], [] \rangle_{\mathcal{CPZ}} \tag{7.13}$$

This $\mathcal{CPZ}$ corresponds to a 4-dimensional box with unit length in each dimension. We further establish a feasible set of initial conditions which are reachable from any position in time. To this end, a reachability algorithm is first executed using the input sets defined in (7.13) and a limited set of initial conditions. This iterative procedure runs until the following set of initial conditions encompasses a significant fraction of the feasible state space (see Algorithm 1):

$$\mathcal{L}_0(k+1) = \{\mathcal{L}_f(t_f, \mathbf{x}_0, \mathbf{u}(.)) \in \mathbb{R}^n \mid \mathbf{x}_0 \in \mathcal{L}_0(k), \forall t : \mathbf{u}(.) \in \mathcal{U}\} \tag{7.14}$$

Algorithm 1 Safe flight envelope computation

Require: $\mathcal{S}_0, \mathcal{U}, t_f, N$
  $\mathcal{L}_0(1) \leftarrow \mathcal{S}_0$
  for $k = 1 : N$ **do**
    $\mathcal{L}_0(k+1) \leftarrow \mathcal{R}_f([0, t_f], \mathcal{L}_0(k), \mathcal{U})$
  end for
  $\mathcal{R}_F \leftarrow \mathcal{R}_f([0, t_f], \mathcal{L}_0(N+1), \mathcal{U})$
  $\mathcal{R}_B \leftarrow \mathcal{R}_b([0, t_f], \mathcal{L}_0(N+1), \mathcal{U})$
  $\mathcal{SFE} \leftarrow \mathcal{R}_F \cap \mathcal{R}_f$

The resulting safe flight envelope is represented in Figure 7.4. It is worth noting that the defined inputs can also be represented as a $\mathcal{CZ}$. As a result, we perform a comparative analysis of the outcomes from both set representations, highlighting a discernible disparity in the results. This discrepancy stems from the fact that the $\mathcal{CPZ}$ representation, being more

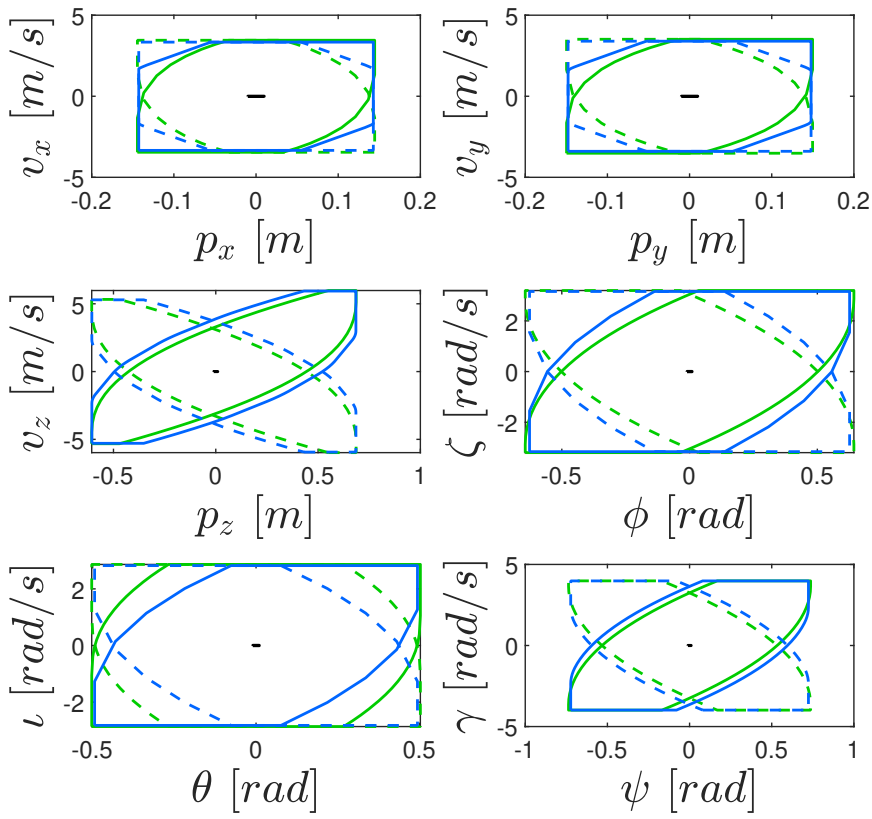


Figure 7.4: 2-D projections of the forward and backward reachable sets using both $\mathcal{CZ}$ and $\mathcal{CPZ}$ representations. Blue lines correspond to $\mathcal{CZ}$, green lines to $\mathcal{CPZ}$. Continuous lines are the forward computation and dotted lines stand for the backward computation.

versatile, requires fewer approximation operations than the $\mathcal{CZ}$ representation.

Furthermore, the $\mathcal{SFE}$ could be intersected with appropriate state constraints $\mathcal{Y} \subset \mathbb{R}^m$ such as regulatory speed limits or obstacles (to avoid collisions):

$$\mathcal{X} = \mathcal{SFE} \cap \mathcal{Y} \tag{7.15}$$

7.4.1 Determination of Ξ and $\mathcal{V}$

Determining both sets, $\Xi \subseteq \mathcal{X}$ and $\mathcal{V} \subseteq \mathcal{U}$, is inherently intricate, particularly due to the necessity of computing robust sets. This task, even when feasible, is quite complex and often necessitates substantial approximations. In addressing this challenge, a straightforward approach proposed by [119] entails imposing relations (7.12a). This simplification streamlines the problem to the determination of λ_1 and λ_2, contingent upon the degree of encountered disturbances. These factors are subsequently incorporated as additional parameters within the MPC framework, enhancing its adaptability and robustness.

7.4.2 Feasibility

By design, the constraint sets are constructed using reachability analysis under the same system dynamics that govern the MPC optimization. This is crucial in guaranteeing that if the initial state lies within the constraint set, the entire nominal trajectory remains feasible over time.

Let $\mathcal{X}_0$ denote the initial state constraint set obtained via reachability analysis. Assuming that the initial condition satisfies $x_0 \in \mathcal{X}_0$, the nominal dynamics used in both the reachability analysis and the optimization ensure that any computed future state x_{k+1} will remain within the appropriate constraint set, provided that $x_k \in \mathcal{X}_k$. In essence, the use of consistent dynamics across both analyses implies that the set $\mathcal{X}_k$ is forward invariant. Thus, if the optimization is feasible at time k, it will remain feasible at time $k+1$, leading to recursive feasibility.

Thus, by ensuring that the initial state is contained within the appropriate tightened constraint set and by using an ancillary control law to handle perturbations, the Tube MPC scheme guarantees that every computed point in the optimization remains feasible.

7.5 Implementation and results

Additionally, the system is influenced by external disturbances represented by vectors $\mathbf{f}_d$ and $\mathbf{m}_d$, expressed as follows:

$$\mathbf{f}_d = \frac{1}{2}\rho C_{Fd} A_F (\mathbf{v}_{wind} - \mathbf{v})^2 \tag{7.16}$$

$$\mathbf{m}_d = \frac{1}{2}\rho C_{Md} A_M \boldsymbol{\omega}_b^2 \tag{7.17}$$

With the square operation being element-wise in those two equations. Modeling these external disturbances involves simplified drag forces and moments [128] under nominal wind conditions described by $\mathbf{v}_{wind}$. These drag forces are associated with drag coefficients, C_{Fd} and C_{Md}.

Additionally, uncertainties on the inertial parameters will be considered further in this study.

The Tube-based methodology is now applied to a Parrot Mambo minidrone whose parameters are reported in Table 3.1.

The results are computed using the MATLAB platform using the Casadi toolbox [129] to solve the optimization problems with an $i5$-$10210U$ processor at $1.60GHz$. To assess the disturbance effects, a Monte-Carlo analysis is achieved and the overall performance is quantified using root mean square error (RMSE) criteria. The mean RMSE is used to assess the average trajectory tracking performance while the variance characterizes the robustness. These criteria read as follows:

$$Mean_{RMSE} = \frac{1}{N_t}\sum_{k=1}^{N_t}\sqrt{\frac{1}{N_{sim}}\sum_{i=1}^{N_{sim}}\frac{1}{4}\sum_{j=1}^{4}\left(\tilde{\boldsymbol{\xi}}_{j,k}(i) - \tilde{\boldsymbol{\xi}}_{j,ref}(i)\right)^2} \tag{7.18}$$

$$Var_{RMSE} = \frac{1}{N_t}\sum_{k=1}^{N_t}\left(\frac{1}{N_{sim}}\sum_{i=1}^{N_{sim}}\frac{1}{4}\sum_{j=1}^{4}\left(\tilde{\boldsymbol{\xi}}_{k}(i) - \tilde{\boldsymbol{\xi}}_{ref}(i)\right)^2\right) - Mean_{RMS}^2 \tag{7.19}$$

The tilde variables indicate the end-user variables for the performance evaluation: $\tilde{\boldsymbol{\xi}} = [p_x, p_y, p_z, \psi]^T$.

UAVs are frequently subject to various disturbances during their operations, with the most notable being the influence of wind, which has the

potential to displace the drone from its designated position. Other disruptive factors include suboptimal parameterization and inherent noise in the system inputs and outputs.

Table 7.1: Parameters of the disturbances

Parameter (symbol)	**Value (units)**
Norm of the wind speed ($\|\|\boldsymbol{\nu}_{wind}\|\|$)	$[0, 0.5]$ m/s
Translational drag coefficient (C_{Fd})	0.1
Moment drag coefficient (C_{Md})	0.001
Inertia uncertainties	20%

Referring to equations (7.16) and the values in Table 7.1, the drone is exposed to forces within the nominal range of $[0, 0.0323]$ N. During horizontal motion, the minidrone tilts at a modest angle of approximately 0.3491rad (20 degrees), generating forces of around $0.2114\,N$. Consequently, disturbances in force amount to approximately 15%. Additionally, disturbances on moments and an additional 20% uncertainty introduced in the inertia values are considered.

A standard MPC is compared to the tube-MPC strategy. For the sake of fairness, identical disturbance conditions and parameters are applied. The trajectories of the standard MPC, shown in Fig. 7.5, predominantly remain close to the desired path, with a few instances of larger deviations due to the effect of the disturbances. A clear contrast can be observed with the results reported in Fig. 7.6. In the Tube-MPC scenario, a tight corridor encompasses the distributed trajectories, reducing the impact of model disturbances. The error quantification in Table 7.2 supports this visual interpretation. Additionally, we illustrate the influence of the parameters λ_1 and λ_2, setting the constraint sets, on the control performance. A discernible trend emerges, revealing that a lower λ_1 value ensures robust and high-performance behavior, while λ_2 appears to have a less pronounced impact. However, it is imperative to remember that λ_1 serves as a critical link between the nominal set Ξ and the constraint set $\mathcal{X}$. If λ_1 is set too low, it drastically reduces the size of Ξ, potentially resulting in conservative constraint enforcement. Therefore, a delicate balance must be found, preventing constraint violation while avoiding conservatism. Notice that the magnitude of the disturbances has been selected such that the standard

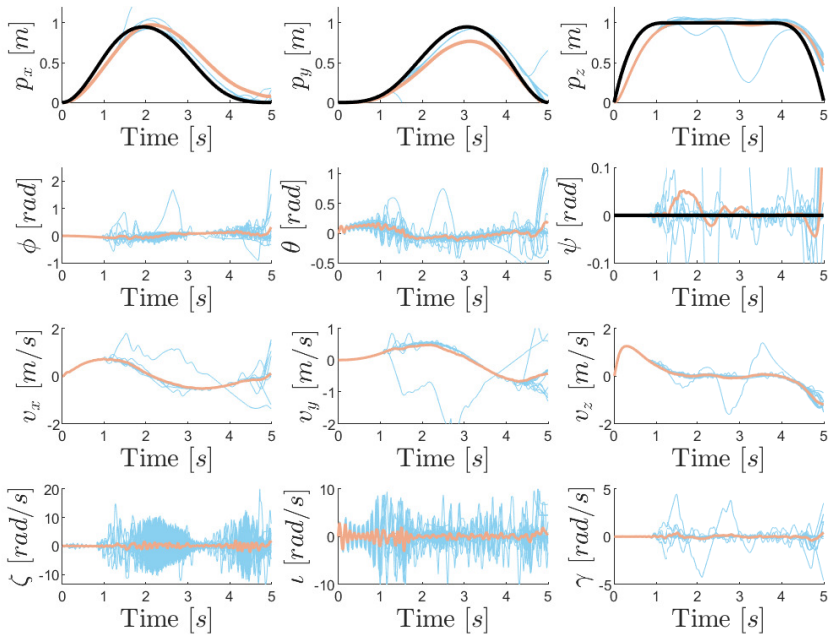


Figure 7.5: Monte-Carlo simulations of the methodology (MPC - $\mathcal{CPZ}$ - INDI) under disturbances. Blue lines are the Monte-Carlo simulations, black lines show the references and orange lines show the mean trajectory

MPC keeps the system stable in all the Monte Carlo simulations. However, Tube MPC can handle more severe disturbances.

The application of tube-based model predictive control combined with incremental nonlinear dynamic inversion to achieve path tracking of a multirotor aircraft with rotary wings in a polynomial zonotopic framework yields promising results. In particular, this study demonstrates the potential of this control strategy for effectively managing disturbances and robust operations of multirotor aircraft.

The control strategy relies on a multi-layered control framework that employs robust feedback linearization in the inner loop. The outer loop then exploits Tube-MPC, which itself builds upon two controllers, a nominal controller and an ancillary controller. Constrained polynomial zonotopes are conveniently used to represent state and input constraints.

One of the noteworthy contributions of this chapter is the introduc-

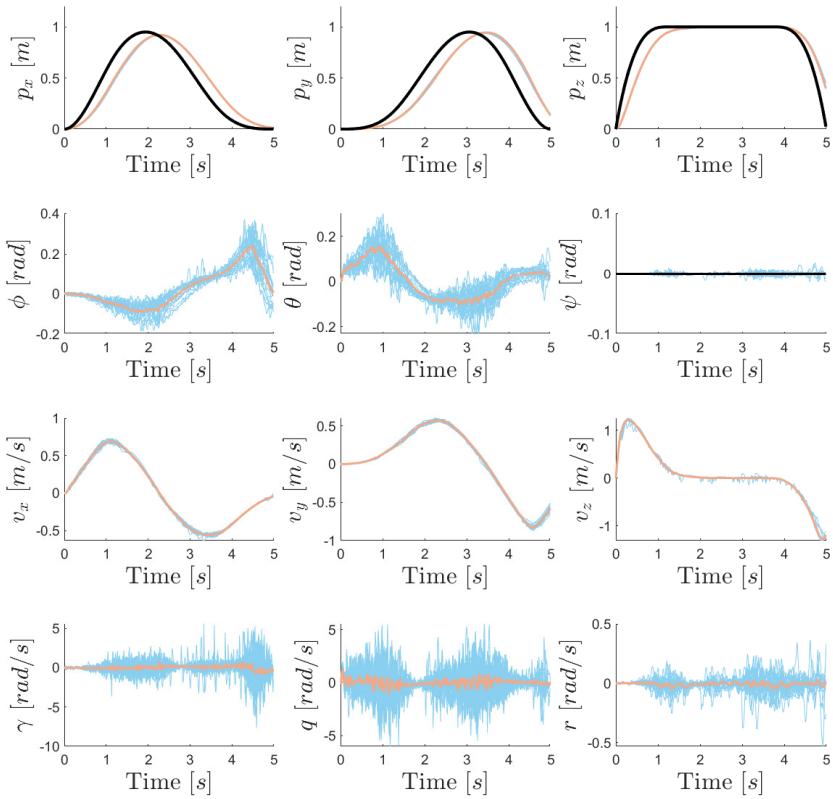


Figure 7.6: Monte-Carlo simulations of the methodology (Tube-MPC - $\mathcal{CPZ}$ - INDI) under disturbances such as inertia, wind and internal approximations. Blue lines are the Monte-Carlo simulations, black lines show the references and orange lines show the mean trajectory

Table 7.2: Performances of the methodology when varying the parameters λ_1 and λ_2.

Method	$Mean_{RMS}$	Var_{RMS}	λ_1	λ_2
MPC	1.6×10^{-1}	7.3×10^{-2}	/	/
Tube MPC	1.2×10^{-1}	6.7×10^{-6}	0.1	0.1
	1.3×10^{-1}	2.6×10^{-5}	0.1	0.5
	1.3×10^{-1}	3.7×10^{-6}	0.1	0.9
	1.2×10^{-1}	1.2×10^{-4}	0.5	0.1
	1.3×10^{-1}	2.1×10^{-3}	0.5	0.5
	1.2×10^{-1}	2.1×10^{-4}	0.5	0.9
	4.8×10^{-1}	4.8×10^{-1}	0.9	0.1
	5.2×10^{-1}	3.1×10^{-1}	0.9	0.5
	5.2×10^{-1}	2.4×10^{-1}	0.9	0.9

tion of constrained polynomial zonotopes in the form of safe flight envelopes, also standing as state and input constraints in the Tube-MPC formulation. This representation offers a significant advantage, enabling the application of standard operations based on set theory. Moreover, the research demonstrates that non-convex representations improve accuracy and broaden the applicability of control strategies. This is particularly valuable for critical applications, where the computation of a safe flight envelope is essential for secure aircraft maneuvering. Additionally, our study reveals that, with carefully chosen parameters, Tube MPC outperforms standard constraint-tightened MPC.

7.6 Experimental Validation

In this section, we describe the experimental validation of our Tube-MPC approach. In simulation, we demonstrated the full potential of our control scheme by implementing a Tube-MPC architecture that employs two MPC layers together with polynomial zonotopes. However, when transi-

tioning to an experimental setup on an embedded UAV platform, practical constraints—most notably limited computational power—necessitate a simplified implementation.

To address these constraints, the experimental validation relies on the use of zonotopes and constrained zonotopes, along with a linear representation of the closed-loop system dynamics. Moreover, we replace the ancillary MPC controller with a state-feedback law to further reduce the computational load.

Additionally, the inner loop of the Tube-MPC does not include position tracking, as the position is measured solely by GPS sensors. Due to the inherent inaccuracy of GPS for exact positioning, this modification allows us to focus on the control of other critical states while ensuring robust and computationally efficient performance.

Overall, this experimental setup illustrates how the advanced control concepts developed in simulation can be adapted to meet the practical limitations of real-time, embedded UAV control.

7.6.1 System Overview

The experimental setup consists of a hexacopter built on a DJI F550 frame with six DC brushless motors, as shown in Figure 2.3. The primary flight controller is a Pixhawk 4, equipped with an IMU and a GPS, which is connected to a Raspberry Pi 5 via the RX interface. The Raspberry Pi is responsible for gathering data from additional sensors, including a LiDAR and a stereo camera. The LiDAR provides accurate distance measurements for obstacle detection, while the stereo camera is used for depth perception and visual data, aiding in more precise navigation. The Raspberry Pi processes sensor data and computes the control strategy in real-time, leveraging the Tube-MPC approach. It then transmits command messages to the Pixhawk 4 using the MAVLink protocol. The goal is to achieve fully autonomous control, where the external computer only needs to send an initial goal, after which the UAV navigates independently. However, during the experiments, a master radio command is kept in the loop in case of malfunctions or unexpected behavior. This allows for manual intervention if necessary, providing an additional layer of security during real-world testing.

The system operates with a sampling time of 0.2 seconds, and the real-time data processing on the Raspberry Pi allows for rapid adjustments to

the UAV's trajectory. The UAV is powered by a 4-cell Li-ion battery (14.8v). Table 7.3 gives more information about the setup.

Table 7.3: UAV Specifications

Parameter	**Value**
Type	Hexarotor
Weight	3.37 kg
Height	37.5 cm
Wingspan	55 cm
Rotor radius	2.2 x 1.2 cm
KV	1000

7.6.2 System Identification

The system we aim to control is the closed-loop setup, where the Pixhawk 4 serves as the base controller to stabilize the UAV. Our goal is to enhance the robustness of this setup using Tube-MPC. Real experimental data was gathered by applying multiple-step inputs to the system. The system has four control inputs: the desired speeds along the three axes of the NED frame (which we associate with the xyz axes) and the yaw angle, measured relative to the z-axis (down axis). The outputs are the corresponding speeds and yaw angle. Figure 7.7 shows a sample of the data gathered during the step response experiments.

At first glance, linear dynamics predominantly drive the system's behavior. While noise and wind disturbances affect the output responses, the system follows the step inputs in a first-order fashion for the three speeds. However, the yaw angle exhibits slight oscillations. These observations are essential, as they will guide our control design, ensuring that the methods can run efficiently on embedded hardware at high frequencies.

The gathered data was split into training and testing sets for the identification process. We used MATLAB's System Identification Toolbox and applied subspace methods to identify a linear model with four states. During the identification, we ensured that the outputs remained independent, as intended by the base controller. Figure 7.8 shows the validation of the identified model compared to the experimental data.

