

COMPARATIVE ANALYSIS OF DATA-DRIVEN PREDICTIVE CONTROL
METHODS FOR QUADROTOR SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

RAMAZAN KÜRŞAT GEDİK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
AEROSPACE ENGINEERING

MARCH 2025

Approval of the thesis:

**COMPARATIVE ANALYSIS OF DATA-DRIVEN PREDICTIVE CONTROL
METHODS FOR QUADROTOR SYSTEMS**

submitted by **RAMAZAN KÜRŞAT GEDİK** in partial fulfillment of the requirements for the degree of **Master of Science in Aerospace Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Serkan Özgen
Head of Department, **Aerospace Engineering**

Assoc. Prof. Dr. Halil Ersin Söken
Supervisor, **Aerospace Engineering, METU**

Examining Committee Members:

Prof. Dr. Ozan Tekinalp
Aerospace Engineering, METU

Assoc. Prof. Dr. Halil Ersin Söken
Aerospace Engineering, METU

Prof. Dr. Klaus Werner Schmidt
Electrical and Electronics Engineering, METU

Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering, METU

Prof. Dr. Murat Efe
Electrical and Electronics Engineering, Ankara University

Date:20.03.2025



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Ramazan Kürşat Gedik

Signature :

ABSTRACT

COMPARATIVE ANALYSIS OF DATA-DRIVEN PREDICTIVE CONTROL METHODS FOR QUADROTOR SYSTEMS

Gedik, Ramazan Kürşat

M.S., Department of Aerospace Engineering

Supervisor: Assoc. Prof. Dr. Halil Ersin Söken

MARCH 2025, 106 pages

Traditional control methods are widely employed in the industry and are mostly based on mathematical models representing system dynamics. While effective, these mathematical models may fail to capture the full complexity of real-world systems due to uncertainties, nonlinearities, or unmodeled dynamics. On the other hand, data-driven methods provide an alternative to model-based methods, using empirical data instead of relying exclusively upon a mathematical model. These methods employ some form of optimization algorithm in order to connect real-time measurements to previously collected system data. The controller designed based on a data-driven approach can operate without having the system model in detail.

This study explores the implementation of data-driven predictive controllers on a quadrotor model to achieve trajectory tracking. Three different data-driven methods are evaluated regarding tracking performance and computational efficiency. The study aims to determine the most suitable data-driven control approach for quadrotor trajectory tracking, considering trade-offs between accuracy and computing time.

Additionally, the best-performing data-driven method is compared with a conven-

tional linear Model Predictive Controller (MPC) under conditions of parameter uncertainty in the quadrotor model. The comparison proves that data-driven methods are robust enough against uncertainties in the system dynamics. This study highlights the significant potential of data-driven predictive control for managing large-scale systems with uncertainties, comparing it with model-based controllers. The findings contribute meaningfully to the development of advanced control strategies, particularly in the field of aerial robotics.

Keywords: data-driven, control, tracking, robotics



ÖZ

DÖRT PERVANELİ SİSTEMLER İÇİN VERİYE DAYALI ÖNGÖRÜLÜ KONTROL YÖNTEMLERİNİN KARŞILAŞTIRMALI ANALİZİ

Gedik, Ramazan Kürşat

Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Halil Ersin Söken

Mart 2025 , 106 sayfa

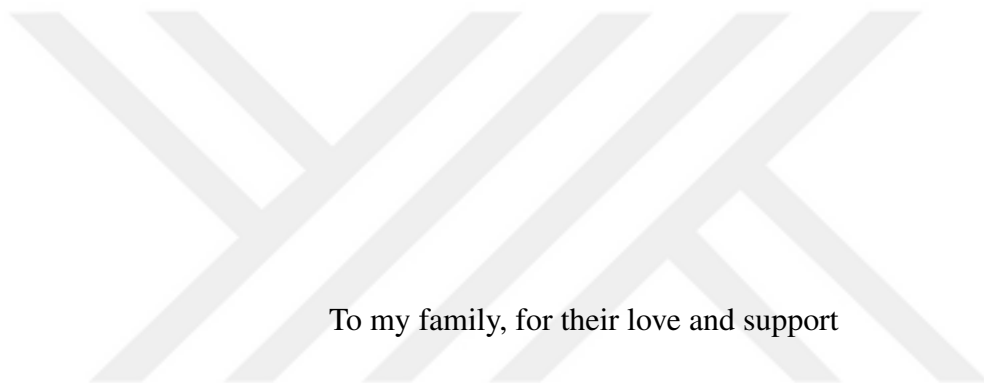
Geleneksel kontrol yöntemleri, endüstride yaygın olarak kullanılmakta ve genellikle sistem dinamiklerini temsil eden matematiksel modellere dayanmaktadır. Etkili olmalarına rağmen, bu matematiksel modeller, belirsizlikler, doğrusal olmayan yapılar veya modellenmemiş dinamikler nedeniyle gerçek sistemlerinin tam karmaşıklığını yakalayamayabilir. Diğer taraftan, veriye dayalı öngörülü kontrol yöntemleri, yalnızca matematiksel modellere dayanmak yerine, deneysel veriler kullanarak bir alternatif sunmaktadır. Bu yöntemler gerçek-zamanlı ölçümleri, daha önce toplanan sistem verileriyle ilişkilendirmek için bazı optimizasyon algoritması türleri kullanır. Veriye dayalı yaklaşıma dayanarak tasarlanan kontrolcü, detaylı sistem modeline sahip olmadan görev yapabilir.