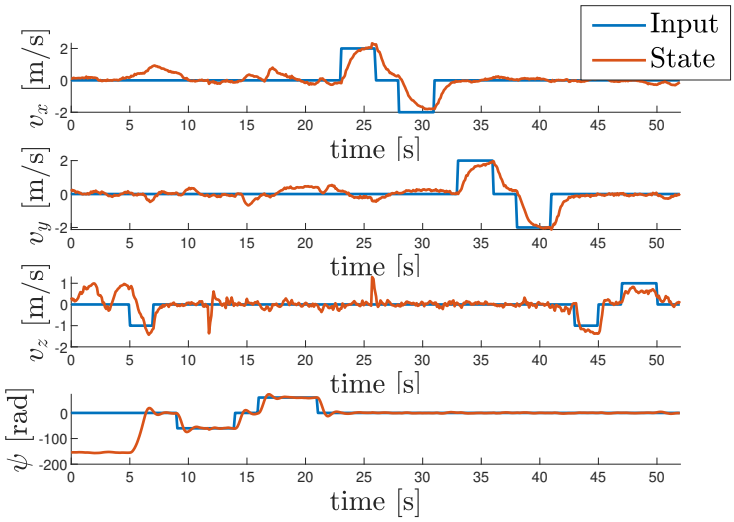


Figure 7.7: Data for identification : UAV steps responses for four states : velocities along x, y and z axis and the yaw angle.

Overall, the system's dynamics are well represented. Some overshoots are observed in the speed responses, and oscillations in the yaw response were not fully captured. However, the control methods we will employ, particularly Tube-MPC, are designed to handle disturbances inherently. Therefore, we can consider these slight discrepancies between the model and real-world experiments as additional disturbances that the control strategy will compensate for.

7.6.3 Implementation of Tube MPC

In the practical implementation, the Tube MPC is designed with a linear MPC for the nominal controller, using an augmented model obtained from the system identification process. The model is augmented by incorporating the dynamics of the position states x, y, and z, achieved by adding three integrators to the system. This augmentation simplifies reference tracking for position inputs, allowing the user to directly set references for x_{ref}, y_{ref}, z_{ref}, and yaw_{ref}. The nominal control is expressed as:

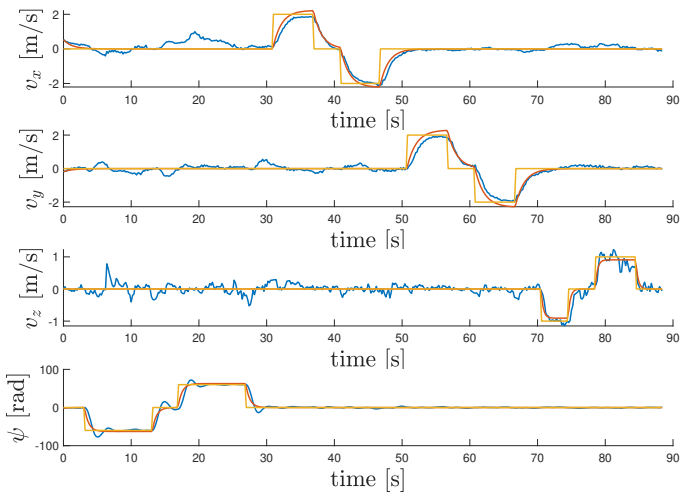


Figure 7.8: Validation - Model vs Data - Yellow lines are the input data provided to the system, blue lines are the validation data used to measure the identified system performances shown in orange lines.

$$
\begin{aligned}
\underset{\boldsymbol{\nu}}{\text{minimize}} \quad & \sum_{k=1}^{N-1} (\boldsymbol{\xi}(k) - \boldsymbol{\xi}_{ref}(k))^T \mathbf{Q} (\boldsymbol{\xi}(k) - \boldsymbol{\xi}_{ref}(k)) \\
& \qquad + (\boldsymbol{\nu}(k) - \boldsymbol{\nu}_{ref}(k))^T \mathbf{R} (\boldsymbol{\nu}(k) - \boldsymbol{\nu}_{ref}(k)) \\
& \qquad + (\boldsymbol{\xi}(N) - \boldsymbol{\xi}_{ref}(N))^T \mathbf{F} (\boldsymbol{\xi}(N) - \boldsymbol{\xi}_{ref}(N)) \qquad (7.20) \\
\text{subject to} \quad & \boldsymbol{\xi}(k+1) = \mathbf{A}_{T,I} \boldsymbol{\xi}(k) + \mathbf{B}_I \boldsymbol{\nu}(k), \\
& \boldsymbol{\xi}(k) \in \Xi, \;\; \boldsymbol{\nu}(k) \in \mathcal{V},
\end{aligned}
$$

where $\boldsymbol{\xi}_{ref}$ and $\boldsymbol{\nu}_{ref}$ are the state and input reference trajectories, respectively. The matrices $\mathbf{A}_{T,I}$ and $\mathbf{B}_I$ represent the linear dynamics of the augmented model. $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{F}$ are weight matrices that adjust the closed-loop behavior, while Ξ and $\mathcal{V}$ are constraint sets that ensure feasibility. N denotes the receding horizon. This study utilizes the do-mpc library [130] in conjunction with the IPOPT solver to address the MPC problem. We expect the computational time to be less than half of the sampling time, given that this will constitute the primary computational burden.

The ancillary controller, responsible for ensuring the real system re-

mains close to the nominal trajectory, is implemented as a state feedback controller. This controller operates based on the identified dynamics without the augmented integrators because the position estimates from the real system are subject to inaccuracies due to the reliance on GPS data. The ancillary controller receives the optimal nominal inputs from the linear MPC and computes the actual control inputs for the system.

The state feedback controller calculates the real control inputs using the following law:

$$\mathbf{u} = \boldsymbol{\nu}^* + \mathbf{K}\left(\mathbf{x} - \boldsymbol{\xi}^*\right) \tag{7.21}$$

where $\boldsymbol{\nu}^*$ represents the optimal inputs from the nominal controller, and $\boldsymbol{\xi}^*$ represents the optimal velocities and yaw angle. The gain matrix $\mathbf{K}$ is designed using the Linear Quadratic Regulator (LQR) method.

This architecture allows the nominal controller to focus on trajectory optimization, while the ancillary controller ensures the real system remains within the desired "tube" around the nominal trajectory. A linear MPC simplifies the optimization problem, while the state feedback controller compensates for the actual system dynamics and disturbances in real time.

7.6.4 Constraint Sets

In our Tube MPC formulation, constraints play a crucial role in the control strategy. These constraints are applied to the nominal model and account for the disturbances that affect the real system, ensuring that the nominal trajectory remains within a defined region. Specifically, we impose state and input constraints on the nominal system:

$$\boldsymbol{\xi}(k) \in \Xi, \quad \boldsymbol{\nu}(k) \in \mathcal{V} \tag{7.22}$$

where $\Xi \subseteq \mathbb{R}^n$ is the set of allowable states, and $\mathcal{V} \subseteq \mathbb{R}^r$ represents the set of allowable control inputs. These constraints ensure that the system operates within physical and performance limits, such as actuator saturation.

To account for disturbances, the constraints are tightened by incorporating a robust invariant set $\mathcal{L}$ around the nominal trajectory. We could rely on the same approach as in the previous section; however, since the system is completely linear and the ancillary controller is implemented as

a state-feedback, the constraint tightening can be determined by another formula. Mathematically, we define the tightened constraints as:

$$\Xi = \mathcal{X} \ominus \mathcal{L}, \quad \mathcal{V} = \mathcal{U} \ominus \mathbf{K}\mathcal{L} \tag{7.23}$$

where $\ominus$ denotes Minkowski subtraction, and $\mathbf{K}$ is the state feedback gain. The set $\mathcal{L}$ represents the bounds on the error between the actual and nominal trajectories, ensuring that the real system remains within the desired bounds $\mathcal{X} \subseteq \mathbb{R}^n$ despite disturbances, as well for the control input $\mathcal{U} \subseteq \mathbb{R}^r$ and it is computed as shown in Section 7.4.

7.7 Results

The results of the method were validated through a series of experiments. For this purpose, reference steps were applied to the four outputs of the system: x_{ref}, y_{ref}, z_{ref}, and yaw_{ref}. Each step change was introduced sequentially to test the controller's ability to track setpoint changes in real time, while compensating for external disturbances.

Figure 7.9 illustrates a MATLAB simulation of the system under the Tube-MPC framework, showing the expected output of the system without disturbances.

To assess consistency and robustness, the actual experiment was repeated four times under the same reference steps as in the simulations. The tests were performed outdoors, where the UAV was subject to environmental disturbances like wind.

Figure 7.10 illustrates the UAV's speed along the x-axis. Despite external disturbances, the UAV consistently followed the path induced by the nominal controller, demonstrating the efficacy of the ancillary feedback controller in compensating for deviations from the nominal trajectory. The Root Mean Squared Error (RMSE) between the nominal trajectories and the system states is computed in Table 7.4. The consistency across the runs highlights the reliability of the proposed control scheme for UAV applications in dynamic environments. A larger error is observed in the yaw angle, as discussed in the identification section.

Figure 7.11 shows the evolution of the nominal states, the states, the nominal inputs, and inputs in the first experiment, with the same references as in simulations. The nominal speed generated by the MPC layer is followed tightly by the ancillary, which aims to drive the system toward

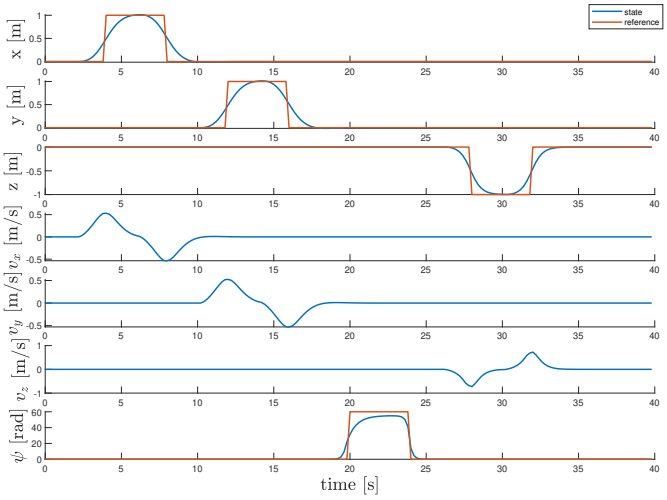


Figure 7.9: Simulation of the system without disturbances using the Tube-MPC framework (MATLAB). States are shown in blue, and references are in orange.

Table 7.4: RMSE values for different experiments

RMSE	v_x	v_y	v_z	ψ
Exp 1	0.1267	0.1549	0.1078	7.5205
Exp 2	0.1583	0.1317	0.0970	6.0900
Exp 3	0.1334	0.1431	0.0991	6.3427
Exp 4	0.1094	0.1307	0.1106	6.3947

the nominal behavior. The main difference appearing is the magnitude of the states, which may come from identification issues, as well as the yaw dynamics, which has been simplified to a one-order system. However, the control can handle disturbances, both wind and dynamic uncertainties.

7.7.1 Experimental Validation

One of the key objectives of this work is to develop a control strategy that is not only robust but also computationally efficient for real-time application on embedded systems. To achieve this, the overall control structure was designed to distribute the computational load across different com-

ponents, leveraging their respective strengths. The control loop consists of a fast low-level controller (in milliseconds), a state feedback computation (also in milliseconds), and alinear MPC with linear constraints. This MPC is specifically tuned to be computationally light, typically completing in less than 100 ms.

The average computation time for the entire control loop on the Raspberry Pi 5 is 0.052 seconds, with an upper limit observed of 0.077 seconds for a horizon of 10 steps. This respects the imposed sampling time and leaves additional room for other computations, such as path planning or high-level decision-making. These results demonstrate that the proposed controller architecture successfully achieves its goal of reducing the computational burden, making it suitable for deployment on resource-constrained platforms.

7.8 Conclusion

This work introduces a robust control approach for UAVs under disturbances by combining Tube-MPC and a zonotopic framework with reachability analysis. The integration of zonotopes to compute robust invariant sets ensures that the actual UAV trajectory remains close to the nominal trajectory, despite disturbances. The real-time implementation of the control strategy was demonstrated on a UAV platform using the Pixhawk 4 and Raspberry Pi 5, validating the effectiveness of the method in an outdoor environment subject to external disturbances. The experiments show that the system can achieve robust trajectory tracking, highlighting the strength of the proposed approach in maintaining robust operation and effective disturbance rejection.

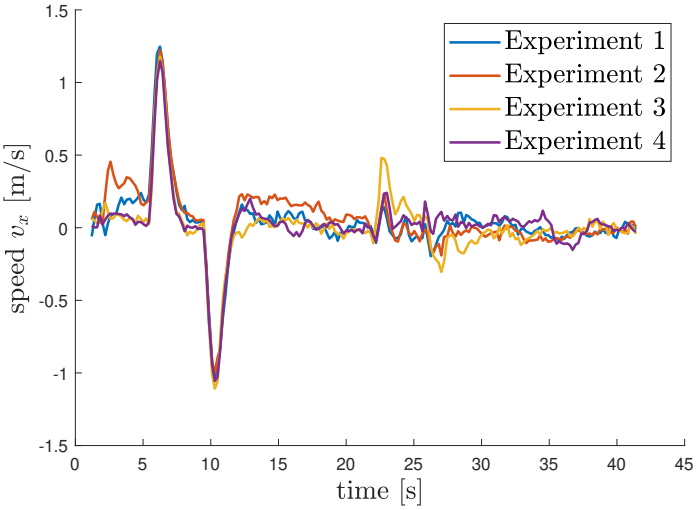


Figure 7.10: System speed along the x-axis during the four different experiments.

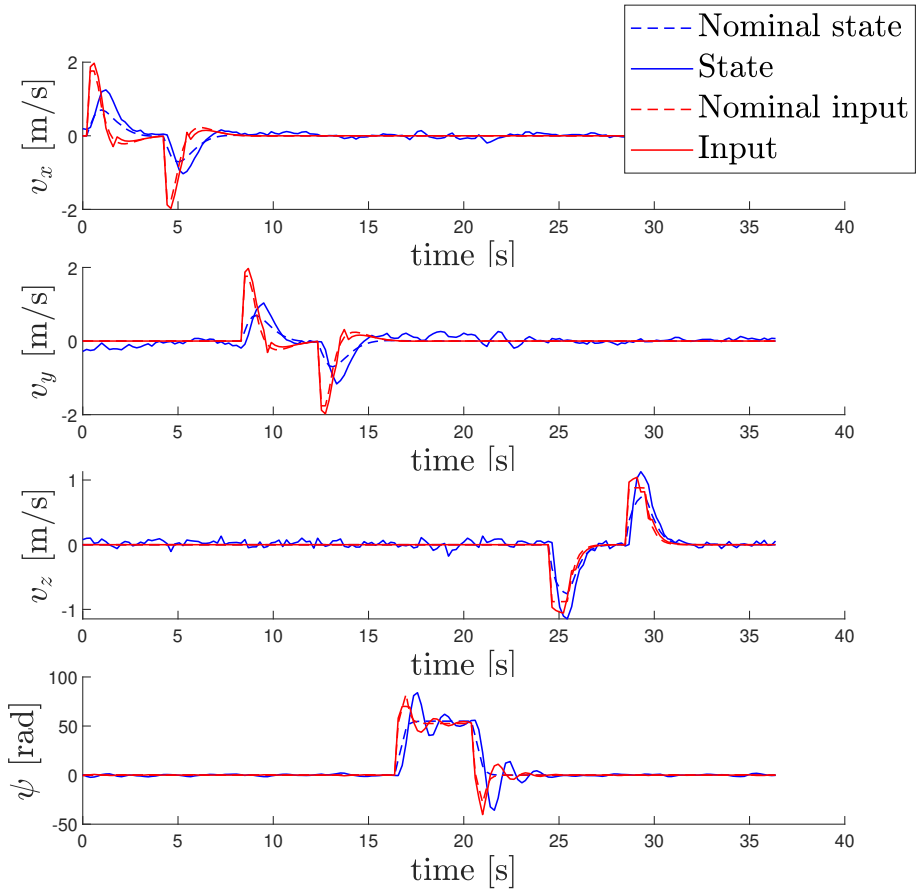


Figure 7.11: Data of the first experiment: nominal states are shown as dotted blue lines, while the actual states are shown as solid blue lines. Nominal inputs are represented by dotted red lines, and the actual inputs are shown as solid red lines.

Chapter 8

Real-Time Path Planning Using Zonotopic Extensions to Rapidly-Exploring Random Trees

8.1 Introduction

While exploring UAV control, several fundamental questions remain unanswered: What about the environment in which the UAV operates? How should we generate reference trajectories for the control scheme? And crucially, how can we ensure that the planned trajectory accounts for obstacles and environmental constraints?

Collision avoidance and path planning are critical capabilities for autonomous robots operating in dynamic and unstructured environments. Efficient navigation while avoiding both static and dynamic obstacles is essential for applications ranging from service robotics to autonomous vehicles and industrial automation. Over the years, numerous approaches have been developed for collision avoidance, each with its own strengths and limitations.

Traditional methods include heuristic-based approaches, such as rule-based systems and navigation functions [131, 132], and potential field methods [133]. While computationally efficient, these methods often suffer from limitations such as local minima issues and may fail in complex en-

vironments.

Sampling-based approaches, including probabilistic roadmaps (PRM) and rapidly exploring random trees (RRT), have revolutionized motion planning by enabling robots to explore high-dimensional spaces efficiently [134]. These methods are particularly well-suited for scenarios where a complete map is unavailable or computational constraints prohibit exhaustive searches.

RRTs, introduced by LaValle [134], incrementally build a tree rooted at the start position, expanding towards random samples to explore the space. Their simplicity and scalability have led to widespread adoption in robotics. The RRT* variant [135], an extension of the original algorithm, guarantees asymptotic optimality by rewiring the tree to optimize paths during the sampling process. Beyond RRT*, several variants have been proposed to enhance performance, including informed RRT* [136], which biases sampling towards the goal to accelerate convergence, or batch processing methods [137], which improve computational efficiency.

However, conventional RRT-based methods face significant challenges, particularly in efficiently exploring high-dimensional state spaces and adapting to dynamic environments. Traditional RRT and its variants may suffer from suboptimal exploration due to random sampling, leading to increased computational burden and potential convergence issues, especially when navigating non-convex and cluttered spaces. Furthermore, a common limitation among these approaches is their reliance on primitive obstacle representations, such as bounding boxes or spheres, which may not accurately capture the true geometry of obstacles in complex environments.

In this chapter, we address these challenges by leveraging zonotopes for path planning and obstacle representation which provide a powerful mathematical construct for representing regions in motion planning. Defined as affine transformations of hypercubes, zonotopes offer computationally efficient operations for collision checking, set containment, and uncertainty representation. A major benefit of zonotopes is their low computational complexity as dimensions increase [138], making them particularly powerful for high-dimensional spaces where other shape representations can become prohibitively complex. While their use in robotics has been explored in the context of reachability analysis [139] and control synthesis, their specific potential for sampling-based motion planning, particularly within RRT algorithms, remains underexplored. By leveraging the algebraic properties of zonotopes, not only individual obstacles but also re-

gions of the state space can be precisely represented, enabling novel sampling strategies that operate on regions instead of discrete points. Such region-based sampling allows for more efficient exploration and facilitates better integration with collision-checking procedures.

This chapter introduces a novel application of zonotopes in real-time RRT* algorithms, addressing key limitations in existing methods. Specifically, we propose a zonotopic representation of obstacles, enabling accurate modeling of complex geometries and dynamic scenarios. Furthermore, we replace traditional point-based sampling with zonotopic region sampling, leveraging the associated algebra for efficient collision checking and path refinement. By integrating these innovations, our method aims to achieve robust and computationally efficient motion planning for UAVs in dynamic and cluttered environments.

8.2 Rapidly-Exploring Random Tree Algorithm

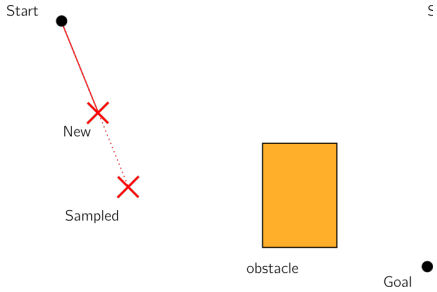
The Rapidly-Exploring Random Tree (RRT) algorithm, first introduced by LaValle [134], is a sampling-based method for solving motion planning problems. RRT incrementally builds a tree in the configuration space by randomly sampling points and connecting them to the nearest node in the tree, prioritizing rapid exploration of the space. While RRT is highly effective for finding feasible paths, it does not guarantee optimality, Fig 8.1.

RRT* [135], introduced by Karaman and Frazzoli, is an extension of RRT that ensures asymptotic optimality, meaning that as the number of samples increases, the solution converges to the optimal path. RRT* modifies the original algorithm by incorporating two key steps: *rewiring* and *local optimization*, enabling the tree to improve its structure as new samples are added.

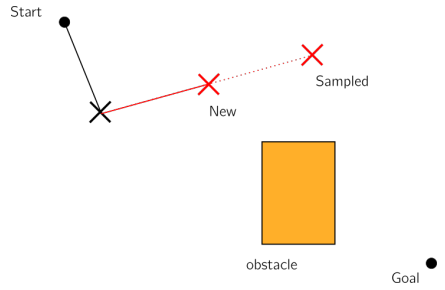
8.2.1 Algorithm Description

The RRT* algorithm operates iteratively by sampling points in the configuration space, connecting them to the tree, and optimizing the tree's structure to reduce the cost of the solution. Its main steps are as follows:

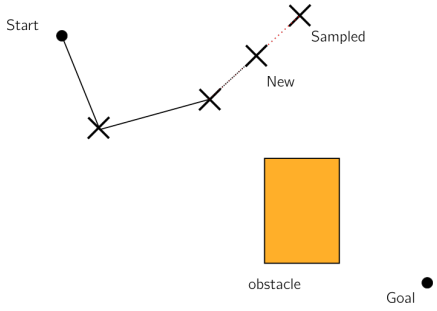
1. **Sampling:** A random point is sampled in the configuration space.