Bu çalışma, bir dört pervaneli sistem modelinde yörünge takibi sağlamak için veriye dayalı öngörülü kontrolcülerin uygulanmasını incelemektedir. Üç farklı veriye dayalı yöntem, yörünge takibi performansı ve hesaplama verimliliği açısından değerlendirilmiştir. Çalışma, dört pervaneli sistemlerin yörünge takibi için doğruluk ve hesaplama süresi arasındaki dengeleri dikkate alarak en uygun veriye dayalı kontrol yaklaşımını

belirlemeyi amaçlamaktadır.

Ayrıca, en iyi performans gösteren veriye dayalı yöntem, dört pervaneli sistem modelindeki parametre belirsizlikleri altında geleneksel bir doğrusal Model Öngörülü Kontrolcü (MÖK) ile karşılaştırılmıştır. Bu karşılaştırma, veriye dayalı yöntemlerin sistem dinamiklerindeki belirsizlikler karşısında yeterince gürbüz olduğunu ortaya koymaktadır. Bu çalışma, büyük ölçekli ve belirsizlik içeren sistemlerin kontrolünde veri odaklı öngörülü yöntemin önemli potansiyelini vurgulamakta ve bunu model tabanlı kontrolcülerle de karşılaştırarak yapmaktadır. Bulgular, özellikle hava robotik sistemleri alanında, farklı kontrol stratejilerinin geliştirilmesine anlamlı bir katkı sunmaktadır.

Anahtar Kelimeler: veriye-dayalı, kontrol, takip, robotik



To my family, for their love and support

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my mother and father for their endless love, support, and encouragement throughout this journey.

I extend my sincere gratitude to my supervisor, Associate Professor Dr. Halil Ersin Söken, for his invaluable guidance, mentorship, and insightful contributions, which have played a crucial role in the successful completion of this work.

I sincerely thank ROKETSAN for the resources and opportunities that enabled this research, as well as my teammates at ROKETSAN for their collaboration and support.

Finally, I want to express my gratitude to my friends for their encouragement and unwavering belief in me throughout this process.

TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vii
ACKNOWLEDGMENTS	x
TABLE OF CONTENTS	xi
LIST OF TABLES	xiv
LIST OF FIGURES	xvi
LIST OF ABBREVIATIONS	xix
LIST OF SYMBOLS	xx

CHAPTERS

1 INTRODUCTION	1
1.1 The History of Quadrotors	1
1.2 The Use of Quadrotors	3
1.3 Control	4
1.4 Data-driven Predictive Control Approach	5
1.5 Thesis Contribution	9
1.6 Outline	9
2 MATHEMATICAL MODEL	11

2.1	Preliminaries	11
2.1.1	Reference Frames	12
2.1.2	Euler Angles	13
2.2	Quadrotor Mathematical Model	15
2.2.1	Forces and Moments	18
2.2.2	Linear Model	19
3	THEORY	23
3.1	Preliminaries	23
3.1.1	Quadratic Programming	23
3.1.2	Interior Point Methods	24
3.1.2.1	The Barrier Method	24
3.1.2.2	Newton's Method	25
3.1.2.3	Karush-Kuhn-Tucker Conditions	26
3.1.3	Model Predictive Control	28
3.1.4	Hankel Matrix	30
3.2	Data-driven Predictive Control	31
3.2.1	Subspace Predictive Control	32
3.2.2	Data-enabled Predictive Control	33
3.2.3	γ -data-driven Predictive Control	38
3.2.4	Summary	40
3.2.5	Implementation Guide	41
3.2.5.1	Data Collection	42
3.2.5.2	Tuning	43

4	SIMULATION RESULTS	47
4.1	Quadrotor Parameters	47
4.2	Data Collection	48
4.3	Choice of Prediction Horizon and Sampling Rate	54
4.4	Choice of Regularization Parameters	55
4.5	Parameter Settings of Data-driven Predictive Control Methods	58
4.6	Comparison of Data-driven Predictive Control Methods	60
4.7	Comparison of Computing Times of Data-driven Predictive Controllers	74
4.8	Comparison of Predictive Controllers with Data-driven and Model-based Methods	75
4.8.1	Ideal Case	75
4.8.2	Model Mismatch Case	82
4.8.3	Computational Cost	89
5	CONCLUSIONS	93
	REFERENCES	97
	APPENDICES	
A	EXPLICIT MATRIX EXPRESSIONS FOR THE MATHEMATICAL MODEL	103
A.1	The Matrix for the Rotational Kinetic Energy	103
A.2	Coriolis Matrix	104
B	SETTINGS FOR LINEAR AND NONLINEAR MPC	105

LIST OF TABLES

TABLES

Table 1.1	Summary table of differences between DDPC and AI-based methods	8
Table 3.1	SPC summary	40
Table 3.2	DeePC summary	41
Table 3.3	γ -DDPC summary	41
Table 4.1	Fixed model parameters for the quadrotor system	47
Table 4.2	MPC settings to generate data	48
Table 4.3	Trajectory parameters to generate the data	49
Table 4.4	Standard deviation of the noises	49
Table 4.5	Error Comparison for Different N and T_s Values	54
Table 4.6	SPC settings	58
Table 4.7	DeePC settings	59
Table 4.8	γ -DDPC settings	60
Table 4.9	Comparison of settling times	64
Table 4.10	Comparison of rise times	65
Table 4.11	Comparison of overshoot values	66
Table 4.12	The number of acceptable overshoot responses on position states . .	66

Table 4.13 Trajectory parameters to be followed	75
Table 4.14 Integral Absolute Error (IAE) Comparison for the Ideal Case	78
Table 4.15 Input Energy (IE) Comparison the Ideal Case	82
Table 4.16 Integral Absolute Error (IAE) Comparison for the Model Mismatch Case	85
Table 4.17 Input Energy (IE) Comparison for the Model Mismatch Case	89
Table B.1 Linear MPC settings for the trajectory tracking	105
Table B.2 Nonlinear MPC settings for the trajectory tracking	106

LIST OF FIGURES

FIGURES

Figure 1.1	Presentation of the historical quadrotor examples	2
Figure 1.2	A basic diagram of data-driven control approach in this study . .	6
Figure 2.1	An example of quadrotors	11
Figure 2.2	NED reference frame	12
Figure 2.3	Illustration of the body-fixed reference frame for the quadrotor .	13
Figure 2.4	Euler angles for 3-2-1 (Z-Y-X) convention	14
Figure 2.5	Scheme of the quadrotor mathematical model	19
Figure 4.1	Position trajectories for 100 randomly generated datasets	50
Figure 4.2	Euler angles trajectories for 100 randomly generated datasets . .	51
Figure 4.3	Velocity trajectories for 100 randomly generated datasets	51
Figure 4.4	Euler angle rates trajectories for 100 randomly generated datasets	52
Figure 4.5	Input trajectories for 100 randomly generated datasets	53
Figure 4.6	Position tracking error with various λ_α and λ_σ	56
Figure 4.7	Position tracking error with various β_2 and β_3	57
Figure 4.8	Responses of the x -position for the data-driven methods with different datasets	61

Figure 4.9	Position mean responses for the methods with different datasets .	62
Figure 4.10	Standard deviations of position responses for the methods with different datasets	63
Figure 4.11	Euler angles responses for the methods with different datasets . .	68
Figure 4.12	Euler angles mean responses for the methods with different datasets	69
Figure 4.13	Standart deviations of Euler angles responses for the methods with different datasets	70
Figure 4.14	Comparative analysis of control inputs with different datasets . .	71
Figure 4.15	Mean control inputs for the methods with different datasets . . .	72
Figure 4.16	Standard deviations of control inputs for the methods with dif- ferent datasets	73
Figure 4.17	Computing time comparison	74
Figure 4.18	Position outputs comparison of data-driven and model-based controller on the ideal case	76
Figure 4.19	Mean position outputs comparison of data-driven and model- based controller on the ideal case	77
Figure 4.20	Control inputs comparison of data-driven and model-based con- troller on the ideal case	79
Figure 4.21	Mean control inputs comparison of data-driven and model-based controller on the ideal case	80
Figure 4.22	Standard deviations of control inputs comparison of data-driven and model-based controller on the ideal case	81
Figure 4.23	Position outputs comparison of data-driven and model-based controller on the model mismatch case	83

Figure 4.24	Mean position outputs comparison of data-driven and model-based controller on the model mismatch case	84
Figure 4.25	Control inputs comparison of data-driven and model-based controller on the model mismatch case	86
Figure 4.26	Mean control inputs comparison of data-driven and model-based controller on the model mismatch case	87
Figure 4.27	Standard deviations of control inputs comparison of data-driven and model-based controller on the model mismatch case	88
Figure 4.28	Computing time comparison of data-driven and model-based controller on an ideal case	90
Figure 4.29	Computing time comparison of data-driven and model-based controller on an the model mismatch case	91

LIST OF ABBREVIATIONS

BC	Boundary Constraint
CPU	Central Processing Unit
DDPC	Data-driven Predictive Controllers
DeePC	Data-enabled Predictive Control
KKT	Karush-Kuhn-Tucker
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
NED	North-East-Down
NLMPC	Nonlinear Model Predictive Control
PID	Proportional Integral Derivative
QP	Quadratic Programming
SMC	Sliding Mode Controller
SPC	Subspace Predictive Control
s.t.	Subject To
std	Standard Deviation
UAV	Unmanned Aerial Vehicle
γ -DDPC	γ -data-driven Control

LIST OF SYMBOLS

$\mathbb{S}_+^n$	Set of $n \times n$ symmetric positive semidefinite matrices
$\mathbb{S}_{++}^n$	Set of $n \times n$ symmetric positive definite matrices
$\mathbb{R}^n$	n -dimensional real vector space
$\mathbb{R}^n$	$n \times n$ -dimensional real matrix space
dom f	Domain of f
$\inf_x f$	The greatest lower bound of f
$\sup_x f$	The lowest upper bound of f
$\mathbb{N}_{\geq a}$	Set of integers greater than a
col	Column vector operator
N	Prediction horizon
rank	Rank operator
$\dagger$	Pseudoinverse operator
$\langle, \rangle$	Inner product operator
$\otimes$	Kronecker product operator

CHAPTER 1

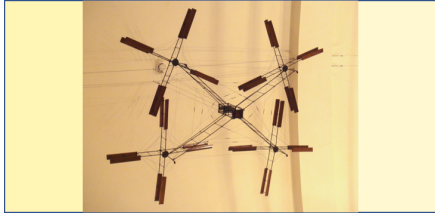
INTRODUCTION

1.1 The History of Quadrotors

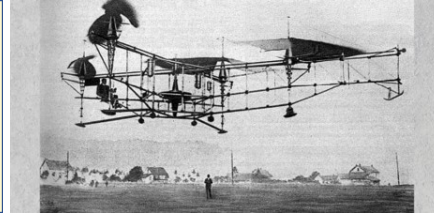
The concept of quadrotors, or four-rotor helicopters, has a rich history dating back to the early 20th century. Aviation pioneers conducted initial experiments with quadrotor designs to create stable and controllable vertical flight vehicles.