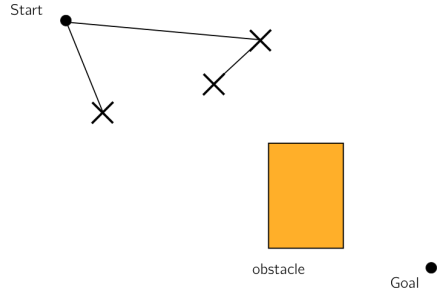
(a) First step of the RRT algorithm



(b) Once a new node is added, the algorithm start searching for a new one



(c) The process continue until a node is found



(d) At some point a rewiring operation is applied to redefine the tree with existing node

Figure 8.1: Illustration of the RRT* algorithm. At each iteration, the algorithm samples a random point in the space and finds the closest node in the tree. A collision check is performed between the new connection (dotted line) and the obstacles. If a collision is detected, the sample is discarded, and a new one is generated. Otherwise, a new point is created using a step-size parameter and added to the tree. Finally, as shown in the rightmost diagram, the tree can be rewired if a shorter path exists, improving overall path efficiency.

2. **Nearest Neighbor:** The nearest node in the tree to the sampled point is identified based on a distance metric.

3. **Steering:** A new node is created by moving from the nearest node toward the sampled point, typically within a predefined step size.

4. **Collision Checking:** The connection between the nearest node and the new node is checked for collision with obstacles. If a collision occurs, the algorithm returns to the sampling phase.

5. **Rewiring:** The algorithm considers all nodes within a given radius of the new node and rewires the tree if connecting through the new node reduces the cost to reach those nodes.

The combination of these steps ensures both exploration and exploitation of the configuration space, balancing feasibility and optimality.

8.2.2 Variants of RRT*

Several variants of the RRT* algorithm have been proposed to address its computational challenges and improve its performance:

Informed RRT* is a variant of the RRT* algorithm that enhances its efficiency by focusing the sampling process on a subset of the search space where the optimal path is more likely to be found. Introduced by Gammell et al. [136], this method significantly improves convergence towards the optimal solution, especially in high-dimensional and constrained environments.

The key insight behind Informed RRT* is the use of an *informed sampling space* that excludes regions that cannot improve the current solution. Once an initial feasible path is found, Informed RRT* defines an ellipsoidal sampling space centered on the start and goal locations, where the cost of any new path is guaranteed to be better than the current solution.

Batch RRT* was proposed by Arslan and Tsiotras [137] and introduces a batch-processing approach to reduce the computational overhead of iterative rewiring in RRT*. Instead of processing samples one by one, this variant collects multiple samples in a batch and performs rewiring collectively. This allows the algorithm to globally optimize the tree structure over multiple samples, rather than incrementally. The batch-based approach also facilitates parallel processing, which can significantly speed

up computation. Batch RRT* is particularly advantageous in scenarios where computational resources are constrained, and frequent tree updates may not be feasible. By using a relaxed dynamic programming formulation, Batch RRT* ensures that paths converge to the optimal solution more efficiently compared to the standard RRT*.

Real-Time RRT* (RT-RRT*) was introduced to address the need for real-time motion planning in dynamic and partially known environments, such as those encountered in robotics and autonomous vehicles [140]. RT-RRT* modifies the RRT* framework to operate within strict time constraints by maintaining a tree that adapts dynamically as new information about the environment becomes available. Instead of aiming for global optimality, RT-RRT* focuses on providing feasible, suboptimal solutions within a limited time window, ensuring that the robot can continuously operate without delays. The algorithm prioritizes computational efficiency by limiting the extent of rewiring and tree exploration in each iteration. A notable feature of RT-RRT* is its use of a replanning mechanism, where the algorithm updates the tree whenever new obstacles or constraints are detected. This makes it particularly suitable for real-time applications where the environment changes rapidly. While it sacrifices some degree of optimality compared to standard RRT*, it ensures consistent performance under time-critical conditions, making it a practical choice for real-world deployments.

8.2.3 Challenges in RRT* Methods

Despite its strengths, RRT* has limitations, particularly in handling complex environments with obstacles. Most implementations lack a mathematical representation for obstacles, relying on computationally expensive collision-checking procedures or inaccurate representations. This work addresses this limitation by leveraging zonotopes and polynomial zonotopes to represent obstacles, enabling efficient and algebraic collision checking.

8.3 Zonotopes in Random sampling methods

8.3.1 Handling obstacles

Bounding box as zonotopes

Bounding boxes are often used to describe the minimal enclosing box around an object in a given coordinate system. A bounding box is typically defined by two points: the minimum corner $(x_{\min}, y_{\min}, z_{\min})$ and the maximum corner $(x_{\max}, y_{\max}, z_{\max})$, which represent the extents of the box along each axis.

To convert a 3D bounding box into a zonotope, we first need to compute the center and extents of the bounding box. The center c of the bounding box is the midpoint of the three corners:

$$\mathbf{c} = \left(\frac{x_{\min} + x_{\max}}{2}, \frac{y_{\min} + y_{\max}}{2}, \frac{z_{\min} + z_{\max}}{2} \right), \tag{8.1}$$

and the extents along the x, y, and z axes are:

$$\ell_x = \frac{x_{\max} - x_{\min}}{2}, \quad \ell_y = \frac{y_{\max} - y_{\min}}{2}, \quad \ell_z = \frac{z_{\max} - z_{\min}}{2}. \tag{8.2}$$

The generator matrix G for the zonotope representation of the bounding box is formed by the extents along each axis. This gives the following generator matrix for a 3D bounding box:

$$\mathbf{G} = \begin{bmatrix} \ell_x & 0 & 0 \\ 0 & \ell_y & 0 \\ 0 & 0 & \ell_z \end{bmatrix}, \tag{8.3}$$

Ellipsoids to Zonotopes

An ellipsoid is a common geometric representation that can be converted into a zonotope. It is mathematically described by a quadratic form:

$$\mathcal{E} = \left\{ \mathbf{x} \in \mathbb{R}^n \mid (\mathbf{x} - \mathbf{c})^T \mathbf{P} (\mathbf{x} - \mathbf{c}) \leq 1 \right\}, \tag{8.4}$$

where $\mathbf{c} \in \mathbb{R}^n$ is the center, and $\mathbf{P} \in \mathbb{R}^{n \times n}$ is a positive definite matrix defining the shape and orientation. [141] proposes an efficient over-approximation method based on a desired number of generators. A key

consideration is the trade-off between zonotopes and traditional models like ellipsoids, particularly regarding modeling precision. Unlike ellipsoids, which are restricted to representing elliptical shapes, zonotopes—being convex polytopes—can approximate any convex object with arbitrary precision. Increasing the number of generators can reduce the approximation error to a small ϵ, making zonotopes highly adaptable to a wide range of convex shapes encountered in real-world environments, where objects are rarely perfectly ellipsoidal. For instance, a complex convex obstacle can be closely matched by a zonotope with sufficient generators. In contrast, an ellipsoid might overestimate or poorly fit the same shape, leading to larger approximation errors. However, both zonotopes and ellipsoids introduce errors for non-convex objects due to their convex nature. To address this, non-convex objects can be decomposed into multiple overlapping convex zonotopes, offering a piecewise approximation that is more precise than a single ellipsoid, albeit at the cost of increased collision-checking computations. Alternatively, polynomial zonotopes could represent non-convex shapes, but this approach incurs significant computational overhead due to the nonlinear optimization required for collision detection.

Vertex Sets to Zonotopes

Another frequent scenario involves converting a set of vertices into a zonotope. Given a set of points $\{v_1, v_2, \dots, v_k\} \subset \mathbb{R}^n$, the convex hull of these points can be enclosed by a zonotope. Tools such as the CORA toolbox [100] provide functions to perform this operation efficiently. The key steps involve determining the center as the mean of the vertices and constructing the generators based on the relative positions of the vertices to the center. This representation allows leveraging zonotope-based operations while preserving the geometric structure of the original vertex set.

8.3.2 Sampling Zonotopes

In random sampling-based methods, such as those used in Rapidly exploring Random Tree (RRT) algorithms, the initial step involves sampling the configuration space. Traditionally, this is achieved by defining the boundaries of the workspace and uniformly sampling individual points within this defined space. While effective, this method often fails to fully

exploit the spatial information of the environment, especially in higher-dimensional spaces. In some advanced approaches, sampling is biased by using additional metrics or incorporating prior knowledge about the workspace to guide exploration more effectively.

The use of zonotopes for sampling provides an alternative paradigm. Instead of sampling discrete points, zonotope sampling involves selecting entire regions of the configuration space in one step. This approach can be particularly beneficial in dynamic or uncertain environments, where representing regions rather than single points offers improved robustness and computational efficiency.

For example, in a 2D environment, a sampled zonotope would include a center, a defined number of generators, and associated generator magnitudes. Similarly, in 3D spaces, a sampled zonotope would represent a volumetric region that could adapt to the environment's complexity. The key steps for sampling zonotopes involve:

1. Center Sampling: The center of the zonotope is sampled using conventional methods, such as uniform random sampling, guided sampling, or biasing techniques.

2. Generator Configuration: The generators of the zonotope, which define its shape and extent, are then sampled. This includes selecting the number of generators, their orientation, and their magnitude.

To ensure computational feasibility, constraints are often imposed on the sampling process. For instance, the number of generators can be fixed to match the dimensionality of the space. In a 3D workspace, a straightforward approach is to construct cubic-like zonotopes with diagonalized generator matrices, such as:

$$\mathbf{G} = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & g_3 \end{bmatrix}, \tag{8.5}$$

where g_1, g_2, and g_3 are tunable parameters that determine the zonotope's size along each axis. These parameters can be chosen randomly or based on predefined exploration strategies.

The advantage of zonotope sampling lies in its ability to explore larger regions of the configuration space in fewer iterations. This can lead to improved efficiency in tree expansion for RRT* algorithms, especially when

dealing with high-dimensional or complex environments. However, selecting the appropriate zonotope parameters is critical. Excessively large zonotopes may lead to poor resolution in sampling, while overly small zonotopes may revert the method to point-based sampling.

In practice, the choice of zonotope shapes and sizes can depend on the specific application and the workspace's geometric constraints. For instance, in structured environments, elongated or directional zonotopes may be more effective, whereas isotropic shapes like cubes may suffice for unstructured, open spaces. Further research and experimentation are needed to establish optimal zonotope sampling strategies for various scenarios.

8.3.3 The Nearest Neighbour

Selecting the nearest neighbor is a fundamental step in the RRT algorithm, as it determines the node to which the sampled point (zonotope in our case) will be connected. This requires defining a suitable metric for evaluating proximity.

Metrics for Distance Calculation

In classical approaches, the metric typically employed is the Euclidean distance between the sampled point and the nodes of the tree. This is straightforward and computationally efficient for point representations. However, when working with sets such as zonotopes, the problem becomes more complex.

For sets, the ideal metric would involve calculating the smallest distance between two sets, defined as:

$$d(\mathcal{Z}_1, \mathcal{Z}_2) = \min\{\|\mathbf{z}_1 - \mathbf{z}_2\| \mid \mathbf{z}_1 \in \mathcal{Z}_1, \mathbf{z}_2 \in \mathcal{Z}_2\}. \tag{8.6}$$

While this approach is theoretically optimal, finding the minimum distance between two zonotopes is computationally intensive. It involves solving a constrained optimization problem, which is not suitable for real-time applications like RRT.

Approximating the Nearest Neighbour

To simplify the computation while maintaining practicality, we approximate the distance by comparing the centers of the zonotopes:

$$d_{\text{approx}}(\mathcal{Z}_1, \mathcal{Z}_2) = \|\mathbf{c}_1 - \mathbf{c}_2\|, \tag{8.7}$$

where $\mathbf{c}_1$ and $\mathbf{c}_2$ are the centers of the zonotopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$, respectively. If all zonotopes have similar dimensions, the center-to-center distance closely reflects the actual spatial relationship between the sets. By constraining the number and size of the generators, the variability in zonotope dimensions is minimized, further validating the use of this approximation.

Although this approximation is less precise than the set-based distance, it significantly reduces computational overhead and aligns well with the primary goal of efficient tree expansion.

8.3.4 Steering the Sampled Node

Once a node has been linked to the tree, the RRT algorithm typically creates a new node along the line segment between the sampled node and the nearest tree node. The position of this new node is determined by a predefined step size along this line. While this approach is straightforward, it does not leverage system dynamics or the underlying structure of the reachable space, which could result in inefficient exploration.

For the zonotopic approach, this corresponds to creating a line segment between the centers of two zonotopes and defining a new center along this line at a fixed step size. However, we propose enhancing this process by incorporating reachability analysis to steer the new zonotopes within a reachable region around the closest tree node.

In the context of the RRT algorithm, we replace the fixed step size with the reachable set computed for a chosen time step. This ensures that each node added to the tree is physically realizable by the system within the specified time.

To reduce computational complexity, we steer the center of the sampled zonotope into a reduced reachable set obtained using the Minkowski difference.

The goal of this operation is to ensure that selecting any center into the reduced set will ensure that the whole zonotope is contained in the

initial reachable set. While constrained zonotopes are not closed under the Minkowski difference, approximations can be computed [142, 100].

The new center is then selected as a point within the reduced reachable set, ensuring the resulting zonotope remains inside the full reachable set. Moreover, we always like to have the steering point in the direction of the sampled zonotope. This can be achieved by creating a line zonotope between the tree node and the sampled node:

$$\mathcal{L} = \langle \mathbf{c}_1, (\mathbf{c}_2 - \mathbf{c}_1), [], [] \rangle_{CZ} \tag{8.8}$$

where $\mathbf{c}_1$ and $\mathbf{c}_2$ are the centers of the tree node and sampled node, respectively. By intersecting this line with the reduced reachable set, we generate a constrained zonotope corresponding to a line from which the vertices can be extracted to obtain the center of the steered zonotope, as illustrated in Figure 8.2. The corresponding corridor, free of obstacles, can be built as well.

8.4 Algorithm Summary

This section provides a concise summary of the algorithm's key steps as well with Fig. 8.3 illustrating the tree build with zonotopes.

1. **Initialization:** Define the start point $\mathbf{x}_s$, goal point $\mathbf{x}_g$, zonotopic obstacles $\mathcal{O}_i$, sampling region $\mathcal{S}$, and reachable set $\mathcal{R}$. Augment obstacles with the agent's size and uncertainty using the Minkowski sum:
$$\mathcal{A}\mathcal{O}_i = \mathcal{O}_i \oplus \mathcal{A} \tag{8.9}$$

2. **Sampling:** Instead of sampling individual points, generate zonotopes within the defined sampling region $\mathcal{S}$, represented as:
$$Z_{sample} = \langle c_{sample}, \mathcal{S} \rangle_Z, \tag{8.10}$$
where c_{sample} is the center.

3. **Nearest Neighbor Selection:** Identify the nearest tree node $\mathcal{Z}_{\text{nearest}}$ to the sampled zonotope $\mathcal{Z}_{\text{sample}}$ by approximating the center-to-center distance:
$$d_{\text{approx}} = \|c_{\text{sample}} - c_{\text{nearest}}\| \tag{8.11}$$

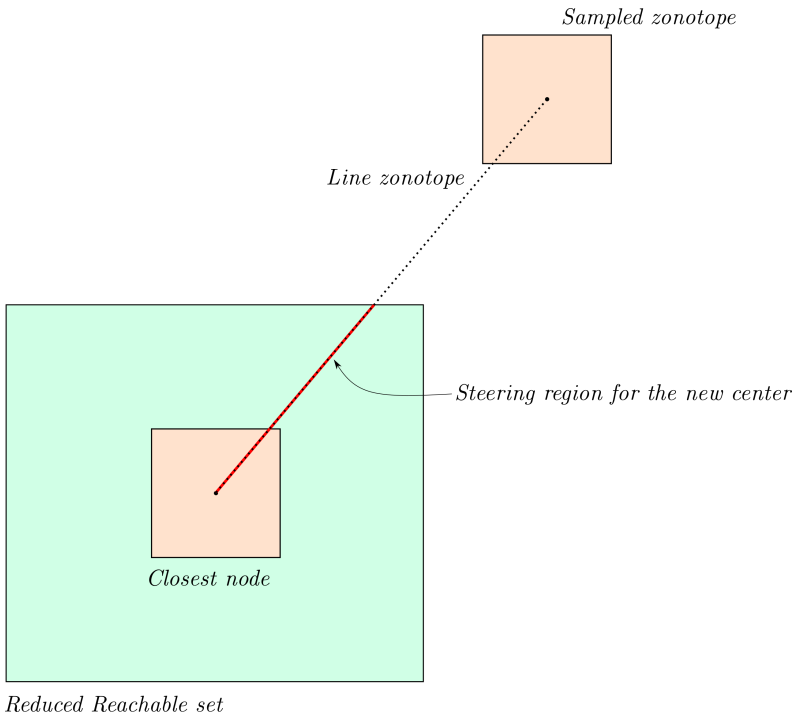
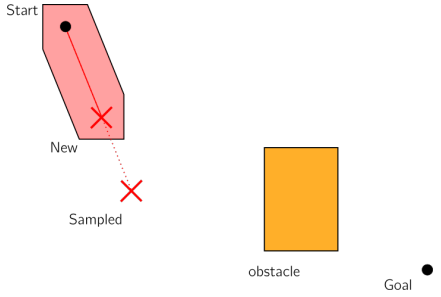
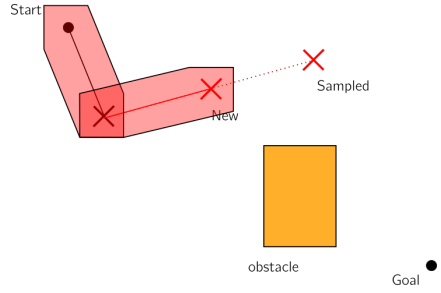


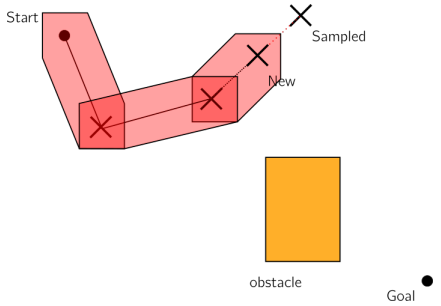
Figure 8.2: Illustration of the steering method. A reduced reachable set (light green) is attached to the center of the closest node in the tree, and a line zonotope (black dotted line) is created between the closest node and the sampled zonotope. This line intersects with the reduced reachable set, which produces the red line, a constrained zonotope from which we sample a steered center.



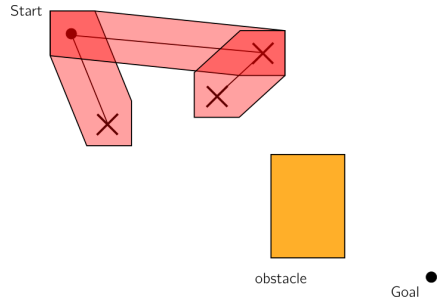
(a) First step of the RRT algorithm



(b) Once a new node is added, the algorithm start searching for a new one



(c) The process continue until a node is found



(d) At some point a rewiring operation is applied to redefine the tree with existing node

Figure 8.3: Zonotopic variant of the RRT⋆ variant. The main steps remain similar to the standard RRT⋆ algorithm, but zonotopes are used for representation and collision checking. At each iteration, a random point is sampled, and the nearest node in the tree is found. The new connection is validated using zonotope-based collision checking. If a collision is detected, the sample is discarded. Otherwise, a new point is generated using a step-size parameter or the reachability analysis and added to the tree. Finally, as shown in the rightmost diagram, the tree can be rewired if a shorter path exists.

4. **Collision Checking:** Use zonotopic algebra to verify $CH(Z_{sample}, Z_{nearest})$ does not collide with any augmented obstacle $\mathcal{AO}_i$. If not feasible, return to step 2.

5. **Steering:** Compute a new zonotope along the line segment between $\mathcal{Z}_{\text{nearest}}$ and $\mathcal{Z}_{\text{sample}}$. Intersect the line segment with a reduced reachable set and select a point along this line:

$$c_{new} \in Z_{line} \cap (\mathcal{R} \ominus G_{sample}) \tag{8.12}$$

6. **Remaining steps in RRTs algorithm** (rewiring, path construction, etc..)

8.4.1 Discussion on Convergence and Optimality of the Modified RRT* Algorithm

In this section, we examine the convergence and optimality properties of the modified RRT* algorithm. We compare these properties to those of the standard RRT* algorithm and discuss their implications for path planning in environments with geometric constraints and uncertainties.

Convergence Properties

In the standard RRT* algorithm, convergence is guaranteed because dense sampling allows the tree to approximate any feasible path with sequences of short line segments. In our modified RRT*, convergence is reinterpreted in terms of finding a feasible corridor — a sequence of convex hulls (derived from zonotopes) that are entirely obstacle-free.

Specifically, if a feasible corridor exists (i.e., there is a sequence of convex hulls between zonotopes that avoids obstacles), the modified RRT* will find it with probability approaching one as the number of samples increases, provided that the sampling is sufficiently dense. When the generator matrix is chosen to be null, the zonotopes reduce to points, and the convex hulls reduce to line segments, thereby recovering the standard RRT* behavior. Thus, the modified RRT* generalizes the standard RRT*.

However, when the generator matrix **G** is non-null, the algorithm may fail to find a corridor in environments with narrow passages that are smaller than the size of the zonotopes or their convex hulls, even if a feasible path

exists for a point robot. This is not a limitation in practice, as the zonotopes are intended to represent the robot's physical size or the required safety margins. In such cases, the absence of a feasible corridor correctly indicates that no safe path exists for the robot.

Thus, the modified RRT* retains a form of convergence: it finds a feasible corridor if one exists under the given zonotopic constraints, analogous to the standard RRT* finding a feasible path when one exists.

Optimality Properties

In the standard RRT*, asymptotic optimality ensures that the algorithm converges to the globally optimal path as the number of samples tends to infinity. In the modified RRT*, the notion of optimality is more nuanced.

Key points include:

- The modified RRT* can be designed to find an "optimal corridor" based on a cost function defined for sequences of convex hulls. However, this optimal corridor does not necessarily contain the globally optimal path. In particular, the globally optimal path may lie outside the corridor if it traverses narrow passages that the convex hulls cannot accommodate due to their predetermined size.

Therefore, while the modified RRT* can be tuned to find an optimal corridor according to a specified cost, it does not guarantee convergence to the globally optimal path. This distinction is crucial: in many applications, such as planning under uncertainty or with strict safety constraints, the primary goal is to find a robust, safe corridor rather than the shortest possible path.

Practical Implications and Applications

Despite the differences in optimality, the modified RRT* offers significant advantages in practical scenarios:

- **Robust Planning**: By planning with corridors, the algorithm inherently accounts for uncertainties in state, control, or the environment, ensuring that the entire corridor remains free of obstacles.

- **Safety Margins**: The use of zonotopes allows the algorithm to model the robot's physical footprint or enforce required safety buffers, making the approach particularly suitable for real-world applications where collision avoidance is critical.
- **Generalization**: Since setting $\mathbf{G} = \mathbf{0}$ recovers the standard RRT*, the modified RRT* provides a flexible framework that can be tuned to balance robustness and performance based on specific application requirements.

In scenarios where ensuring safety and robustness is more critical than finding the globally optimal path, the modified RRT* serves as a practical and effective solution for autonomous navigation in cluttered environments.

8.5 Numerical results

We consider a two-dimensional environment where an agent starts at a designated initial position, $\mathbf{x}_s = [-3, 0.5]^T$, and aims to reach a target goal point, $\mathbf{x}_g = [7, 2]^T$. The environment contains two obstacles represented as zonotopes that block the straight-line path between the start and goal positions. The zonotopes for the obstacles are defined as:

$$\mathcal{O}_1 = \langle \mathbf{c}_1, G_1 \rangle_{CZ}, \quad \mathbf{c}_1 = [1, 1]^T, \quad G_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$
$$\mathcal{O}_2 = \langle \mathbf{c}_2, G_2 \rangle_{CZ}, \quad \mathbf{c}_2 = [3, 3]^T, \quad G_2 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}.$$

We define an additional zonotope to account for the agent's physical size and uncertainties in its displacement. The center of this zonotope is determined dynamically by the algorithm, while the generator matrix is predefined:

$$\mathcal{A} = \langle c_3, G_3 \rangle_{CZ}, \quad c_3 = [0, 0]^T, \quad G_3 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

The combined effects of the agent size and displacement uncertainties are integrated into the algorithm through augmented obstacles, which are computed as:

$$\mathcal{AO}_i = \mathcal{O}_i \oplus \mathcal{A}, \tag{8.13}$$

where $\oplus$ denotes the Minkowski sum.

The sampling region for the path-planning algorithm is also defined as a zonotope:

$$\mathcal{S} = \langle c_4, G_4 \rangle_{CZ}, \quad c_4 = [0, 0]^T, \quad G_4 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

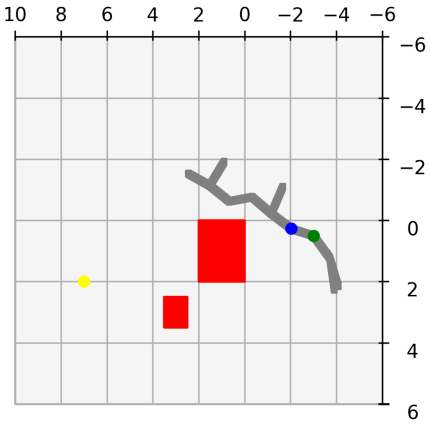
Additionally, the agent's reachable set at any given time is represented by another zonotope:

$$\mathcal{R} = \langle c_5, G_5 \rangle_{CZ}, \quad c_5 = [0, 0]^T, \quad G_5 = \begin{bmatrix} 1.2 & 0 \\ 0 & 1.2 \end{bmatrix}.$$

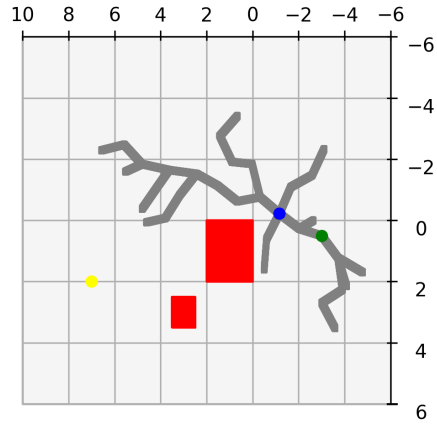
The primary objective is to compute a feasible path from $\mathbf{x}_s$ to $\mathbf{x}_g$ that avoids obstacles while respecting reachability constraints, vehicle dimensions, displacement uncertainties, and zonotopic sampling considerations.

Figure 8.4 illustrates the path generated by the algorithm at various time instances as the agent navigates the environment. The resulting zonotopic corridor provides a feasible region of states for further optimization using approaches such as Model Predictive Control (MPC). The zonotopic representation enables efficient collision checking and constraint handling.

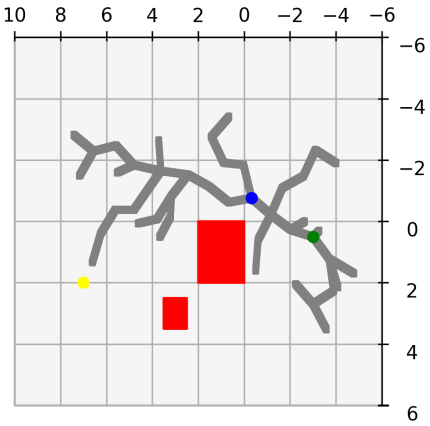
The zonotopic framework can further enhance exploration by adjusting the sampling region and reachable set parameters. Figures 8.6 and 8.7 demonstrate the algorithm's performance with increased values for these parameters. Each parameter influences the algorithm differently. Choosing a higher step size (or a larger reachable set) results in greater spacing between nodes, potentially creating longer branches. Conversely, increasing the sampled region makes each branch broader. Selecting optimal parameters largely depends on the specific characteristics of the environment. The expanded sampling region and reachable set allow the



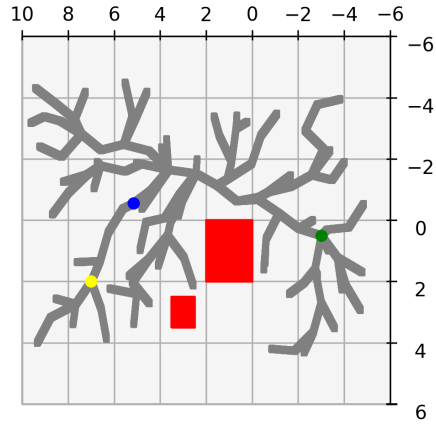
(a) Tree after 1 second.



(b) Tree after 2 seconds.



(c) Tree after 3 seconds.



(d) Tree after 10 seconds.

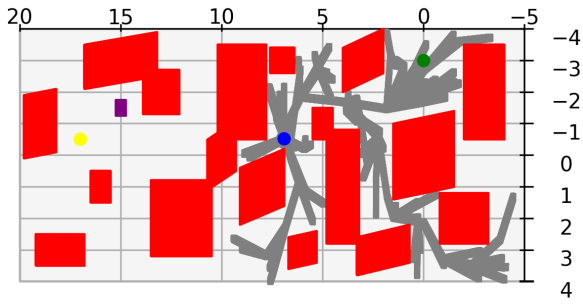
Figure 8.4: Illustration of the real-time Rapidly-Exploring Random Tree (RRT*) algorithm with zonotopes. The starting point (green), goal point (yellow), obstacles (red), agent (blue), and tree (gray) are shown. In (a), the tree begins to form, allowing initial movement along collision-free paths. By (b), the tree expands significantly, guiding the agent closer to the goal. In (c) and (d), the tree fully explores the environment, enabling the agent to reach the goal without collision.

algorithm to explore almost the entire environment, leaving only the areas contained within the augmented obstacles unexplored. Figure 8.5 illustrates a denser environment, similar to a maze, where the algorithm finds its way toward the final goal. In this specific example, a moving obstacle is represented in purple. The real-time version of the RRT⋆ [140] algorithm is capable of handling such cases, where if the moving obstacle enters into collision with the existing tree, the affected branches are pruned and rewired if possible. Dense environments inherently increase collision risks due to the proximity of obstacles, a problem that becomes particularly pronounced in narrow passages. In the context of the zonotopic RRT* algorithm, which samples entire regions (zonotopes) rather than individual points, a significant challenge arises when these sampled regions are too large to fit through tight spaces. If the sampled zonotope exceeds the width of a narrow passage, it will intersect with obstacles, preventing the algorithm from expanding the tree through that region and tracking a feasible path. This limitation can be adjusted by setting the dimensions of the sampled region to be smaller or even set to zero (which would return to the standard algorithm).

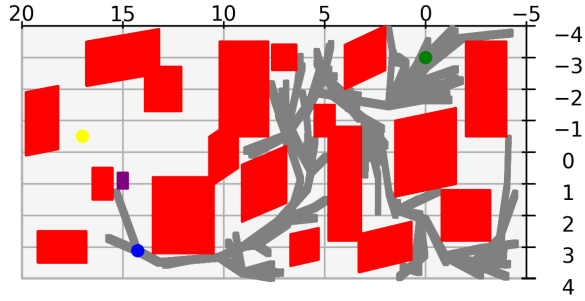
These simulations showcase the robustness of the proposed method in generating collision-free paths in environments with obstacles. The results highlight how parameter tuning can balance exploration and collision avoidance, particularly in dense environments where careful adjustments are essential for constructing non-colliding zonotopes.

Traditional RRT* variants typically rely on point-based sampling and simplified obstacle representations (e.g., bounding boxes or spheres), which can limit their efficiency and accuracy in complex, non-convex environments. In contrast, our method employs zonotopes to model both obstacles and sampling regions, enabling more precise geometric representations and efficient collision checking through algebraic operations. For example, in the dense, maze-like environment with a moving obstacle (Figure 8.5), our algorithm successfully generated a collision-free path, demonstrating its robustness in dynamic and cluttered scenarios. The use of zonotopic corridors further provides a compact and feasible representation of the state space, which is advantageous for real-time applications requiring rapid decision-making.

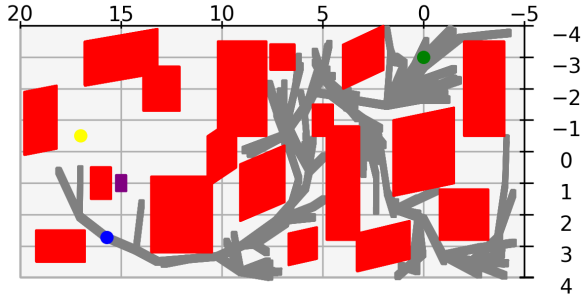
However, our approach is not without limitations. The computational complexity of zonotopic operations may increase in high-dimensional spaces or with a large number of generators, potentially offsetting some efficiency



(a) Tree after 15 seconds.



(b) Tree close to the moving obstacle.



(c) Tree slightly pruned after the moving obstacle pass through a branch.

Figure 8.5: Illustration of the real-time Rapidly-Exploring Random Tree (RRT*) algorithm with zonotopes in a dense environment. The starting point (green), goal point (yellow), obstacles (red), agent (blue), tree (gray), and moving obstacle (purple) are shown. In (a), the tree is finding its way in the maze. In (b), the tree is almost reaching the goal point, but a moving obstacle blocks the path and overlaps with a branch of the tree. By (c), the tree expands toward the goal and prunes itself where the moving obstacle overlapped with the tree, allowing it to reach the goal safely.

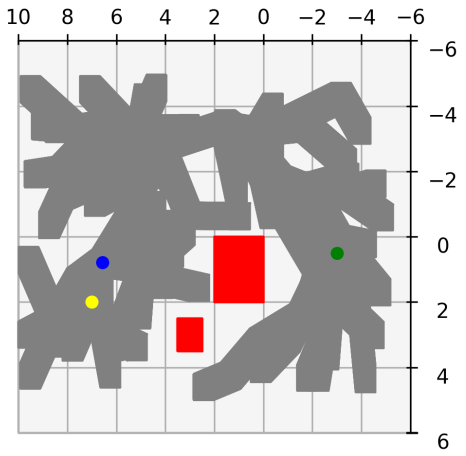


Figure 8.6: Tree after 10 seconds with medium exploration parameters. The sampling region parameter increases from 0.1 to 0.4, and the reachable set parameter rises from 1.2 to 2. This adjustment enhances the explored space while avoiding collisions.

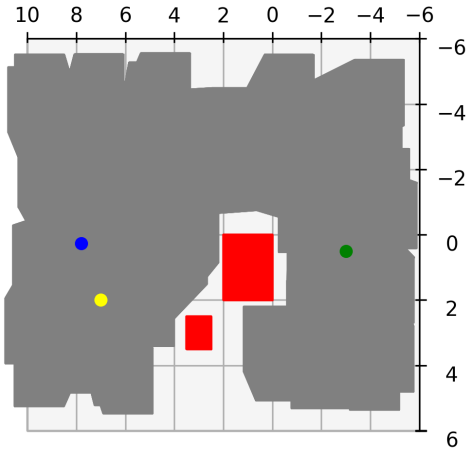


Figure 8.7: Tree after 10 seconds with large exploration parameters. The sampling region parameter increases from 0.1 to 1, and the reachable set parameter rises from 1.2 to 4. This adjustment significantly enhances the explored space while avoiding collisions.

gains compared to simpler methods. Additionally, the approximation used in nearest neighbor selection, based on center-to-center distances, may occasionally lead to suboptimal tree expansions, as it does not fully capture the true proximity between zonotopes. Importantly, the variants of RRT*—including ours—are not mutually exclusive; rather, some can be combined together to achieve synergistic improvements.

8.6 Experimental Validation

The proposed algorithm is rigorously validated through an experimental setup designed to assess its performance in a controlled indoor environment. The chosen platform for these experiments is the Crazyfly 2.1 drone, depicted in Figure 8.8. This drone operates within a dedicated indoor facility, as shown in Figure 8.9, which is equipped with a high-precision camera detection system for accurate position feedback. A central control computer manages the experimental process, executing the path planning algorithm and transmitting real-time commands directly to the Crazyfly drone. To facilitate clear visualization and analysis of the algorithm's behavior, the experimentation was conducted using a series of static obstacles, configured to form a two-dimensional maze.

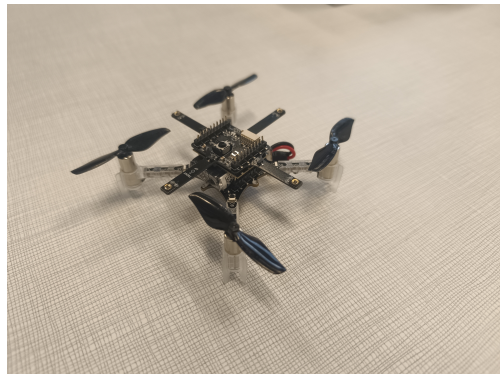


Figure 8.8: Drone Crazyfly 2.1 used for experimental validation.

Several key factors were considered and controlled during the experimentation to ensure a consistent and representative validation. Firstly, the physical dimensions of the Crazyfly drone were bounded by a box of 12 cm $\times$ 12 cm $\times$ 10 cm. Secondly, the camera-based position feedback

Figure 8.9: Indoor operation room setup, featuring an external computer and camera system for precise position feedback.

system exhibited high precision, with an estimated upper bound for positional uncertainty set at ± 5 cm. Furthermore, the sampling parameter for the environment exploration was set to 0.05 s, corresponding to sampled squares with side lengths of 0.1 m. Finally, the drone's control loop operated with a sampling time of 0.1 seconds, ensuring timely command execution.

Figure 8.10 illustrates the drone's successful navigation within the small maze, effectively identifying a collision-free path to reach the designated goal point while actively avoiding obstacles. The visualization further demonstrates that the underlying tree structure, generated by the algorithm, effectively captures the available free space for the drone's movement, highlighting the efficacy of the proposed path planning approach.

8.7 Conclusion

This chapter proposes extending the Rapidly-Exploring Random Tree (RRT) algorithm by incorporating a zonotopic framework to enhance path planning in complex environments. By leveraging the mathematical properties of zonotopes and constrained zonotopes, we demonstrated how this approach could be incorporated into RRT algorithm and can effectively navigate non-convex spaces.

Using zonotopic sampling regions allows for greater exploration of the search space, enabling the generation of large corridors even in crowded

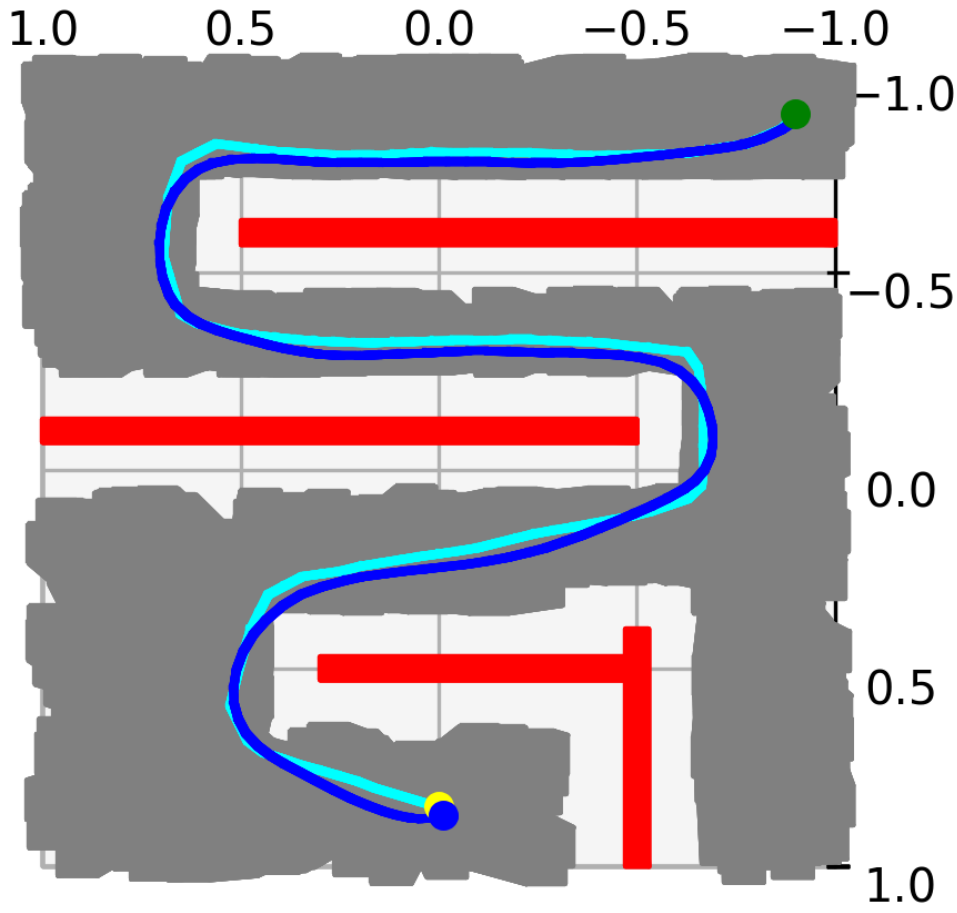


Figure 8.10: Result of the experimental validation: The Crazyfly drone successfully navigates the 2D maze, avoiding obstacles and reaching the goal. The starting point (green), goal point (yellow), obstacles (red), agent trajectory (blue), the node to node trajectory reference provided to the crazy fly (cyan) and the zonotopic tree (gray) are shown.

environments with obstacles. Simulations illustrated the versatility of the proposed method, showing that with appropriate parameter adjustments, the algorithm can balance exploration and constraint satisfaction. Furthermore, the generated zonotopic corridors provide a feasible and compact representation of the state space for integration with optimization-based controllers such as Model Predictive Control.

Chapter 9

Partitioning and Distributed Tube-MPC for UAV Swarms

9.1 Introduction

Thus far, our research has concentrated on controlling a single UAV. However, as robotic systems evolve and their applications expand—ranging from disaster response to planetary exploration—swarms of robots are increasingly integral to these processes. This fundamental shift necessitates scalable methods to address the inherent challenges of swarm robotics. Controlling a large swarm of UAVs remains a formidable task due to the high dimensionality of the problem and the demand for real-time computation [143]. Many different approaches have been used to address these challenges, often in a multi-layered [144] and application-oriented manner.

One of the main features of these swarm applications is formation control, a type of multi-agent control strategy that involves coordinating the movements of a group of agents to maintain a specific geometric shape or formation while performing a task [145]. A common consensus-based approach for maintaining formation control involves the exchange of positional information among agents [146]. Alongside this, collision avoidance [147] is a critical consideration to ensure the safety of each agent within the swarm. Formation control strategies can be broadly classified based on their underlying principles. These include force field methods, where agents interact based on repulsive fields to achieve emergent for-

mations [148]; model predictive control (MPC), which allows for defining hierarchical formations and optimizing trajectories [149]; and leader-follower methods, which simplify the control problem by designating a leader for the rest of the agents [150]. While leader-follower approaches reduce complexity, they may introduce issues such as single-point failures and limited formation flexibility. Other approaches to formation control encompass behavior-based methods and virtual structure methods.