In 1907, Louis and Jacques Breguet, in collaboration with Charles Richet, developed Gyroplane No. 1 [30], marking the first rotary-wing aircraft to achieve vertical takeoff with a pilot aboard, although it required ground stabilization. It featured a rectangular steel chassis with four arms and fabric-covered, four-blade biplane rotors, creating 32 lifting surfaces. The pilot, M. Volumard, weighed 150 lbs (68 kg) and helped achieve modest lift, but the aircraft lacked control and steerability. In 1908, Gyroplane No. 2 [30] was introduced with a 55-horsepower Renault engine and fixed wings for more lift. While it performed better, it was damaged in a crash and later displayed as a static exhibit. Due to limitations in engine technology, Breguet shifted his focus to fixed-wing aircraft designs in subsequent years.

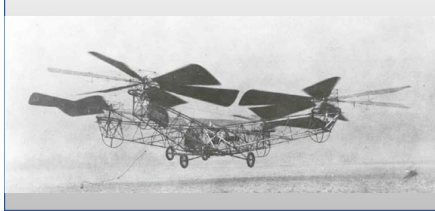
In 1920, Étienne Oehmichen, an engineer at Peugeot, began developing rotary-wing aircraft, creating six vertical take-off machines. His notable Oehmichen No. 2 featured a steel-tube framework, four rotors, and eight propellers powered by a 120 hp Le Rhône engine, later upgraded to 180 hp. It achieved milestones like the first closed-circuit helicopter flight in 1924 and a record distance of 525 meters. Unsatisfied with its altitude, he later shifted to a single main rotor with two anti-torque rotors, culminating in a 1938 design that revisited the balloon-assisted concept, significantly advancing rotary-wing aviation. [30].



(a) Gyroplane No. 1 [37]



(b) Oehmichen No. 2 [21]



(c) The de Bothezat helicopter [38]



(d) Model A Quadrotor [39]

Figure 1.1: Presentation of the historical quadrotor examples

Following Oehmichen's contributions, Dr. George de Bothezat and Ivan Jerome introduced another notable quadrotor in the early 1920s under the U.S. Army's sponsorship. Their aircraft, the de Bothezat helicopter [54], featured an X-shaped structure with six-bladed rotors and employed variable-pitch propellers for thrust and yaw control. The design reached a maximum altitude of about 5 meters but was complex mechanically and had serious reliability issues. While it showed that quadrotor flight was possible, it struggled due to weak engines and complicated control systems, which made it demanding for the pilot to operate.

Further development in quadrotor technology continued into the mid-20th century. In 1956, Convertawings introduced the Model A Quadrotor as a prototype for larger civil and military quadrotor helicopters [23]. The design utilized two engines to drive four rotors with attached wings for additional lift during forward flight. Unlike its predecessors, the Model A did not require a tail rotor, as control was achieved through differential thrust between the rotors. Despite successful test flights, the Model A did not reach commercial production due to a lack of demand, though it did lay the groundwork for subsequent advancements in quadrotor technology.

The history of quadrotors reflects diversified challenges and innovations in mastering

controlled vertical flight. While modern designs overcame most problems, they still face issues such as mechanical complexity and limits in control systems. Recent technological advances are proving to be a game changer. We can now navigate more precisely and take on various tasks with improved algorithms of control and lighter materials. From early prototypes to today's high-performance drones, there has been a big jump in this field, showing continuous interest in the peculiar capabilities of quadrotor aircraft.

1.2 The Use of Quadrotors

Numerous studies, research, and investigations are underway on quadrotors used in the military and civilian sectors. They present the only solution for specific problems that cannot be solved by autonomous ground and underwater vehicles. Additionally, applications that demand hovering flight restrict the use of autonomous aircraft to rotary-wing vehicles like quadrotors. The application domains range from surveillance, search and rescue, aerial photography, disaster management, forest monitoring, military war, and aerobatic flights for entertainment [24, 52]. UAVs, especially quadrotors, have served as an attractive aerial research platform for the design and experimental validation of new strategies for guidance, state estimation, and control systems since their natural behavior raises various challenges that are not addressed in other robotic systems [4].

Quadrotors, available in various shapes and sizes, are becoming more common in everyday activities across the globe [41]. Currently, enhancing drone flight capabilities is increasingly difficult, particularly in accurately stabilizing and tracking their trajectories. Many perceive these vehicles unfavorably, which is inherently tied to the performance of the control algorithm. Developing an efficient flight control algorithm for drones remains a work in progress. This is particularly true when moving from a simple theoretical model to real-world use, where many uncertainties arise from imperfect models, inaccurate aerodynamic coefficients, and outside factors [51, 47]. With all these factors, the control of these aircraft becomes an essential task even in case of minor deficiencies in the model, and there is an excellent possibility of the system moving out of the targeted path or following an undesired path that may cause

losing the control and even crashing the aircraft.

Significant investment has been made in quadrotors for several application fields in fully autonomous or remotely piloted configurations. Various attempts have been made to integrate such machines into more complex tasks, like flying longer distances with payloads [46, 11]. Today, quadrotor drones are being used for medical deliveries, which can reduce delivery times by more than half while minimizing human contact, which is particularly important for decreasing the transmission of contagious diseases, such as the coronavirus [36]. In some regions, governments have deployed quadrotors to disinfect public spaces without human intervention. Other countries have mounted quadrotors with speakers to issue specific directives and awareness among the citizens to motivate them to stay at home. Quarantine areas are being maintained through quadrotors [14].