As the size of the UAV formation grows, so does the complexity of the control problem. This escalating complexity renders a centralized control approach computationally intensive for each agent to solve independently, making real-time implementation impractical or even infeasible for real-world deployment in large swarms. To address this critical hurdle, particularly in the context of large-scale formations, this chapter proposes a partitioning method rooted in graph theory. This approach divides the swarm into smaller, more manageable subgroups, thereby significantly reducing the dimensionality of the formation control problem for each subset of agents. Similar partitioning strategies have been explored for various applications, such as regrouping wind turbines based on wind conditions to enhance efficiency in wind farms [151], or event-triggered partitioning for non-centralized MPC strategies [152, 153].

The focus of this chapter is to develop a robust solution for formation control in UAV swarms. Our proposed methodology involves partitioning the swarm into smaller subgroups, each governed by a distributed Tube Model Predictive Control (Tube-MPC) strategy. This approach is augmented with a communication network between partitions, ensuring the exchange of critical information between subgroups. This strategy enables each subgroup to maintain its formation while coordinating with others, thereby ensuring cohesive swarm behavior.

9.2 Partitioning the swarm

When designing control strategies for multi-agent systems, one quickly realizes that the number of variables involved can become very large, leading to significant optimization problems. These large-scale problems are generally unsuitable for real-time applications as they require fast computation. A partitioning approach allows us to tackle the overall control problem in a more efficient manner as illustrated in Fig. 9.1 and Fig. 9.2.

9.2.1 Graph Theory

We begin by reviewing some concepts of graph theory [154]. A graph $\mathcal{G}$ is a mathematical structure consisting of a set of nodes $\mathcal{F}$ and a set of edges $\mathcal{E}$ that connect them. Formally, $\mathcal{E} \subseteq \mathcal{F} \times \mathcal{F}$, where an edge $(i, j) \in \mathcal{E}$ indicates that node i receives information from node j.

In our problem, we consider a set of indices $\mathcal{K} = \{1, \dots, m\}$, representing the $m \in \mathbb{Z}_0^+$ possible partitions to which each agent can belong. We define the partitioning $\mathcal{P} = \{\mathcal{F}^l : l \in \mathcal{K}\}$, each partition $\mathcal{F}^l$ forms a subgraph where the nodes within the same partition communicate directly with each other. The partitioning is generated such that each node belongs to exactly one partition, i.e., $\bigcup_{l\in\mathcal{K}} \mathcal{F}^l = \mathcal{F}$ and $\bigcap_{l\in\mathcal{K}} \mathcal{F}^l = \emptyset$, therefore there is no overlapping among the resultant subsystems.

To specify whether a node $i \in \mathcal{F}$ belongs to a partition $\mathcal{F}^l, l \in \mathcal{K}$, we define the function $\delta : \mathcal{F} \times \mathcal{F}^l \to \{0, 1\}$ as follows:

$$\delta_{il} = \begin{cases} 1, & \text{if } i \in \mathcal{F}^l \\ 0, & \text{otherwise.} \end{cases} \tag{9.1}$$

This function will be useful in defining the partitioning algorithm in the next section.

The graph $\mathcal{G}$ possesses a few characteristics that define it and will help decide the partitioning. Consider that each node i has a position in space $\mathbf{p}_i \in \mathbb{R}^{n_p \times 1}$. A distance function is built to evaluate the distances between the nodes in the graph as $dp : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_0^+$ and the resulting application of this function to all the nodes is contained in a distance matrix **DP**, where each element $dp_{ij} = dp(i, j)$ and defined as:

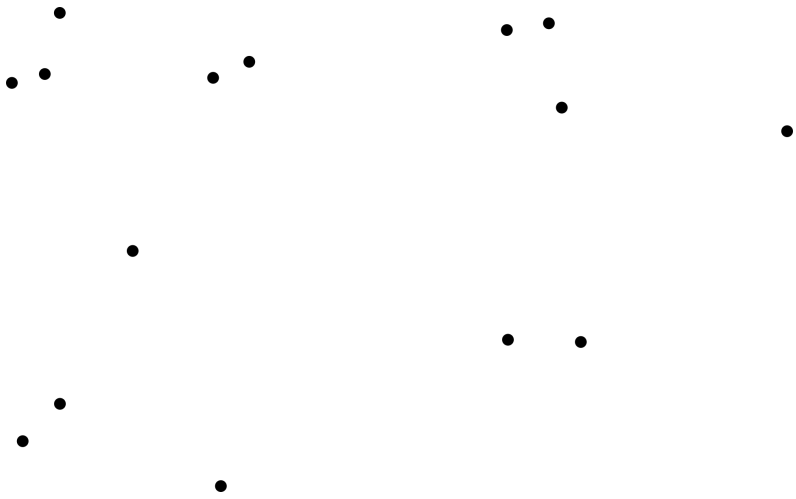
$$dp_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\|_2 \tag{9.2}$$

$\|.\|_2$ denotes the Euclidean norm (2-norm). Let us define a second function $dl : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_0^+$ that represents the distribution of the agents respective to a center, a chosen parameter such as the central position of the swarm. The application of this function to all the nodes yields a distance matrix **DL** where each element $dl_{ij} = dl(i, j)$:

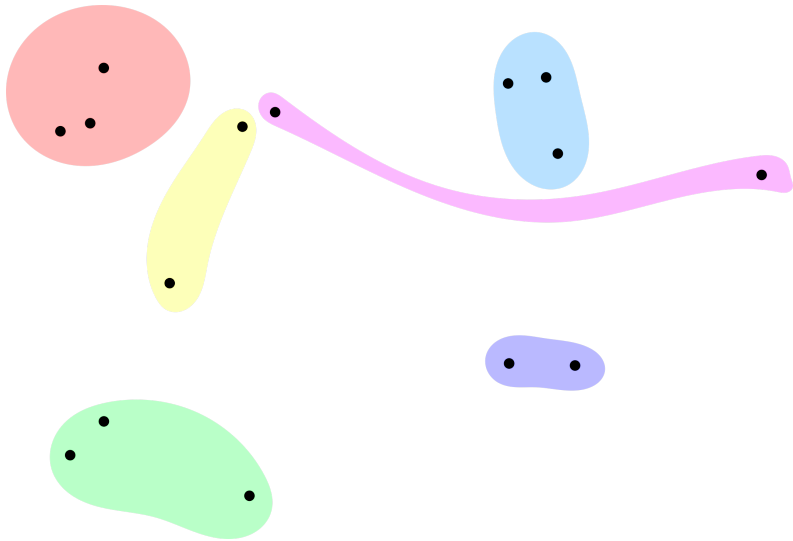
$$dl_{i,j} = \|\|\mathbf{p}_i - \mathbf{c}\|_2 - \|\mathbf{p}_j - \mathbf{c}\|_2\|_2 \tag{9.3}$$

The number of nodes in a partition as $na : \mathcal{F} \to \mathbb{Z}_0^+$, defined by:

$$na_l = \sum_{i\in\mathcal{F}} \delta_{il} \tag{9.4}$$

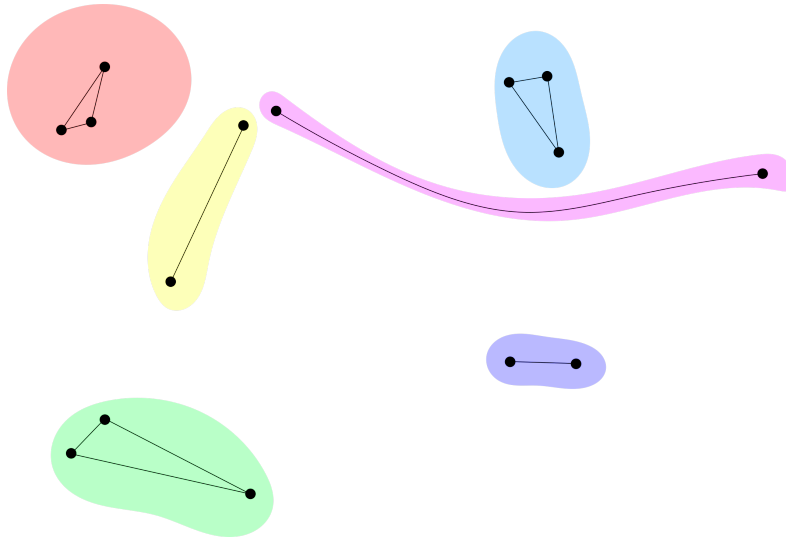


(a) Original swarm with fifteen agents, each agent is a node of the graph.

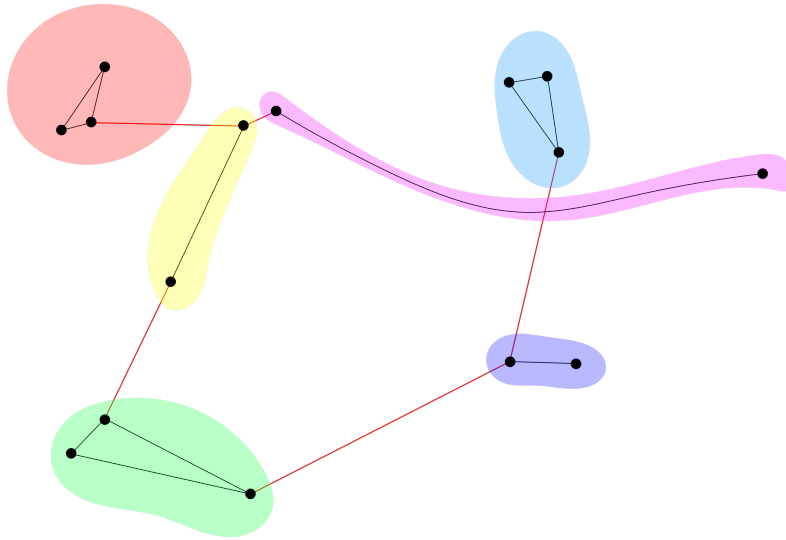


(b) The partition algorithm regroups the agents considering the optimal cost.

Figure 9.1: Algorithm partitioning and communication network (Part I)



(a) Each partition is a complete graph and has a distributed controller.



(b) A communication graph between partitions is established.

Figure 9.2: Algorithm partitioning and communication network (Part II)

Avoiding large disparities in the number of nodes in a partition will be useful.

9.2.2 Partitioning problem

Now consider the graph $\mathcal{G}$ and define its partitioning through the following optimization problem:

$$\min_{\delta_{il}} \sum_{k} we_k f_k(\delta_{il}) \tag{9.5a}$$

$$\sum_{i\in\mathcal{F}} \delta_{il} \geq 1, \quad \forall l \in \mathcal{K} \tag{9.5b}$$

$$\sum_{l\in\mathcal{K}} \delta_{il} = 1, \quad \forall i \in \mathcal{F} \tag{9.5c}$$

In (9.5), the variable δ enables the creation of different subsystems. The objective function is a weighted sum of several components, which are defined below. The weights we_k and functions f_k are designed to reflect the desired properties of the partitions, such as minimizing the distance between agents, their distribution, and the number of agents in each partition. Constraint (9.5b) ensures that each partition is non-empty, while constraint (9.5c) guarantees that each node is assigned to exactly one partition, with no overlapping between partitions.

First, the agents that are close to each other can be grouped into the same partition:

$$f_1 = \sum_{l\in\mathcal{K}} \sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{F}} dp_{ij}\delta_{il}\delta_{jl}. \tag{9.6}$$

Second, a layered partition can be created by grouping agents at the same distance of a chosen center:

$$f_2 = \sum_{l\in\mathcal{K}} \sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{F}} dl_{ij}\delta_{il}\delta_{jl}. \tag{9.7}$$

Third, it is preferable if the agents equidistribute in the several partitions:

$$f_3 = \sum_{l=1}^{m-1} \sum_{l^*=l+1}^{m} |\sum_{i\in\mathcal{F}} \delta_{il} - \sum_{j\in\mathcal{F}} \delta_{jl^*}|. \tag{9.8}$$

|.| denotes the absolute value. The optimization problem in (9.5) is nonlinear due to quadratic terms in the first two components of the cost functions and absolute values in the third component. By introducing new binary variables and constraints, we can transform the nonlinear problem into a linear one, which can be solved more efficiently [155]. To this end, let us introduce a new binary variable $\delta_{ijl} = \delta_{il}\delta_{jl}$, so that:

$$f_1 = \sum_{l\in\mathcal{K}}\sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{F}} d_{ij}\delta_{ijl} \tag{9.9}$$

$$-\delta_{il} + \delta_{ijl} \leq 0, \quad \forall i, j \in \mathcal{F}, \forall l \in \mathcal{K} \tag{9.10}$$

$$-\delta_{jl} + \delta_{ijl} \leq 0, \quad \forall i, j \in \mathcal{F}, \forall l \in \mathcal{K} \tag{9.11}$$

$$\delta_{il} + \delta_{jl} - \delta_{ijl} \leq 1, \quad \forall i, j \in \mathcal{F}, \forall l \in \mathcal{K}. \tag{9.12}$$

The expression of f_1 has been linearized by introducing the new variables and additional constraints, ensuring that the equality between the variables holds in the binary case (this transformation comes at the cost of introducing additional variables and constraints). The same strategy can be applied to linearize the second component of the cost function, f_2.

Furthermore, introducing a slack variable can transform a problem containing absolute values into a linear problem [156]. In this case, the slack variable, denoted as ρ_{ll^*}, represents the absolute difference between two sums of decision variables, δ_{il} and δ_{jl^*}. The slack variable can take any value, except that when it is positive, it must not exceed the actual difference, and when it is negative, it must not fall below the negative value of the difference. This is enforced by the following constraints:

$$f_3 = \sum_{l=1}^{m-1}\sum_{l^*=l+1}^{m} \rho_{ll^*} \tag{9.13}$$

$$\sum_{i\in\mathcal{F}} \delta_{il} - \sum_{j\in\mathcal{F}} \delta_{jl^*} \leq \rho_{ll^*}, \quad 1 \leq l \leq m-1 \tag{9.14}$$

$$\sum_{i\in\mathcal{F}} \delta_{il} - \sum_{j\in\mathcal{F}} \delta_{jl^*} \geq -\rho_{ll^*}, \quad 1 \leq l \leq m-1. \tag{9.15}$$

9.2.3 Weighting

The weights assigned to each term in an optimization problem play a crucial role in determining the optimal solution. In this section, we provide

some guidance on how to choose the tuning parameters.

Firstly, we normalize the different costs by adjusting the weights. For the first component, we set the weight as

$$we_1 = \frac{\alpha_1}{\sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{F}} dp_{ij}}. \tag{9.16}$$

This normalization ensures that the first component is adjusted with respect to the distances between the agents, and the same configuration of agents results in the same value for the first component, regardless of the unit length chosen.

Similarly, we normalize the second component f_2 by setting the weight as

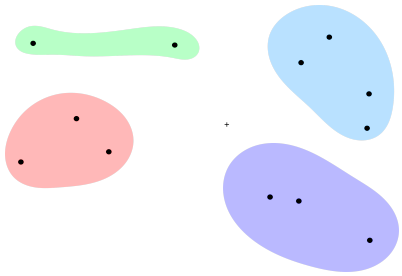
$$we_2 = \frac{\alpha_2}{\sum_{i\in\mathcal{F}} \sum_{j\in\mathcal{F}} dl_{ij}}. \tag{9.17}$$

Additionally, we tune the third term by dividing it by the total number of nodes in the graph. It is important to note that our goal is to partition the swarm to avoid excessive computational effort for the controller of each partition. Therefore, the third component must be highly weighted to prevent inefficient partitioning.

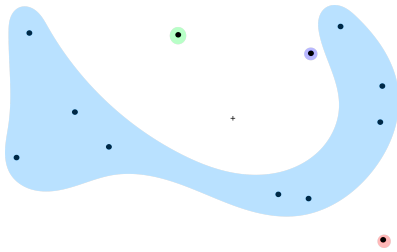
The choice of the weights α_1, α_2, and α_3 depends on the specific problem and the desired trade-off between the different components of the cost function. A sensitivity analysis can be performed to study the effect of the weights on the optimal solution and to choose appropriate values.

9.2.4 Partitioning results

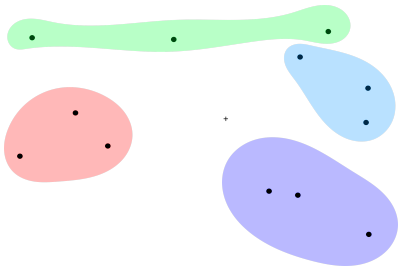
We apply the partitioning algorithm to a group of 12 agents that must be divided into four partitions. The results are presented in Fig. 9.3 for various values of α_1, α_2, and α_3. In particular, we focus on the impact of the third component of the cost function, which significantly affects the partitioning results. By adjusting the weight of this component, we can create circular partitions. The center, represented by a cross in the figure, can be chosen arbitrarily; here, it has been placed in the middle of the swarm. This flexibility allows us to adapt the partitioning strategy to different scenarios and requirements.



(a) $\alpha_1 = 100$, $\alpha_2 = 1$ and $\alpha_3 = 1$. The partitioning focuses on regrouping the agents close to each other.



(b) $\alpha_1 = 1$, $\alpha_2 = 100$ and $\alpha_3 = 1$. The partitioning focuses on regrouping agents that are at the same distance with respect to the cross + point.



(c) $\alpha_1 = 1$, $\alpha_2 = 1$ and $\alpha_3 = 100$. The partitioning focuses on balancing the number of agents in each group.

Figure 9.3: Partitioning example of a group with 12 agents into four partitions

9.2.5 Number of Partitions

One fundamental parameter of the algorithm is the number of partitions to choose. This number can be easily bounded by an upper and lower limit. First, it is clear that the maximum number of partitions is equal to the number of agents in the swarm, as this would correspond to each agent being alone in its partition. Adding more partitions would only create empty partitions, as no overlapping is allowed in the optimization problem. A lower bound for this parameter is determined by the computational capabilities of the agents in the swarm. Initially, agents are grouped into smaller clusters since they have finite resources and cannot communicate with or consider the entire network in their computations and control scheme. By defining a maximum number of agents that can be handled within a single system, we can effectively determine the minimum required number of partitions. This minimum is given by the total number of agents divided by the maximum number of agents that a single agent can manage.

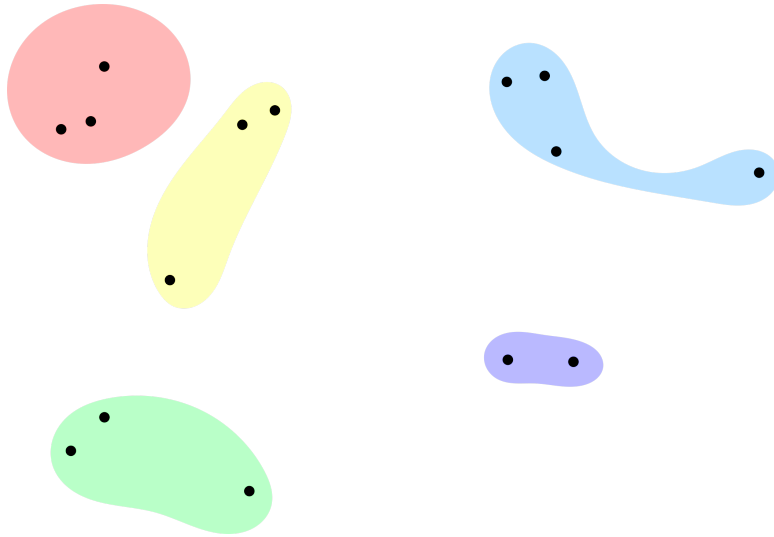
Figures 9.4a and 9.4b illustrate the results of partitioning with different numbers of partitions. When too many partitions are chosen, compromises must be made, leading to partitions that appear unnecessary, such as the pink partition in Figure 9.4b.

9.2.6 Time-varying partitioning

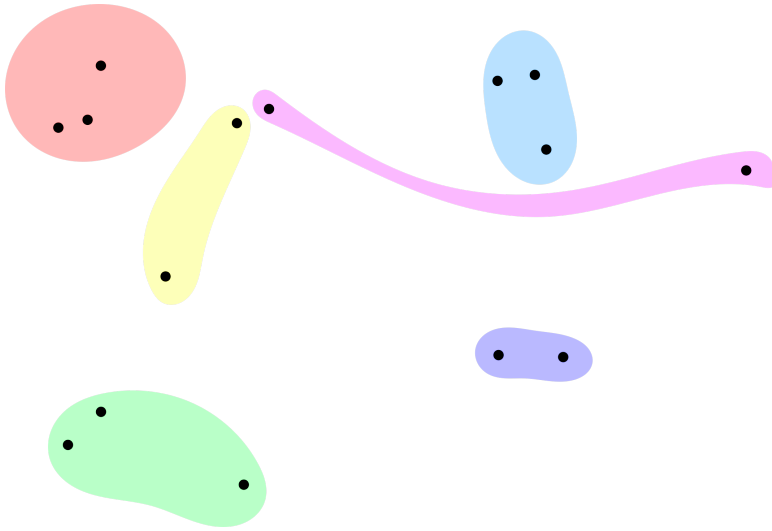
Time-varying partitioning with event triggering is an effective way to adapt the partition of a group of agents in response to changing conditions or requirements. For instance, if a new formation is required or a higher command instructs the agents to go around an object, the partitioning algorithm would be re-executed to generate a new partition that satisfies the new requirements. This time-varying partitioning strategy enables the agents to quickly and efficiently adapt to changing circumstances, improving the overall performance and flexibility of the system.