1.3 Control

The design and analysis of a control system are commonly based on a model-based approach. If the mathematical model is available for the system, the controller is designed according to appropriate design laws. On the other hand, when mathematical modeling is not possible, system identification methods can be employed. System identification involves collecting empirical data from the system and using statistical [2] or machine learning [28] techniques to construct a model representing the system's behavior. This model can then be used to design a control system.

For the quadrotor case, many control methods based on a model-based approach are available in the literature. As two straightforward examples, Proportional Integral Derivative (PID) [5] and Linear Quadratic Regulator (LQR) [19] controllers can be used. Moreover, nonlinear control methods can be considered to get better performance. Feedback linearization is used for optimal tracking control in [10], and a double loop integral Sliding Mode Controller (SMC) is used for robust trajectory tracking control in [48]. Also, Model Predictive Control (MPC) has recently been used in quadrotor control due to its advantages when operating with constraints and state prediction. The study in [22] presents an MPC for trajectory tracking under

disturbances to achieve a robust performance for the quadrotor. In addition, [40] provides real-time operation of MPC accounting aerodynamic forces, making trajectory tracking challenging.

On the other hand, data-driven predictive controllers (DDPC) directly try to achieve the control objective by relying on the collected empirical data from the system. Similarly to the MPC, the DDPC can obey the constraints and predict the future states. In the literature, DDPC approaches are divided into two classes, indirect and direct methods [18]. Indirect methods are based on identifying prediction matrices used in the controller structure from the collected data. One example of an indirect method is the Subspace Predictive Control (SPC) [15]. Also, direct methods can be used in the DDPC approach as in [13] where the states are predicted directly from the collected data by using the behavioral system theory and the Willems' fundamental lemma [50]. In this thesis, we present the implementation of three DDPC approaches: two direct methods and one indirect method.

1.4 Data-driven Predictive Control Approach

The increasing complexity of modern systems and the prevalence of uncertainties have driven the need for innovative control methodologies that transcend traditional model-based designs. Classical control relies on deriving precise mathematical models of system dynamics, which are often infeasible in real-world scenarios due to noise, nonlinearity, or unmodeled dynamics. In response, data-driven control has emerged as a transformative approach that leverages empirical system data to design and optimize controllers directly, bypassing the need for explicit parametric models [13, 27]. Rooted in the principles of behavioral systems theory, this paradigm focuses on capturing the set of all trajectories a system can exhibit under certain constraints, providing a model-agnostic foundation for control synthesis [49].

Data-driven control approaches, such as SPC, Data-Enabled Predictive Control (DeePC), and γ -data-driven Control (γ -DDPC), exploit input-output data to achieve desired performance while accommodating system uncertainties [42, 43]. These methods are particularly advantageous in scenarios where deriving accurate models is challenging,

such as in systems with high-dimensional dynamics, time-varying parameters, or significant environmental disturbances. γ -DDPC, for instance, offers a robust framework by addressing noisy measurements and parameter uncertainties, making it suitable for applications ranging from robotics to aerospace systems.

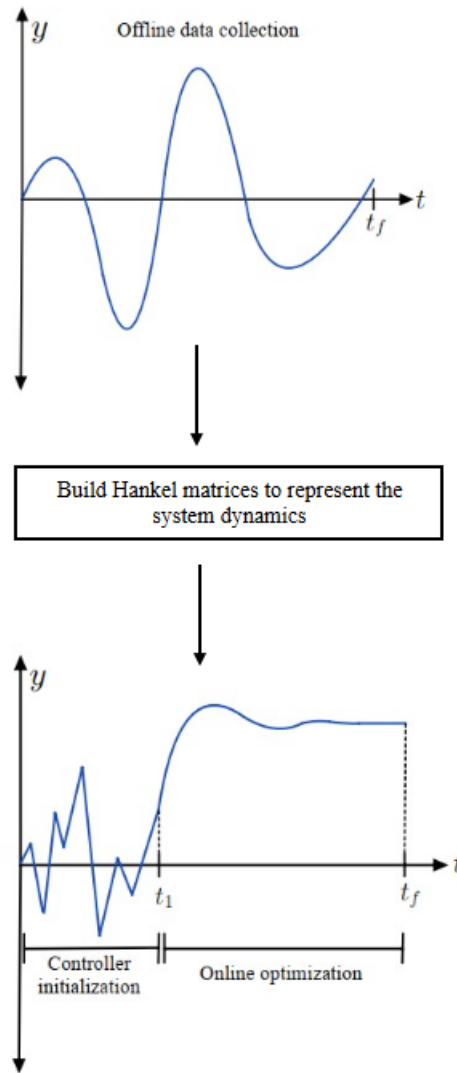


Figure 1.2: A basic diagram of data-driven control approach in this study

The shift towards data-driven control has also opened new avenues for addressing key challenges in control theory, such as robustness to uncertainties, adaptation to varying conditions, and scalability to large systems [18]. By focusing directly on the

data rather than a fixed mathematical model, these approaches inherently adapt to the system's observed behavior, enabling improved performance in dynamic and uncertain environments. As the field advances, integrating data-driven control with modern computational tools and optimization techniques continues to push the boundaries of what can be achieved in both theoretical and applied control systems.

Figure 1.2 presents a foundational diagram outlining the data-driven approach that is utilized in this study. To begin with, offline data is collected from the system, which serves as the basis for the construction of Hankel matrices—detailed explanations of which will be provided in the subsequent sections. During the online phase, data is gathered at the onset of the simulation to set the stage for initializing the optimization problem. This initial data collection is considered crucial as it establishes the parameters needed to guide the optimization process. Once the initialization phase is successfully completed, the online optimization problem is ready to be tackled. Through this iterative process, the optimization algorithm continuously refines its calculations, determining the optimal control inputs over successive iterations, thereby enhancing the system's performance throughout the simulation. DDPC and AI-based data-driven control methods differ significantly in their modeling approaches, adaptability, stability guarantees, computational complexity, and interpretability. DDPC, commonly implemented as data-driven MPC, utilizes measured process data and optimization techniques, such as trajectory-based formulations and behavioral theory, to develop control laws without explicitly identifying a system model [29]. This method provides strong theoretical guarantees, including stability and recursive feasibility, making it suitable for safety-critical applications like industrial process control, robotics, and power systems [26, 1].

In contrast, AI-based data-driven methods, such as reinforcement learning (RL) and deep learning-based control, use machine learning models to approximate system behavior, often requiring extensive offline training and large datasets [17, 34]. While these methods offer greater flexibility and adaptability in high-dimensional, uncertain environments, they typically lack explicit stability guarantees and are computationally intensive [3]. Additionally, AI-driven approaches often act as black-box models, which limits their explainability. In contrast, DDPC maintains interpretability due to its reliance on convex optimization and explicit constraint handling [17].

Although AI-based control has gained traction in autonomous systems and adaptive robotics, its practical application in real-time safety-critical environments remains challenging [16]. Recent advancements in safe reinforcement learning and hybrid predictive control strategies aim to bridge this gap by combining the adaptability of AI-based learning with the robustness and stability guarantees offered by DDPC [50, 53].

Table 1.1 provides a detailed comparison highlighting the key distinctions between data-driven methods and AI-based approaches. Data-driven methods primarily rely on analyzing and interpreting existing datasets to identify patterns and make predictions. In contrast, AI-based methods utilize advanced algorithms and models that can learn from data, adapt over time, and make decisions independently. This table outlines specific differences, such as the techniques used, the degree of human intervention required, and the scalability of each approach. It comprehensively explains how these methodologies diverge in their applications and effectiveness.

Table 1.1: Summary table of differences between DDPC and AI-based methods

Feature	DDPC	AI-based methods
Model Dependence	Uses past trajectory data or system identification	Uses learned models (NN, RL policies, etc.)
Adaptability	Real-time updates based on recent data	Learns offline; adapts over long-term use
Theoretical Guarantees	Stability and feasibility ensured (e.g., robust MPC)	Often lacks strict stability guarantees
Computational Complexity	Solves real-time optimization problems	Requires intensive training (deep learning, RL)
Explainability	Highly interpretable	Often a black-box
Application Domain	Process control, smart grids, robotics	Autonomous systems, high-dimensional control

1.5 Thesis Contribution

This thesis demonstrates the transformative potential of data-driven predictive methods in enhancing quadrotor systems. Due to their inherent complexity and nonlinear dynamics, controlling these systems poses significant challenges. However, integrating data-driven methods offers a powerful solution to these obstacles. This contribution provides a comprehensive implementation guide for utilizing DDPC techniques specifically tailored for quadrotor applications, paving the way for innovation in this field.

Though the study in [13] and [12] employs the DeePC algorithm effectively for the quadrotor case, there is no comparison of other DDPC methods for quadrotors in the literature. This thesis compares these methods and identifies their advantages and disadvantages with respect to each other.

1.6 Outline

The remainder of this paper is structured as follows. Chapter 2 presents the mathematical model for the quadrotor used in the simulation study and controller designs, detailing the dynamic equations and assumptions made for simplification. Chapter 3 discusses the theoretical framework underlying this study, covering both model-based and data-driven control techniques. Specifically, it introduces SPC, DeePC, and γ -DDPC, explaining their fundamental principles. Chapter 4 presents the simulation results, comparing the performance of model-based and data-driven control approaches under various conditions, including parameter uncertainties. This chapter provides an analysis of the control accuracy, robustness, and computational efficiency of each method. Chapter 5 concludes the study with a summary of findings and insights gained from the simulation comparisons, offering recommendations for practical applications. It also discusses potential areas for future research and development, such as implementing these control strategies in real-time embedded systems or extending them to other types of aerial vehicles.



CHAPTER 2

MATHEMATICAL MODEL

2.1 Preliminaries

A quadrotor, commonly referred to as a quadcopter, is a type of unmanned aerial vehicle (UAV) distinguished by its ability to lift and navigate through the air using four rotors. Each rotor is composed of a motor that drives a propeller, enabling vertical takeoff and landing capabilities. The design typically features an electronic control board strategically located at the center of the frame, orchestrating the operation of the four motors situated at the corners of the quadrotor.



Figure 2.1: An example of quadrotors

To accurately develop a comprehensive mathematical model that captures the behavior of a quadrotor, it is vital to establish the reference frames that define its spatial position and orientation within the environment. There are two primary reference frames used in this context: the inertial reference frame, which acts as a stable, fixed

point in the surrounding space, and the body-fixed reference frame, which dynamically moves and rotates along with the quadrotor itself. Understanding these two frames is essential for a thorough analysis of the quadrotor's dynamics and control systems, as they provide a foundation for modeling its movements. In addition to these reference frames, Euler angles are utilized to precisely articulate the orientation of the quadrotor in three-dimensional space. These angles facilitate a clearer understanding of the quadrotor's rotational dynamics, making it easier to model its behavior and control its flight effectively.