9.3 Communication between the partition

The partitioning strategy aims to build efficient distributed controllers for each subgroup. However, it is crucial to maintain information sharing be-



(a) Partionning into 5 partitions.



(b) Partionning into 6 partitions.

tween partitions to ensure overall swarm coherence and coordination. In this section, we present an approach to establishing a partition communication network that optimizes information flow while maintaining connectivity across the entire swarm.

9.3.1 Graph Representation of Partition Communication

Consider a graph $\mathcal{G} = (\mathcal{F}, \mathcal{E})$, where $\mathcal{F}$ is the set of m nodes representing partitions, and $\mathcal{E}$ is the set of edges representing communication links between partitions. The adjacency matrix $\mathbf{A}_d \in \{0,1\}^{m\times m}$ defines the graph structure, with elements a_{ij} such that

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

We define the graph to be undirected, ensuring $a_{ij} = a_{ji}$ for all $i, j \in \{1, \dots, m\}$. The adjacency matrix takes the form:

$$\mathbf{A}_d = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1m} \\ a_{21} & 0 & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & 0 \end{bmatrix}. \tag{9.18}$$

To model the practical constraints of the communication network, we introduce a distance function $d_p : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_0^+$. This function represents the minimum distance between any two agents from different partitions, i.e.,

$$d_p(i,j) = \min_{u\in l, v\in l^*} dp(u,v), \tag{9.19}$$

where l and l^* are two different partitions, and $dp(u,v)$ is the Euclidean distance between agents u and v.

9.3.2 The optimization problem

Our goal is to find the optimal communication network that balances a distance cost. We formulate this problem as the following binary integer programming problem:

$$\min_{a_{ij}} \sum_{i=1}^{m} \sum_{j=i+1}^{m} a_{ij} d_p(i,j) \tag{9.20a}$$

$$\text{subject to } \sum_{j=1}^{m} a_{ij} \geq 1, \ \ \forall i \in \{1,\dots,m\} \tag{9.20b}$$

$$a_{ij} = a_{ji}, \ \ \forall i,j \in \{1,\dots,m\} \tag{9.20c}$$

$$a_{ii} = 0, \ \ \forall i \in \{1,\dots,m\} \tag{9.20d}$$

$$\sum_{k=1}^{m} (\mathbf{A}_d^k)_{ij} \geq 1, \ \ \forall i,j \in \{1,\dots,m\}, i \neq j \tag{9.20e}$$

$$a_{ij} \in \{0,1\}, \ \ \forall i,j \in \{1,\dots,m\}. \tag{9.20f}$$

Constraint (9.20b) imposes that all partitions must have at least one active communication with any other partition in the network to avoid partition isolation. Constraint (9.20e) imposes that a path always exists between two partitions in the network so that no sub-group of partition is isolated from the rest of the network [157].

9.3.3 Results

Using the third case generated by the partitioning algorithm in Fig 9.3c, the communication network between the partitions is built as shown in Fig 9.5. We clearly see that the constraints are respected: the closest distances between partitions with respect to agent positions are well applied through the $d_p(i,j)$ function. Moreover, every partition is linked to the network, allowing access to all other partitions.

9.4 Distributed Tube Model Predictive Control

Distributed Tube Model Predictive Control extends the principles of Tube-MPC to large-scale systems composed of interconnected subsystems. This

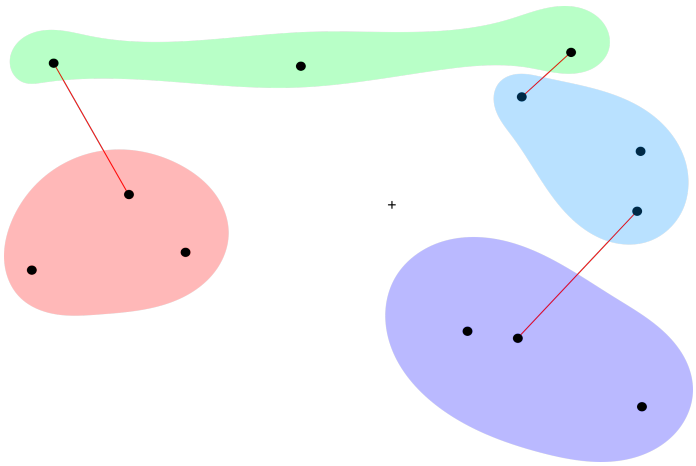


Figure 9.5: Communication network built between partition. Red lines represent the communication path.

approach is particularly useful for complex systems where centralized control might be impractical due to computational limitations and/or communication constraints.

The overall system is decomposed into multiple subsystems, each with its own local controller. These local controllers cooperate to achieve global control objectives while respecting individual constraints. The large-scale system is divided into N_s subsystems such that $i \in \{1, ..., N_s\}$, each described by

$$\mathbf{x}_i(k+1) = \mathbf{f}_i(\mathbf{x}_i(k), \mathbf{u}_i(k), \mathbf{w}_i(k)), \tag{9.21}$$

where $\mathbf{x}_i$, $\mathbf{u}_i$, and $\mathbf{w}_i$ are the state, input, and disturbance vectors of subsystem i, respectively.

Each subsystem implements its own Tube MPC, maintaining a local invariant set for its error system. This localized approach allows for robust control at the subsystem level, with the error system constrained within a local invariant set $\mathcal{L}_i$, i.e.,

$$\mathbf{e}_i(k+1) = \mathbf{x}_i(k+1) - \boldsymbol{\xi}_i(k+1) \in \mathcal{L}_i, \tag{9.22}$$

where $\boldsymbol{\xi}_i$ represents the nominal state of subsystem i. The interconnected nature of the subsystems necessitates the consideration of coupling constraints. These constraints, expressed as $\mathbf{g}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \leq \mathbf{0}$, are incorporated into each local MPC problem, ensuring that the actions of one subsystem

do not adversely affect its neighbors. The process involves iterative communication and optimization among local controllers to solve the global control problem.

9.4.1 Application to a UAV partition

Consider n_o UAV agents equipped with an inner controller for which the whole dynamics are given by

$$\dot{\mathbf{p}}_i = \mathbf{p}_i + \mathbf{w}_{p,i} \tag{9.23}$$
$$\dot{\mathbf{v}}_i = \mathbf{u}_i + \mathbf{w}_{v,i}, \tag{9.24}$$

where $\mathbf{p}_i$ describes the three-dimensional position of agent i along three reference axes, $\mathbf{v}_i$ represents the 3 velocities along that same axes, both of them are contained in a vector $\mathbf{x}_i = [\mathbf{p}_i, \mathbf{v}_i] \in \mathbb{R}^{6\times 1}$. The dynamics are driven by the inputs $\mathbf{u}_i$. The system can be under disturbances that are applied to either positions $\mathbf{w}_{p,i}$ and velocities $\mathbf{w}_{v,i}$. The nominal dynamics of the system are considered disturbance-free, i.e.,

$$\dot{\boldsymbol{\xi}}_{p,i} = \boldsymbol{\xi}_{v,i} \tag{9.25}$$
$$\dot{\boldsymbol{\xi}}_{v,i} = \boldsymbol{\nu}_i, \tag{9.26}$$

with $\boldsymbol{\xi}_{p,i}$ the nominal position of the agent, unaffected by the disturbances, $\boldsymbol{\xi}_{v,i}$ the nominal velocities and $\boldsymbol{\nu}_i$ the nominal inputs. The whole states vector is $\boldsymbol{\xi}_i = [\boldsymbol{\xi}_{p,i}, \boldsymbol{\xi}_{v,i}] \in \mathbb{R}^{6\times 1}$.

The formation and a main reference are considered known a priori, leading to the direct computation of references $\boldsymbol{\xi}_{e,i}$ for each agent in the swarm.

In each partition, a distributed Tube MPC is used to control the agents. The first layer of the control methods for each agent, the nominal con-

troller, is defined as a model predictive control:

$$\begin{aligned}\min_{\upsilon \boldsymbol{\nu}_i} \quad & \sum_{k=1}^{N_h-1} [(\boldsymbol{\xi}_i(k) - \boldsymbol{\xi}_{e,i}(k))^T(i)\mathbf{Q}_{i,n}(\boldsymbol{\xi}_i(k) - \boldsymbol{\xi}_{e,i}(k)) \\ & \qquad + (\boldsymbol{\nu}_i(k) - \mathbf{u}_e)^T\mathbf{R}_{i,n}(\boldsymbol{\nu}_i(k) - \mathbf{u}_e) \\ & + \sum_{j=1}^{m_l}(\boldsymbol{\xi}_i(k) - \boldsymbol{\xi}_{e,j}(k) + \mathbf{d}_{ij})^T\mathbf{P}_{ij,n}(\boldsymbol{\xi}_i(k) - \boldsymbol{\xi}_{e,j}(k) + \mathbf{d}_{ij})] \\ & \qquad + (\boldsymbol{\xi}_i(N) - \boldsymbol{\xi}_{e,i}(N))^T\mathbf{H}_{i,n}(\boldsymbol{\xi}_i(N) - \boldsymbol{\xi}_{e,i}(N)) \qquad (9.27a)\end{aligned}$$

$$\boldsymbol{\xi}_i(k+1) = \mathbf{f}_i(\boldsymbol{\xi}_i(k), \boldsymbol{\nu}_i(k)), \quad \boldsymbol{\xi}(0) = \boldsymbol{\xi}_0 \qquad (9.27b)$$

$$\boldsymbol{\xi}_i(k) \in \Xi_i \qquad (9.27c)$$

$$\boldsymbol{\nu}_i(k) \in \mathcal{V}_i \qquad (9.27d)$$

$$\mathbf{g}_{ij}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j) \leq 0. \qquad (9.27e)$$

The parameter N_h defines the prediction horizon, m_l defines the number of agents in the actual partitions, and matrices $\mathbf{Q}_{i,n}$,$\mathbf{R}_{i,n}$,$\mathbf{P}_{ij,n}$,$\mathbf{H}_{i,n}$ are the weight matrices of the optimization problem. Besides $\boldsymbol{\xi}_0$ is the initial conditions of the problem, $\boldsymbol{\xi}_{e,i}$ and $\boldsymbol{\xi}_{e,j}$ are the tracking references of both agent i and j and $\mathbf{u}_e$ the input setpoint. Eventually, d_{ij} is the distance to be maintained between two agents in each dimensions, $\mathcal{Z}_i$ and $\mathcal{V}_i$ are constrained sets applied to the nominal states and inputs, respectively.

In this cost function, the first term penalizes the deviation of the nominal state $\boldsymbol{\xi}_i(k)$ from the reference trajectory $\boldsymbol{\xi}_{i,j}$. It aims to keep the UAV as close as possible to its desired path. The second term penalizes the deviation of the nominal input $\boldsymbol{\nu}_i(k)$ from a setpoint $\mathbf{u}_e$. It encourages the controller to use inputs close to a desired equilibrium or operating point. The third term considers the relative positions of UAV i with respect to all other UAVs j in the same partition (total of m_l UAVs). It helps maintain desired formations by penalizing deviations from planned relative positions.

The cost of those terms can be tuned through the weight matrices, which create behavior such as, for instance, a leader-follower strategy, where the leader receives more weight in the scheme. Eventually, a terminal cost is used, penalizing the state error at the end of the prediction horizon, which helps ensure stability and performance over a longer time scale.

Constraint (9.27b) enforces the nominal system dynamics,i.e., it ensures that the predicted states follow the system model $\mathbf{f}_i$, starting from

the initial condition $\boldsymbol{\xi}_0$. The second constraint (9.27c) and the third constraint (9.27d) ensure that the nominal states and inputs remain within the feasible set. This set might represent the physical limitations of the UAV, desired operating regions, and obstacle avoidance mechanism. A more detailed approach for computing those sets is shown in [158, 159]. Eventually, a collision avoidance constraint is ensured in the nominal controller and embedded in the convex set Ξ_i.

Once the problem is solved, an optimal trajectory for both states $\boldsymbol{\xi}_i^*$ and inputs $\boldsymbol{\nu}_i^*$ is available and can be fed to a second layer, the ancillary controller. We define it as a model predictive controller:

$$\begin{aligned}
\min_{u_i} \quad & \sum_{k=1}^{N-1} [(\mathbf{x}_i(k) - \boldsymbol{\xi}_i^*(k))^T(i)\mathbf{Q}_{i,a}(\mathbf{x}_i(k) - \boldsymbol{\xi}_i^*(k)) \\
& \qquad + (\mathbf{u}_i(k) - \boldsymbol{\nu}_i^*(k))^T\mathbf{R}_{i,a}(\mathbf{u}_i(k) - \boldsymbol{\nu}_i^*(k)) \\
& + \sum_{j=1}^{m_l} (\mathbf{x}_i(k) - \boldsymbol{\xi}_j^*(k) + \mathbf{d}_{ij})^T\mathbf{P}_{i,j,a}(\mathbf{x}_i(k) - \boldsymbol{\xi}_j^*(k) + \mathbf{d}_{ij}] \\
& \qquad + (\mathbf{x}_i(N) - \boldsymbol{\xi}_i^*(N))^T(N)\mathbf{H}_{i,a}(\mathbf{x}_i(N) - \boldsymbol{\xi}_i^*(N)) \qquad (9.28a)
\end{aligned}$$

$$\mathbf{x}_i(k+1) = f_i(\mathbf{x}_i(k), \mathbf{u}_i(k), 0), \quad x(0) = \mathbf{x}_0, \qquad (9.28b)$$

m_l defines the number of agents in the actual partition and matrices $\mathbf{Q}_{i,a}$, $\mathbf{R}_{i,a}$, $\mathbf{P}_{ij,a}$, $\mathbf{H}_{i,a}$ are the weight matrices of the optimization problem. Moreover, c_0 is the initial condition of the problem.

The feasibility of the Tube-MPC scheme can be analyzed: Initial feasibility can be assumed if the constraints are satisfied at the start of the optimization process and a feasible solution exists for the first time step.

The terminal cost, $\mathbf{H}_{i,n}$, is chosen by solving an unconstrained Linear Quadratic Regulator (LQR) for the nominal system from step N to infinity. This approach ensures that the terminal cost reflects the optimal behavior of the system beyond the prediction horizon, driving the system toward a stable state [160].

The terminal set $\mathcal{Z}_N$ is computed through a reachability analysis, which involves calculating the set of all possible trajectories that can lead the system into the desired terminal region. By ensuring that the predicted state ends in this terminal set, we guarantee that the system will remain feasible for all subsequent time steps.

Finally, the ancillary controller operates without explicit constraints. As it aims to correct deviations from the nominal trajectory provided by the first MPC layer, it will always find a feasible solution, as no constraints are imposed. This inherent flexibility ensures that the ancillary controller will not violate any state or input bounds, and thus, the overall Tube-MPC scheme remains feasible at all times.

9.4.2 Feasibility

In a multi-agent system, individual feasibility does not guarantee collective feasibility because agents' trajectories are interdependent; a change in one trajectory may compromise another agent's constraints. In a distributed Tube-MPC framework, the use of tubes as buffers helps accommodate slight deviations and enhances collision avoidance robustness.

Feasibility is maintained if all agents begin within their respective feasible sets and the overall configuration satisfies the collision avoidance constraints. Under these conditions, the framework can sustain feasibility over time. However, if agents are initialized too closely or uncertainties exceed expected bounds, the tubes may not ensure sufficient separation, risking collisions. Moreover, in highly dynamic environments, communication delays or errors in trajectory sharing may further undermine constraint satisfaction.

The study of such feasibility and its guarantee remains one of the challenges that should be further investigated.

9.4.3 Application to one partition

To evaluate the effectiveness of our proposed distributed Tube-MPC approach, we conducted a series of simulations involving a group of three agents located in a single partition. The simulations explore various scenarios, each highlighting different aspects of the control strategy. The results are presented in five cases where 3 agents start in line and must form a triangular formation.

In the first scenario, Fig. 9.6, we observe the behavior of agents without consideration for other agents in the partition. The results show that each agent moved directly towards its reference point, following a straight-line trajectory. This case serves as a baseline, demonstrating the default behavior of agents when formation control is not weighted in the problem.

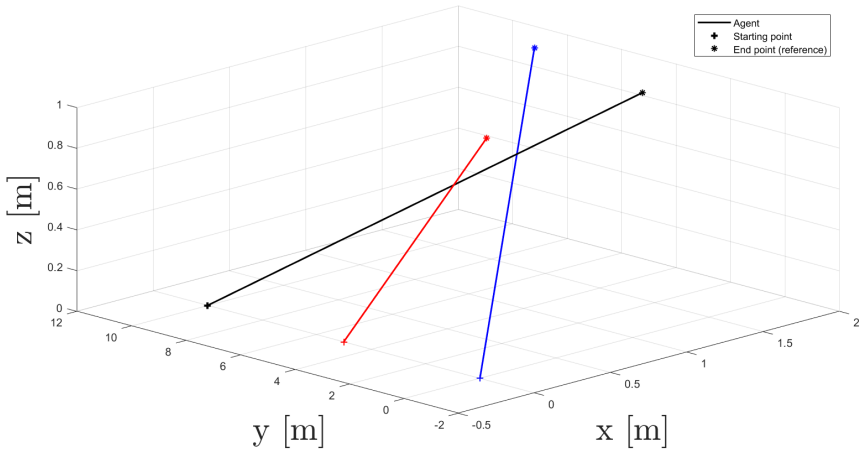


Figure 9.6: First case: matrices P are null. The three agents rush to their reference point.

The second scenario, Fig. 9.7 introduces a slight weighting for formation control. The results indicate that the agents no longer follow the straight-line trajectories observed in case 1. This adaptation demonstrates the initial impact of formation control as agents begin to consider their positions relative to other swarm members while still prioritizing their individual reference points.

In the third scenario, Fig 9.8, the weighting for formation control is significantly increased. The results show a marked change in agent behavior compared to the previous cases. Agents quickly form the desired pattern before collectively moving towards their reference points. This behavior highlights the effectiveness of our approach in prioritizing swarm cohesion when formation parameters are given higher importance.

The fourth scenario, Fig. 9.9, explores leader-follower dynamics by assigning different weights to different agents. One agent is given a high weight, effectively designating it as the leader, while others are assigned lower weights, making them followers. The results demonstrate this hierarchy: The leader agent moves directly towards its reference point, similar to the behavior observed in Case 1. Follower agents prioritize maintaining formation with the leader, adjusting their trajectories accordingly. This case showcases the flexibility of our approach in implementing various swarm behaviors through weight adjustments.

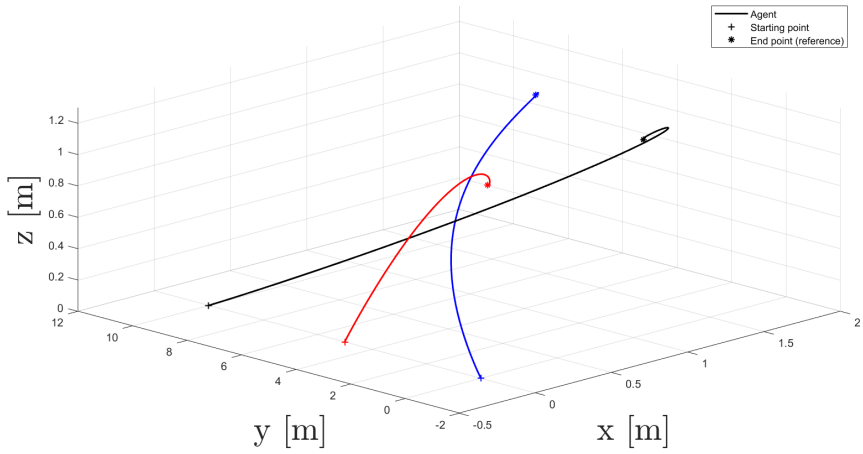


Figure 9.7: Second case: the agents take account of the relative motion in their path to the reference endpoints.

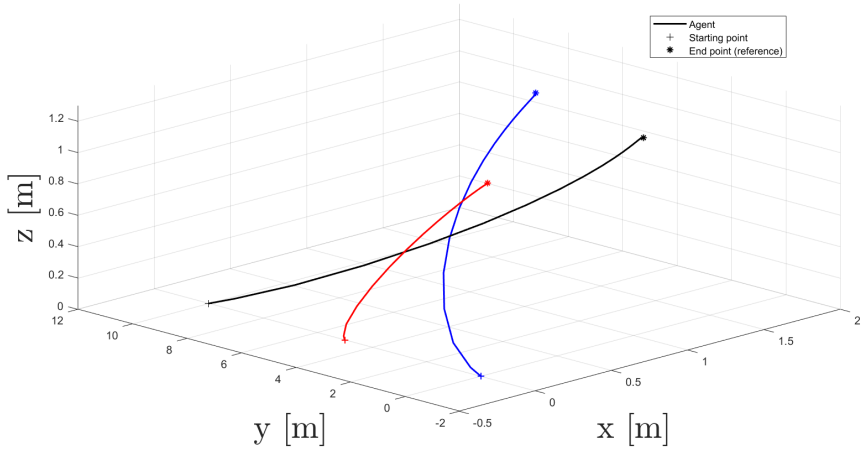


Figure 9.8: Third case: the three agents create a pattern before moving towards the references.