2.1.1 Reference Frames

The inertial reference frame is where the first Newton's laws are considered valid. The North-East-Down (NED) reference frame is assumed to be an inertial reference frame. Axes x , y , and z are aligned with north, east, and down, respectively. Also, the inertial reference frame is denoted by $\mathcal{F}_I$, and the coordinates to be used in the mathematical model are x_I , y_I , and z_I . A graphical representation of the NED reference frame is given in Figure 2.2.

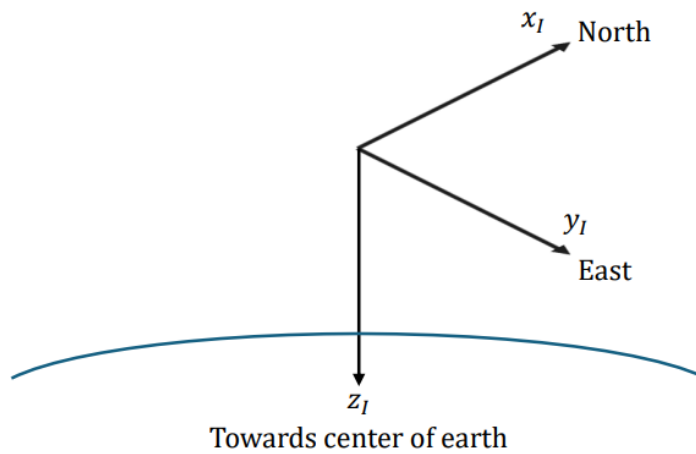


Figure 2.2: NED reference frame

The body-fixed reference frame is a frame moving with the quadrotor. Its origin is located at the center of gravity on the quadrotor. Most flight vehicles have a plane of

symmetry, which is chosen as the xz plane, with z directed downward. However, the quadrotor has two identical planes of symmetry, xz and yz . So, as a notation, x and y axes point out the first and second rotors of the quadrotor, respectively. As mentioned before, the z axis is aligned with the downward direction of the body. The body-fixed reference frame is denoted by $\mathcal{F}_B$, and the coordinates to be used in the mathematical model are x_B , y_B , and z_B . A graphical representation of the body-fixed reference frame is given in Figure 2.3.

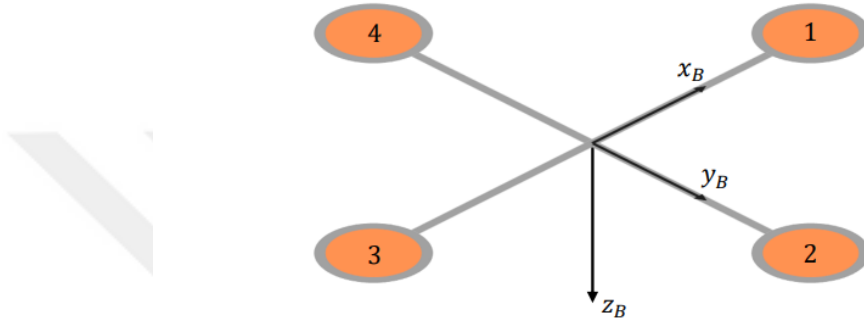


Figure 2.3: Illustration of the body-fixed reference frame for the quadrotor

2.1.2 Euler Angles

Three angles can give the orientation of any reference frame relative to another: the consecutive rotations about the axes z , y , and x in that order that carries one frame into coincidence with the other. This is a particular case of Euler angles. In flight dynamics of the quadrotor, the Euler angles used are those that rotate the fixed reference frame $\mathcal{F}_I$ such that it coincides with the body's fixed reference frame. The angles are denoted (ψ, θ, ϕ) for body axes. Figure 2.4 shows the sequence of rotations and Euler angles concerning the axes below.

1. A rotation ψ about z_I , carrying the axes to $x_1 y_1 z_1$, ψ is the yaw angle.
2. A rotation θ about y_1 , carrying the axes to $x_2 y_2 z_2$, θ is the pitch angle.
3. A rotation ϕ about x_2 , carrying the axes to their final position $x_B y_B z_B$, ϕ is the roll angle.

yaw angle.

In order to avoid ambiguities that can otherwise result with another set of angles (ψ, θ, ϕ) , the ranges are limited to

$$\begin{aligned} -\pi &\leq \psi < \pi \\ -\frac{\pi}{2} &\leq \theta < \frac{\pi}{2} \\ -\pi &\leq \phi < \pi \end{aligned}$$

The Euler angles are then unique for most orientations of the vehicle. However, it should be noted that in a continuous steady rotation, such as rolling, the time variation of ϕ , for example, is a discontinuous sawtooth function.

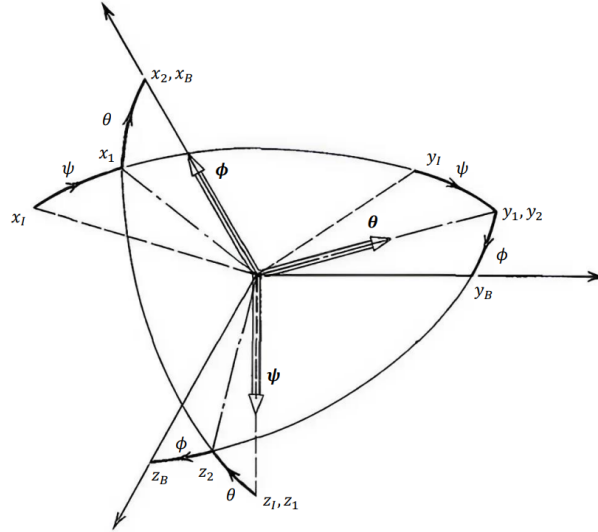


Figure 2.4: Euler angles for 3-2-1 (Z-Y-X) convention

As noted before, Euler angles represent a sequence of three elemental rotations, i.e., rotations about the axes of a coordinate system, since any orientation can be achieved by composing three elemental rotations. These rotations start from a known standard

orientation. The following rotation matrices describe this combination:

$$\begin{aligned}
R_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \\
R_y(\theta) &= \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \\
R_z(\psi) &= \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.1}$$