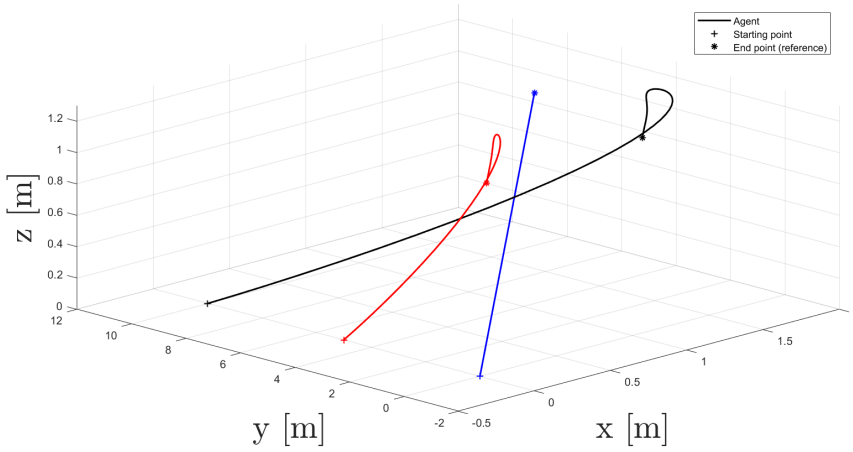


Figure 9.9: Fourth case: an agent is a leader (blue) and reaches its reference without looking at the formation, while followers (red and black) try to manage the formation before going towards their reference endpoints.

In the final scenario, Fig. 9.10, the robustness of our control strategy is tested by introducing velocity uncertainties of 0.2 m/s in every direction while maintaining equal, low formation weights for all agents. The results demonstrate that the agents successfully maintain their formation despite the introduced uncertainties. The overall swarm behavior remains stable, with agents effectively compensating for the velocity disturbances. This case validates the robustness of our distributed Tube-MPC approach in handling real-world uncertainties while maintaining desired swarm behavior.

Our results demonstrate the versatility and effectiveness of the proposed distributed Tube-MPC approach for UAV swarm control. The method allows for flexible control of swarm behavior by adjusting formation weights. It successfully implements various swarm configurations, from independent agent behavior to tight formations. The approach can create leader-follower dynamics within the swarm by assigning different weights to agents. Most importantly, the control strategy shows robust performance despite uncertainties, maintaining stable swarm behavior under realistic conditions.

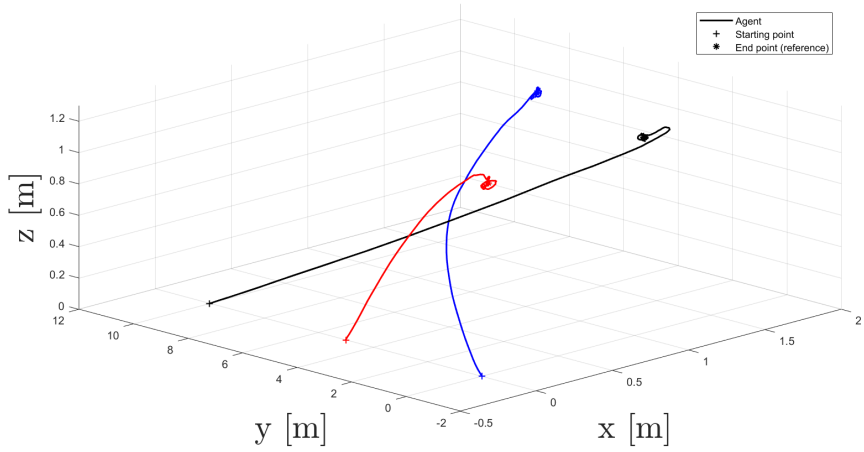


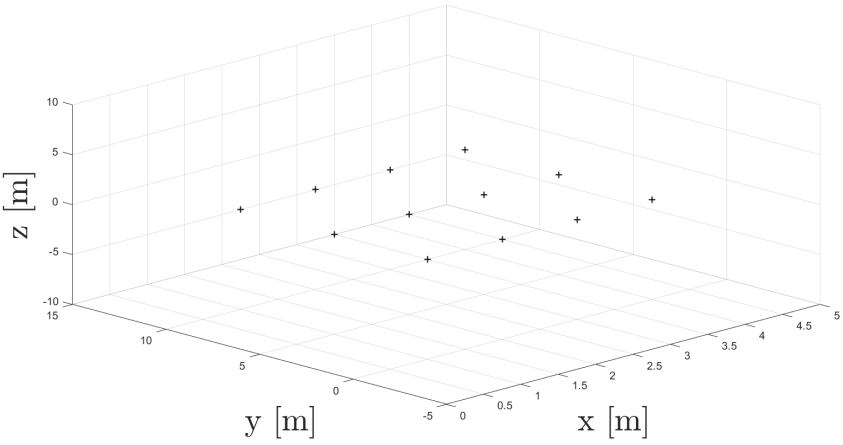
Figure 9.10: Fith case: velocity uncertainties of 0.2 m/s are introduced.

9.5 Simulation results

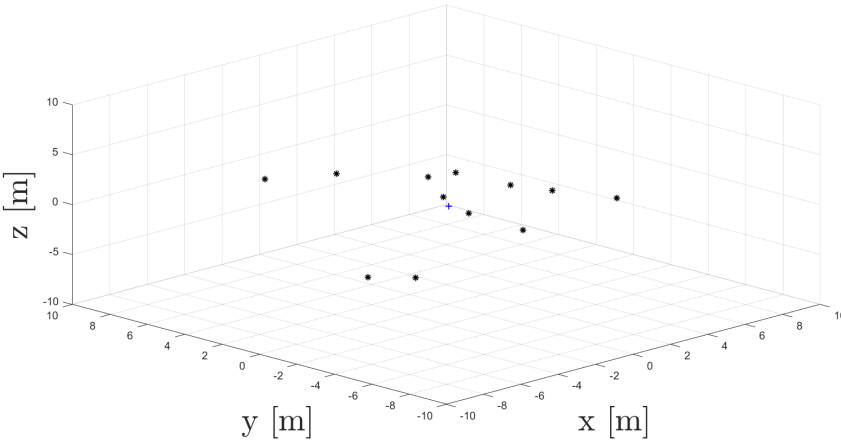
Consider a swarm of twelve UAVs that have just taken off from the ground. Initially, they were aligned in a grid formation. They have to obey a specific formation given by a higher architecture level, Fig. 9.11, and to move towards the y-axis direction to clear the take-off platform.

To achieve this objective, the partition and network communication is established for the desired reference, Fig. 9.12; then, the distributed Tube-MPC algorithm takes over to guide the swarm into the desired formation, Fig. 9.13.

Notice that the partitions are assigned to the swarm at the initial time. Specifically, from the grid-like formation, we assign a partition number to each UAV in an ordered manner. Then, the swarm moves to reach the formation and advances towards the y-axis direction. The collision avoidance mechanism is also employed, and we can see how each trajectory avoids colliding with others. The results demonstrate the ability to handle larger swarms of UAVs effectively.



(a) Starting position of the swarm after takeoff, in a grid-like way.



(b) Reference formation to reach.

Figure 9.11: Starting and reference formation

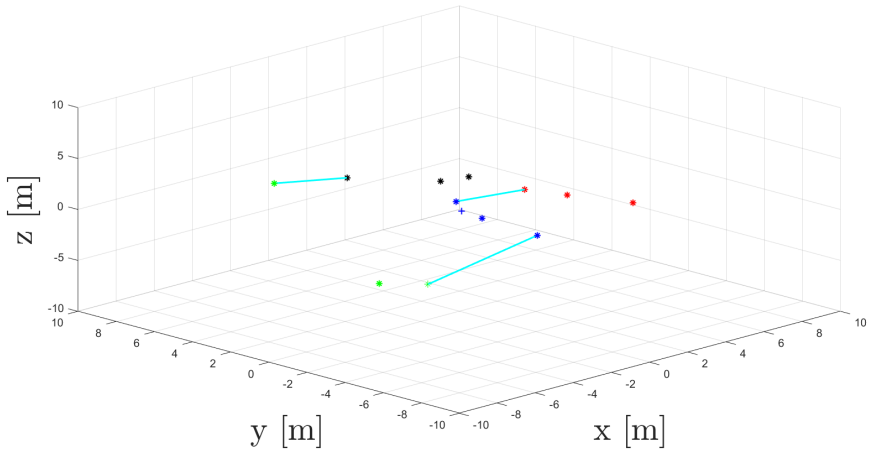


Figure 9.12: Resulting partitioning for the 3D formation provided. Each color represents one partition. The cyan lines represent the communication link between the partitions.

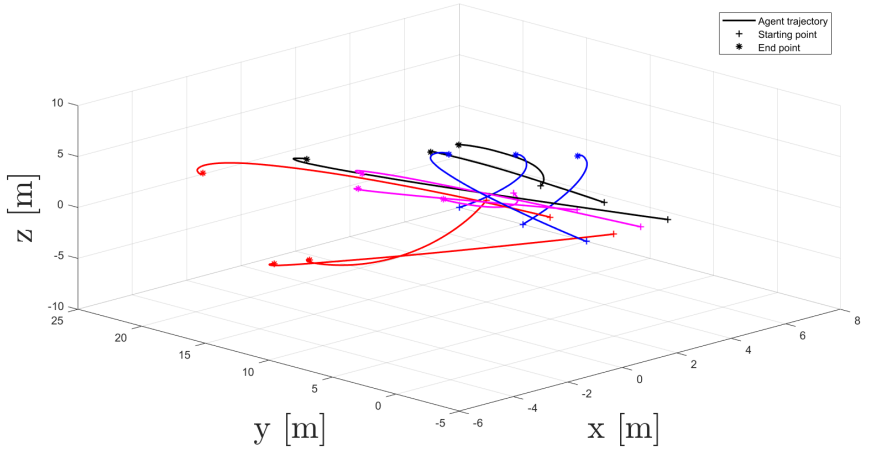


Figure 9.13: Trajectories of the UAVs swarm under the specified starting position and the desired formation. The formation is set to move 15 m along the y-axis.

9.6 Conclusion

This chapter presents a distributed control strategy for managing large UAV swarms by partitioning the collective into smaller, more manageable subgroups. The core contribution is a framework that leverages a graph-theory-based partitioning algorithm, alongside dedicated communication protocols to ensure robust control. By structuring the swarm into these independent yet coordinated units, we transform an otherwise intractable large-scale control problem into a set of localized, solvable ones. Simulation studies demonstrate the effectiveness of this approach in maintaining formation control. The key outcome is the ability to impose a structured optimization problem to redefine a large swarm into a practical and manageable groups.

Future work will focus on extending this partitioning concept to handle more complex, multi-layered command scenarios. The logical next step is to validate the proposed subgroup management and control strategies through real-life hardware experiments.

Chapter 10

Conclusion and Perspectives

This thesis has navigated the complex domain of autonomous UAV control, from low-level stabilization to high-level multi-agent planning. We have argued that the integration set-based methods, particularly zonotopes, into a multi-layered control architecture is a critical step toward achieving robust autonomy. This concluding chapter summarizes the core contributions, provides a critical reflection on the limitations of this work, and outlines promising directions for future research.

10.1 Contributions

The primary contribution of this thesis is a comprehensive, multi-layered framework designed to guarantee robust UAV operation in uncertain environments. At the foundational level, we developed a novel feedback linearization strategy that simplifies the control problem without augmenting the system's state, thereby reducing computational overhead.

Building on this, we successfully integrated zonotopic reachability analysis to provide formal guarantees. Validated against Monte Carlo simulations, this approach proved to be significantly more computationally efficient. We further enhanced robustness through a zonotopic reference governor for adaptive constraint management and a Tube-MPC implementation using polynomial zonotope* to mitigate residual nonlinearities. For high-level autonomy, this culminated in a path planner capable of determining optimal, collision-free trajectories in cluttered environments. Finally, the framework was extended to multi-agent systems, demonstrating

the feasibility of decentralized, coordinated swarm behavior through partitioned communication and optimization-based control.

The table 10.1 summarizes the various simulations and experiments conducted throughout this work. It highlights the diverse platforms used and the evolution of the control methodologies applied. From validating classic controllers like PID and INDI, the work progressed to implementing more advanced solutions, including Tube-MPC and RRT*, to ensure system robustness, safety, and autonomy. These experiments and simulations represent the core contributions of this project.

10.2 Limitations

While the results presented are promising, a critical perspective is essential to understand the boundaries of the proposed methods and the challenges encountered during implementation.

A primary challenge of the zonotopic framework is the inherent trade-off between representation accuracy and computational complexity. While zonotopes are less conservative than many other set-based methods, their conservatism still restricts reachable sets of highly nonlinear systems or non-convex safe regions, potentially preventing the drone from entering a valid, safe state. The use of polynomial zonotopes mitigates this but at a much higher computational cost. Furthermore, scalability remains a concern; the complexity of zonotope operations for long prediction horizons or in high-dimensional state spaces can become a bottleneck for real-time implementation on computationally constrained hardware. The number of generators can grow exponentially in some scenarios if not carefully managed through order-reduction techniques, which themselves introduce further approximation errors. A fundamental computational challenge also arises from the generator-based definition of zonotopes: unlike simpler sets, there is no straightforward analytical solution for the zonotope membership problem, and obtaining an explicit description of its boundary is computationally prohibitive. This means operations requiring such a description, like precise collision checking against arbitrary non-zonotopic obstacles or verifying containment within a general safe region, can be very slow and often rely on simpler, more conservative outer approximations.

The two-platform experimental strategy (Parrot Mambo and custom

hexacopter) was a deliberate choice to validate different layers of the control stack, but it also highlighted key implementation gaps. We successfully validated the feedback linearization and INDI control on the Parrot Mambo due to its accessible firmware. However, implementing this custom low-level controller on the Pixhawk-based hexacopter proved infeasible due to the deeply integrated nature of the PX4 flight stack. While the hexacopter served as an excellent demonstrator for high-level algorithms like Tube-MPC and the RRT planner, which command an existing autopilot, this approach meant our high-level controller was blind to the true state of the internal PID loops of the autopilot. Unmodeled delays or saturation within the PX4 stack could, in some cases, degrade performance in ways our model did not predict, representing a gap between simulation and reality.

10.3 Future Work

The limitations and insights gained from this work open up several exciting avenues for future research.

A significant opportunity lies in bridging the gap between our model-based methods and data-driven machine learning techniques. While this thesis assumed a simplified model, future work could employ Gaussian Processes or Physics-Informed Neural Networks to learn complex, unmodeled aerodynamic effects from flight data. The key challenge is to then bound the prediction error of these learned models within a zonotope, creating a "safe learning" framework that combines the accuracy of ML with the formal guarantees of reachability analysis.

Our work on multi-agent systems laid a foundation for decentralized control, and the next steps involve tackling more complex and realistic swarm scenarios. Future research should investigate how zonotopic predictions can be used to maintain safety during communication dropouts. If an agent loses contact, its neighbors could propagate its last known state and control input forward in time using a reachable set, ensuring collision avoidance with a guaranteed "ghost" tube until contact is re-established.

The most forward-looking research direction involves integrating our robust, low-level autonomy with emerging large-scale AI, such as Vision-Language Models (VLMs). Currently, a human operator defines mission goals. In the future, a user could issue a complex, natural-language com-

mand like, "Safely inspect the facade of that building for cracks, avoiding the trees." A VLM could parse this command, identify the building and obstacles from visual input, and translate the abstract goal into a series of waypoints, safe zones, and constraints that our zonotope-based planner can then execute with formal safety guarantees. This would represent a true leap in cognitive autonomy for robotics.

10.4 Concluding Remarks

The rapid proliferation of autonomous drones from research labs into our daily lives marks a pivotal moment in robotics. Their potential for positive impact, in logistics, safety, and data gathering, is immense. However, this potential can only be fully realized if these systems are trusted to operate safely and reliably. The core of this trust lies not just in performance, but in predictability and guarantees.

Table 10.1: Summary of Simulations and Experiments

Platform & Validation	**Control Method**	**Additional Features**
Simulation & Experimental (Parrot Mambo)	Cascade PID Controller	N/A
Simulation & Experimental (Parrot Mambo)	INDI Controller	N/A
Simulation (Parrot Mambo)	Classic Feedback Linearization	N/A
Simulation (Parrot Mambo)	Classic Feedback Linearization	Reference Governor
Simulation (Parrot Mambo)	INDI Controller	Reference Governor
Simulation (DIY Multicopter)	INDI + TubeMPC (Both MPC)	Constrained Polynomial Zonotopes (safe set)
Simulation (DIY Multicopter)	Cascade PID (Px4) + TubeMPC (State Feedback + MPC)	Zonotopes (safe set)
Experimental (DIY Multicopter)	Cascade PID (Px4)	N/A
Experimental (DIY Multicopter)	Cascade PID (Px4) + TubeMPC (State Feedback + MPC)	Zonotopes (safe set)
Simulation (Multicopter)	INDI + TubeMPC (Both MPC)	Constrained Polynomial Zonotopes (safe set) + Real-time RRT*
Simulation (Multicopter)	Cascade PID (Px4) + TubeMPC (State Feedback + MPC)	Zonotopes (safe set) + Real-time RRT*
Experimental (Crazyflie)	Base Controller	Real-time RRT* Algorithm
Simulation (Distributed System)	Partitioning	Distributed TubeMPC

Bibliography

[1] D. Floreano and R. J. Wood. Science, technology and the future of small autonomous drones. *nature*, 521(7553):460–466, 2015.

[2] N. Aliane. A survey of open-source uav autopilots. *Electronics*, 13(23):4785, 2024.

[3] R. Mogili and B. Deepak. Review on application of drone systems in precision agriculture. *Procedia Computer Science*, 133:502–509, 2018. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

[4] F. Elghaish, S. Matarneh, S. Talebi, M. Kagioglou, M. Hosseini, and S. Abrishami. Toward digitalization in the construction industry with immersive and drones technologies: a critical literature review. *Smart and Sustainable Built Environment*, 10(3):345–363, Jan 2021.

[5] M. Moshref-Javadi and M. Winkenbach. Applications and research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications*, 177:114854, 2021.

[6] A. Chovancová, T. Fico, P. Hubinský, and F. Duchoň. Comparison of various quaternion-based control methods applied to quadrotor with disturbance observer and position estimator. *Robotics and Autonomous Systems*, 79:87–98, 2016.

[7] H. Abdulrahman and M. Tariq. Quadrotor control via robust generalized dynamic inversion and adaptive non-singular terminal sliding mode. *Asian Journal of Control*, 21(3):1237–1249, 2019.

[8] R. Warier, K. Sanyal, M. Dhullipalla, and S. Viswanathan. Trajectory tracking control for underactuated thrust-propelled aerial vehi-

cles. *IFAC-PapersOnLine*, 51(13):555–560, 2018. 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018.

[9] T. Baca, D. Hertl, G. Loianno, M. Saska, and V. Kumar. Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6753–6760, 2018.

[10] T. Nascimento and M. Saska. Position and attitude control of multi-rotor aerial vehicles: A survey. *Annual Reviews in Control*, 48:129–146, 2019.

[11] B. Rego and G. Raffo. Suspended load path tracking control using a tilt-rotor uav based on zonotopic state estimation. *Journal of the Franklin Institute*, 356(4):1695–1729, 2019.

[12] H. Pham, D. Ichalal, and S. Mammar. Lpv and nonlinear-based control of an autonomous quadcopter under variations of mass and moment of inertia. *IFAC-PapersOnLine*, 52(28):176–183, 2019. 3rd IFAC Workshop on Linear Parameter Varying Systems LPVS 2019.

[13] S. Ullah, A. Mehmood, Q. Khan, S. Rehman, and J. Iqbal. Robust integral sliding mode control design for stability enhancement of under-actuated quadcopter. *International Journal of Control, Automation and Systems*, 18(7):1671–1678, Jul 2020.

[14] C. Nicol, C.J.B. Macnab, and A. Ramirez-Serrano. Robust adaptive control of a quadrotor helicopter. *Mechatronics*, 21(6):927–938, 2011.

[15] N. Michel, S. Bertrand, S. Olaru, G. Valmorbida, and D. Dumur. Design and flight experiments of a tube-based model predictive controller for the ar.drone 2.0 quadrotor. *IFAC-PapersOnLine*, 52(22):112–117, 2019. 1st IFAC Workshop on Robot Control WROCO 2019.

[16] M. Althoff, F. Goran, and A. Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):369–395, 2021.

[17] M. Althoff. *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, Technische Universität München, 2010.

[18] M. Chen, Q. Hu, C. Mackin, J. F. Fisac, and C. J. Tomlin. Safe platooning of unmanned aerial vehicles via reachability. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 4695–4701, 2015.

[19] T. Sunaga et al. Theory of an interval algebra and its application to numerical analysis. *Japan Journal of Industrial and Applied Mathematics*, 26(2):125, 2009.

[20] D. G. Maksarov and J. P. Norton. State bounding with ellipsoidal set description of the uncertainty. *International Journal of Control*, 65(5):847–866, 1996.

[21] P. McMullen. G. m. ziegler lectures on polytopes. *Proceedings of the Edinburgh Mathematical Society*, 39(1):189–190, 1996.

[22] B. Grünbaum. *Polytopes*, pages 35–60. Springer New York, New York, NY, 2003.

[23] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69:126–136, 2016.

[24] N. Kochdumper and M. Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*, 66(9):4043–4058, 2021.

[25] N. Kochdumper and M. Althoff. Constrained polynomial zonotopes. *Acta Informatica*, 60(3):279–316, May 2023.

[26] B. Schürmann and M. Althoff. Optimizing sets of solutions for controlling constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 66(3):981–994, 2021.

[27] Y. Cao, W. Tan, and Z. Wu. Aircraft icing: An ongoing threat to aviation safety. *Aerospace Science and Technology*, 75:353–385, 2018.

[28] C. V. Oster, J. S. Strong, and C. K. Zorn. Analyzing aviation safety: Problems, challenges, opportunities. *Research in Transportation Economics*, 43(1):148–164, 2013. The Economics of Transportation Safety.

[29] D. D. Boyd. A review of general aviation safety (1984–2017). *Aerospace Medicine and Human Performance*, 88(7):657–664, July 2017.

[30] Y. Zhang, C. de Visser, and Q. P. Chu. Database building and interpolation for an online safe flight envelope prediction system. *Journal of Guidance, Control, and Dynamics*, 42:1–9, 01 2019.

[31] T. Lombaerts, J. Kaneshige, S. Schuet, B. L. Aponso, K. H. Shish, and G. Hardy. Dynamic inversion based full envelope flight control for an evtol vehicle using a unified framework. In *AIAA Scitech 2020 Forum*, page 1619, 2020.

[32] S. Sun and C. C. de Visser. Quadrotor safe flight envelope prediction in the high-speed regime: A monte-carlo approach. In *AIAA Scitech 2019 Forum*, page 0948, 2019.

[33] E. Eyang, C. Combastel, and A. Zolghadri. Determination of reachable flight regions: a zonotopic approach. *IFAC-PapersOnLine*, 50(1):4014–4020, 2017. 20th IFAC World Congress.

[34] H. G. Harno and Y. Kim. Flight envelope estimation for helicopters under icing conditions via the zonotopic reachability analysis. *Aerospace Science and Technology*, 102:105859, 2020.

[35] P. Pounds, R. Mahony, P. Hynes, and J. M. Roberts. Design of a four-rotor aerial robot. In *The Australian Conference on Robotics and Automation (ACRA 2002)*, pages 145–150, 2002.

[36] R. W. Prouty. *Helicopter Performance, Stability, and Control*. PWS Engineering, Boston, 1986.

[37] A. T. Conlisk. Modern helicopter rotor aerodynamics. *Progress in Aerospace Sciences*, 37(5):419–476, 2001.

[38] N. Guenard, T. Hamel, and V. Moreau. Dynamic modeling and intuitive control strategy for an "x4-flyer". In *2005 International Conference on Control and Automation*, volume 1, pages 141–146 Vol. 1, 2005.

[39] W. B. Heard. *Rigid Body Mechanics: Mathematics, Physics and Applications*. Physics textbook. Wiley, 2008.

[40] P. Pounds, R. Mahony, and P. Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699, 2010.

[41] J.P. Bristeau, P. Martin, E. Salaün, and N. Petit. The role of propeller aerodynamics in the model of a quadrotor uav. In *2009 European control conference (ECC)*, pages 683–688. IEEE, 2009.

[42] K. J. Åström and R. Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2021.

[43] K. Ogata. Modern control engineering. 2020.

[44] G. F. Franklin, J. D. Powell, A. Emami-Naeini, et al. *Feedback control of dynamic systems*, volume 10. Pearson Upper Saddle River, NJ, 2010.

[45] D. Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[46] G. Welch, G. Bishop, et al. An introduction to the kalman filter. 1995.

[47] H. K. Khalil and J. W. Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[48] R. C. Dorf and R. H. Bishop. Modern control systems twelfth edition. *Pearson*, pages 859–868, 2011.

[49] B. C. Kuo. *Automatic control systems*. Prentice Hall PTR, 1987.

[50] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell. *Feedback control of dynamic systems*, volume 4. Prentice hall Upper Saddle River, 2002.

[51] F. Riether. *Agile quadrotor maneuvering using tensor-decomposition-based globally optimal control and onboard visual-inertial estimation*. PhD thesis, Massachusetts Institute of Technology, 2016.

[52] M. R. Kaplan, A. Eraslan, A. Beke, and T. Kumbasar. Altitude and position control of parrot mambo minidrone with pid and fuzzy pid controllers. In *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*, pages 785–789. IEEE, 2019.

[53] A. Noordin, M. A. M. Basri, and Z. Mohamed. Simulation and experimental study on pid control of a quadrotor mav with perturbation. *Bulletin of Electrical Engineering and Informatics*, 9(5):1811–1818, 2020.

[54] I. Trenev, A. Tkachenko, and A. Kustov. Movement stabilization of the parrot mambo quadcopter along a given trajectory based on pid controllers. *IFAC-PapersOnLine*, 54(13):227–232, 2021.

[55] T. V. Glazkov and A. E. Golubev. Using simulink support package for parrot minidrones in nonlinear control education. In *AIP Conference Proceedings*, volume 2195, page 020007. AIP Publishing LLC, 2019.

[56] A. Golubev and T. Glazkov. Nonlinear quadrotor control based on simulink support package for parrot minidrones. In *MMSC*, pages 113–127, 2020.

[57] A. Talaeizadeh, E. Najafi, H. N. Pishkenari, and A. Alasty. Deployment of model-based design approach for a mini-quadcopter. In *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, pages 291–296. IEEE, 2019.

[58] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar. Pid, lqr and lqr-pid on a quadcopter platform. In *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 1–6, 2013.

[59] J. Qiao, Z. Liu, and Y. Zhang. Gain scheduling pid control of the quad-rotor helicopter. In *2017 IEEE International Conference on Unmanned Systems (ICUS)*, pages 1594–1601, 2017.

[60] V. K. Tripathi, L. Behera, and N. Verma. Design of sliding mode and backstepping controllers for a quadcopter. In *2015 39th National Systems Conference (NSC)*, pages 1–6, 2015.

[61] H. Voos. Nonlinear control of a quadrotor micro-uav using feedback-linearization. In *2009 IEEE International Conference on Mechatronics*, pages 1–6, 2009.

[62] H. Merabti, I. Bouchachi, and K. Belarbi. Nonlinear model predictive control of quadcopter. In *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 208–211, 2015.

[63] E. J. J. Smeur, Q. Chu, and G. C. H. E. de Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, March 2016.

[64] R. van 't Veld, E. J. Van Kampen, and Q. P. Chu. Stability and robustness analysis and improvements for incremental nonlinear dynamic inversion control. In *2018 AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2018.

[65] J. E. Slotine, W. Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[66] M. A. Lotufo, L. Colangelo, and C. Novara. Control design for uav quadrotors via embedded model control. *IEEE Transactions on Control Systems Technology*, 28(5):1741–1756, 2020.

[67] S. Sieberling, Q. Chu, and J. A. Mulder. Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction. *Journal of guidance, control, and dynamics*, 33(6):1732–1742, 2010.

[68] G. Farin. *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.

[69] S. R. Jacobson. Aircraft loss of control causal factors and mitigation challenges. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pages 1–18 (2010–8007), Toronto, Canada, 2010.

[70] C. M. Belcastro, J. V. Foster, G. H. Shah, I. M. Gregory, D. E. Cox, D. A. Crider, L. Groff, R. L. Newman, and D. H. Klyde. Aircraft loss of control problem analysis and research toward a holistic solution. *Journal of Guidance, Control, and Dynamics*, 40(4):733–775, 2017.

[71] M. D. White, G. D. Padfield, L. Lu, and S. Advani. Rotorcraft loss of control in-flight – The need for research to support increased fidelity in flight training devices, including analogies with upset recovery for fixed-wing aircraft. In *Proceedings of the 44th European Rotorcraft Forum*, pages 1–2, Delft, Netherlands, 2018.

[72] R. Pandita, A. Chakraborty, P. Seiler, and G. Balas. Reachability and region of attraction analysis applied to gtm dynamic flight envelope assessment. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pages 1–21 (2009–6258), Chicago, IL, USA, 2009.

[73] N. Grande, S. Tierney, J. F. Horn, and J. W. Langelaan. Safe autorotation through wind shear via backward reachable sets. *Journal of the American Helicopter Society*, 61(2):1–11 (022006), 2016.

[74] H. Pfifer, R. Venkataraman, and P. Seiler. Quantifying loss-of-control envelopes via robust tracking analysis. *Journal of Guidance, Control, and Dynamics*, 40(4):1042–1050, 2017.

[75] S. Schuet, T. Lombaerts, D. Acosta, J. Kaneshige, K. Wheeler, and K. Shish. Autonomous flight envelope estimation for loss-of-control prevention. *Journal of Guidance, Control, and Dynamics*, 40(4):847–862, 2017.

[76] D. C. Hartman. Identification of hazardous flight conditions to establish a safe flight envelope for autonomous multirotor aircraft. In *Proceedings of the AIAA Scitech 2019 Forum*, pages 1–15 (2019–1292), San Diego, CA, USA, 2019.

[77] S. Sun and C. C. de Visser. Quadrotor safe flight envelope prediction in the high-speed regime: A Monte-Carlo approach. In *Proceedings of the AIAA Scitech 2019 Forum*, pages 1–12 (2019–0948), 2019.

[78] M. Yin, Q. P. Chu, Y. Zhang, M. A. Niestroy, and C. C. de Visser. Probabilistic flight envelope estimation with application to unstable

overactuated aircraft. *Journal of Guidance, Control, and Dynamics*, 42(12):2650–2663, 2019.

[79] W. Zheng, Y. Li, L. Qu, and G. Yuan. Dynamic envelope determination based on differential manifold theory. *Journal of Aircraft*, 54(5):2005–2009, 2017.

[80] G. Yuan and Y. Li. Determination of the flight dynamic envelope via stable manifold. *Measurement and Control*, 52(3–4):244–251, 2019.

[81] R. van den Brandt and C. C. de Visser. Safe flight envelope uncertainty quantification using probabilistic reachability analysis. In *Proceedings of the 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, volume 51, pages 628–635, 2018.

[82] Y. Zhang, C. C. de Visser, and Q. P. Chu. Database building and interpolation for an online safe flight envelope prediction system. *Journal of Guidance, Control, and Dynamics*, 42(5):1166–1174, 2019.

[83] Z. Yu, Y. Li, Z. Zhang, W. Xu, and Z. Dong. Online safe flight envelope protection for icing aircraft based on reachability analysis. *International Journal of Aeronautical and Space Sciences*, 21(4):1174–1184, 2020.

[84] J. Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004.

[85] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton–Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005.

[86] T. Lombaerts, S. Schuet, D. Acosta, J. Kaneshige, K. Shish, and L. Martin. Piloted simulator evaluation of safe flight envelope display indicators for loss of control avoidance. *Journal of Guidance, Control, and Dynamics*, 40(4):948–963, 2017.

[87] R. Helsen, E. J. van Kampen, C. C. de Visser, and Q. Chu. Distance-Fields-Over-Grids method for aircraft envelope determination. *Journal of Guidance, Control, and Dynamics*, 39(7):1470–1480, 2016.

[88] A. K. Akametalu, C. J. Tomlin, and M. Chen. Reachability-based forced landing system. *Journal of Guidance, Control, and Dynamics*, 41(12):2529–2542, 2018.

[89] M. Chen and C. J. Tomlin. Hamilton-Jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:333–358, 2018.

[90] A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification. *IEEE Transactions on Automatic Control*, 48(1):64–75, 2003.

[91] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, LNCS 3414, pages 291–305, Berlin, Heidelberg, Germany, 2005. Springer.

[92] A. A. Kurzhanskiy and P. Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 52(1):26–38, 2007.

[93] H. N. V. Pico and D. C. Aliprantis. Reachability analysis of linear dynamic systems with constant, arbitrary, and Lipschitz continuous inputs. *Automatica*, 95:293–305, 2018.

[94] M. Althoff, G. Frehse, and A. Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 2021.

[95] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, pages 291–305, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[96] A. Girard, C. L. Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control*, LNCS 3927, pages 257–271, Berlin, Heidelberg, 2006. Springer.

[97] M. Althoff. Reachability analysis of large linear systems with uncertain inputs in the Krylov subspace. *IEEE Transactions on Automatic Control*, 65(2):477–492, 2020.

[98] S. Sun and C. de Visser. Quadrotor safe flight envelope prediction in the high-speed regime: A monte-carlo approach. In *AIAA Scitech 2019 Forum*, 2019.

[99] G. Delansnay, A. Vande Wouwer, H. G. Harno, and Y. Kim. Zonotopic reachability analysis of multirotor aircraft. In *2021 25th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 576–581. IEEE, 2021.

[100] M. Althoff, N. Kochdumper, and M. Wetzlinger. *CORA 2025 Manual*. Technische Universität München, 87548 Garching, Germany, 2025.

[101] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69:126–136, 2016.

[102] M. Althoff. *Reachability Analysis and Its Application to the Safety Assessment of Autonomous Cars*. PhD thesis, Lehrstuhl für Steuerungs- und Regelungstechnik, Technische Universität Müunchen, Germany, 2010.

[103] A. Girard, C. L. Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control*, pages 257–271. Springer Berlin Heidelberg, 2006.

[104] I. Yavrucuk, J. V. R. Prasad, and S. Unnikrishnan. Envelope protection for autonomous unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 32(1):248–261, 2009.

[105] J. N. Yasin, S. A. S. Mohamed, M. H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila. Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches. *IEEE Access*, 8:105139–105155, 2020.

[106] S. Huang, R. S. H. Teo, and K. K. Tan. Collision avoidance of multi unmanned aerial vehicles: A review. *Annual Reviews in Control*, 48:147–164, 2019.

[107] M. M. Nicotra, R. Naldi, and E. Garone. A robust explicit reference governor for constrained control of unmanned aerial vehicles. In *2016 American Control Conference (ACC)*, pages 6284–6289, 2016.

[108] A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In J. P. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, pages 257–271, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[109] E. Garone, S. Di Cairano, and I. Kolmanovsky. Reference and command governors for systems with constraints: A survey on theory and applications. *Automatica*, 75:306–328, 2017.

[110] M. Althoff, G. Frehse, and A. Girard. Set Propagation Techniques for Reachability Analysis. *Annual Review of Control, Robotics, and Autonomous Systems* , 4(1), May 2021.

[111] V. Raghuraman and J. P. Koeln. Set operations and order reductions for constrained zonotopes. *Automatica*, 139:110204, 2022.

[112] F. Riether. *Agile quadrotor maneuvering using tensor-decomposition-based globally optimal control and onboard visual-inertial estimation*. PhD thesis, Massachusetts Institute of Technology, 2016.

[113] X. Yang and J. K. Scott. A comparison of zonotope order reduction techniques. *Automatica*, 95:378–384, 2018.

[114] R. Steffensen, A. Steinert, Z. Mbikayi, S. Raab, J. Angelov, and F. Holzapfel. Filter and sensor delay synchronization in incremental flight control laws. *Aerospace Systems*, pages 1–20, 2023.

[115] D. Q. Mayne, S. V. Rakovic, R. Findeisen, and F. Allgöwer. Robust constrained model predictive control: A survey. *Automatica*, 44(8):2100–2115, 2008.

[116] B. Kouvaritakis and M. Cannon. Min-max model predictive control: a unifying overview on stability. *European Journal of Control*, 18(2):149–165, 2012.

[117] S. Lucia, R. Paulen, and S. Engell. Multi-stage nonlinear model predictive control with verified robust constraint satisfaction. *IEEE Transactions on Automatic Control*, 62(8):4007–4014, 2017.

[118] T. K. Do and H. Nhan. A comparative study of robust model predictive control methods. *Vietnam Journal of Science and Technology*, 60(1):30–41, 2022.

[119] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi. Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21(11):1341–1353, 2011.

[120] I. Alvarado, D. Limon, T. Alamo, and E. F. Camacho. Output feedback robust tube based mpc for tracking of piece-wise constant references. In *2007 46th IEEE Conference on Decision and Control*, pages 2175–2180, 2007.

[121] F. A. Bayer, M. A. Müller, and F. Allgöwer. Tube-based robust economic model predictive control. *Journal of Process Control*, 24(8):1237–1246, 2014. Economic nonlinear model predictive control.

[122] B. Sakhdari, E. M. Shahrivar, and N. L. Azad. Robust tube-based mpc for automotive adaptive cruise control design. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017.

[123] M. Mirshams and M. Khosrojerdi. Attitude control of an underactuated spacecraft using tube-based mpc approach. *Aerospace Science and Technology*, 48:140–145, 2016.

[124] P. Hang, X. Xia, G. Chen, and X. Chen. Active safety control of automated electric vehicles at driving limits: A tube-based mpc approach. *IEEE Transactions on Transportation Electrification*, 8(1):1338–1349, 2021.

[125] N. Michel, S. Bertrand, S. Olaru, G. Valmorbida, and D. Dumur. Design and flight experiments of a tube-based model predictive controller for the ar.drone 2.0 quadrotor. *IFAC-PapersOnLine*, 52(22):112–117, 2019. 1st IFAC Workshop on Robot Control WROCO 2019.

[126] M. A. Santos, A. Ferramosca, and G. V. Raffo. Tube-based mpc with nonlinear control for load transportation using a uav. *IFAC-PapersOnLine*, 51(25):459–465, 2018. 9th IFAC Symposium on Robust Control Design ROCOND 2018.

[127] L. Dewasme, M. Mäkinen, and V. Chotteau. Multivariable robust tube-based nonlinear model predictive control of mammalian cell cultures. *Computers and Chemical Engineering*, 183:108592, 2024.

[128] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library. Cambridge University Press, 2000.

[129] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[130] F. Fiedler, B. Karg, L. Lüken, D. Brandner, M. Heinlein, F. Brabender, and S. Lucia. do-mpc: Towards fair nonlinear and robust model predictive control. *Control Engineering Practice*, 140:105676, 2023.

[131] J. Gutmann, M. Fukuchi, and M. Fujita. Real-time path planning for humanoid robot navigation. In *IJCAI*, pages 1232–1237, 2005.

[132] C. Tam and R. Bucknall. Path-planning algorithm for ships in close-range encounters. *Journal of marine science and technology*, 15:395–407, 2010.

[133] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[134] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[135] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[136] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ international conference on intelligent robots and systems*, pages 2997–3004. IEEE, 2014.

[137] O. Arslan and P. Tsiotras. Dynamic programming guided exploration for sampling-based motion planning algorithms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4819–4826. IEEE, 2015.

[138] A. Girard. Reachability of uncertain linear systems using zonotopes. In *International workshop on hybrid systems: Computation and control*, pages 291–305. Springer, 2005.

[139] M. Althoff and B. H. Krogh. Zonotopes: A compact representation of reachable sets for linear systems. *Proceedings of the 48th IEEE Conference on Decision and Control (CDC)*, pages 345–351, 2010.

[140] K. Naderi, J. Rajamäki, and P. Hämäläinen. Rt-rrt* a real-time path planning algorithm based on rrt. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 113–118, 2015.

[141] V. Gaßmann and M. Althoff. Scalable zonotope-ellipsoid conversions using the euclidean zonotope norm. In *2020 American Control Conference (ACC)*, pages 4715–4721. IEEE, 2020.

[142] V. Raghuraman and J. P. Koeln. Set operations and order reductions for constrained zonotopes. *Automatica*, 139:110204, 2022.

[143] Y. Zhou, B. Rao, and W. Wang. Uav swarm intelligence: Recent advances and future trends. *IEEE Access*, 8:183856–183878, 2020.

[144] J. D. Boskovic, R. K. Prasanth, and R. K. Mehra. A multi-layer autonomous intelligent control architecture for unmanned aerial vehicles. *J. Aerosp. Comput. Inf. Com munic.*, 1:605–628, 2004.

[145] F. F. Lizzio, E. Capello, and G. Guglieri. A review of consensus-based multi-agent uav applications. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1548–1557, 2021.

[146] R. Olfati-Saber. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[147] J. N. Yasin, S. A. S. Mohamed, M. H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila. Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches. *IEEE Access*, 8:105139–105155, 2020.

[148] A. D. Dang, H. M. La, T. Nguyen, and J. Horn. Formation control for autonomous robots with collision and obstacle avoidance using a rotational and repulsive force–based approach. *International Journal of Advanced Robotic Systems*, 16(3):1729881419847897, 2019.

[149] H. Lim, Y. Kang, J. Kim, and C. Kim. Formation control of leader following unmanned ground vehicles using nonlinear model predictive control. In *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 945–950, 2009.

[150] G. Franzé, W. Lucia, and F. Tedesco. A distributed model predictive control scheme for leader–follower multi-agent systems. *International Journal of Control*, 91(2):369–382, 2018.

[151] S. Siniscalchi-Minna, F. D. Bianchi, C. Ocampo-Martinez, J. L. Domínguez-García, and B. De Schutter. A non-centralized predictive control strategy for wind farm active power control: A wake-based partitioning approach. *Renewable Energy*, 150:656–669, 2020.

[152] W. Ananduta and C. Ocampo-Martinez. Event-triggered partitioning for non-centralized predictive-control-based economic dispatch of interconnected microgrids. *Automatica*, 132:109829, 2021.

[153] J. Barreiro-Gomez, C. Ocampo-Martinez, and N. Quijano. Time-varying partitioning for predictive control design: Density-games approach. *Journal of Process Control*, 75:1–14, 03 2019.

[154] C. D. Godsil and G. F. Royle. Algebraic graph theory. In *Graduate texts in mathematics*, 2001.

[155] F. Glover and E. Woolsey. Further reduction of zero-one polynomial programming problems to zero-one linear programming problems. *Operations Research*, 21(1):156–161, 1973.

[156] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[157] N. Biggs. *Cambridge mathematical library: Algebraic graph theory*. Cambridge University Press, Cambridge, England, 2 edition, February 1994.

[158] G. Delansnay and A. V. Wouwer. Design of a reference governor in a zonotopic framework applied to a quadrotor under feedback linearization control strategies. *Journal of Intelligent and Robotic Systems*, 109(1), August 2023.

[159] G. Delansnay, L. Dewasme, and A. V. Wouwer. Tube-based nonlinear model predictive control of multirotor aircraft using a polynomial zonotopic framework. (1), 2024.

[160] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.