So, the inertial reference frame and the body-fixed reference frame are related by the rotation matrix $R^{(B,I)}(\phi, \theta, \psi)$.

$$\begin{aligned}
R^{(B,I)}(\phi, \theta, \psi) &= R_x(\phi)R_y(\theta)R_z(\psi) \\
&= \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi & \sin \phi \cos \theta \\ \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi & \cos \phi \cos \theta \end{bmatrix}
\end{aligned} \tag{2.2}$$

This matrix describes the rotation from the inertial reference frame to the body fixed reference frame.

2.2 Quadrotor Mathematical Model

This section provides a mathematical model of the quadrotor by applying Euler-Lagrange equations, which effectively capture the dynamics and interactions involved in the 3D motion of a rigid body, allowing for an accurate representation of its behavior under various conditions [9]. Firstly, generalized coordinates, μ , which describe the position and orientation of the quadrotor, are introduced. As introduced before, (ϕ, θ, ψ) describes the orientation of the quadrotor. The position states of the center of mass of the quadrotor in the inertial reference frame are denoted by (x, y, z) . Then, the generalized coordinates for the quadrotor are,

$$\mu = (x, y, z, \phi, \theta, \psi) \in \mathbb{R}^6 \tag{2.3}$$

Then, the approach of the mathematical model splits in translational and rotational coordinates

$$\begin{aligned}\zeta &= (x, y, z) \in \mathbb{R}^3 \\ \eta &= (\phi, \theta, \psi) \in \mathbb{S}^3\end{aligned}\tag{2.4}$$

The translational kinetic energy of the quadrotor is

$$T_{\text{trans}} = \frac{1}{2} m \dot{\zeta}^\top \dot{\zeta}\tag{2.5}$$

where m is the mass of the quadrotor. The rotational kinetic energy is

$$T_{\text{rot}} = \frac{1}{2} \dot{\eta}^\top J \dot{\eta}\tag{2.6}$$

The matrix J is the matrix for the rotational kinetic energy of the quadrotor expressed directly in terms of the generalized coordinates η . To explain in more detail, the rate of Euler angles, $\dot{\eta}$, is related to angular velocities in the body fixed frame, ω_B . The ω_B term can be obtained from the matrix operator on $\dot{\eta}$.

$$\begin{aligned}\omega_B &= T \dot{\eta} \\ \text{where } T &= \begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{bmatrix}\end{aligned}\tag{2.7}$$

The rotational kinetic energy is,

$$T_{\text{rot}} = \frac{1}{2} \omega_B^\top I \omega_B\tag{2.8}$$

where I is the inertia matrix. For this study, it is assumed that I has diagonal elements.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}\tag{2.9}$$

When Equation 2.7 is put into Equation 2.8, it gives,

$$T_{\text{rot}} = \frac{1}{2} \dot{\eta}^\top T^\top I T \dot{\eta}\tag{2.10}$$

Then, the matrix J can be written as,

$$J = T^\top I T\tag{2.11}$$

Furthermore, the potential energy on the quadrotor is the standard gravitational potential given by

$$U = -mgz \quad (2.12)$$

It is noted that the negative sign on the expression arises from the NED frame definition. Then, the Lagrangian is

$$\begin{aligned} L(\mu, \dot{\mu}) &= T_{\text{trans}} + T_{\text{rot}} - U \\ L(\mu, \dot{\mu}) &= \frac{1}{2}m\dot{\zeta}^\top \dot{\zeta} + \frac{1}{2}\dot{\eta}^\top J\dot{\eta} + mgz \end{aligned} \quad (2.13)$$

The mathematical expression of the model for the quadrotor dynamics is obtained from the Euler-Lagrange equations with external generalized force

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\mu}} - \frac{\partial L}{\partial \mu} = F \quad (2.14)$$

where $F = (F_\zeta, \tau)$. τ is the generalized moments and F_ζ is the translational force applied to the quadrotor due to the control inputs. The small body forces can be neglected since they are smaller in magnitude order than principal control inputs u and τ . Then, the translational force applied to the quadrotor in the body fixed frame can be expressed as,

$$\hat{F} = -u\hat{e}_3^B \quad (2.15)$$

where u is the vertical thrust. It is noted that $\hat{e}_3^B$ represents a unit vector along the z-axis in the body fixed reference frame. The force expression should be rotated into the inertial fixed frame by the rotation matrix $R^{(I,B)}$. Since $R^{(B,I)}$, the rotation matrix introduced before, is an orthogonal matrix, $R^{(I,B)}$ can be obtained by $(R^{(B,I)})^\top$.

$$F_\zeta = R^{(I,B)} \hat{F} \quad (2.16)$$

The generalized moments on the η variables are

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \quad (2.17)$$

Since the Lagrangian includes no multiplied terms of $\dot{\zeta}$ and $\dot{\eta}$ in the kinetic energy, the Euler-Lagrange equation can be split into the dynamics for the ζ coordinates and

the η dynamics. It is noted that $\hat{e}_3^I$ represents a unit vector along the z-axis in the inertial frame.

$$m\ddot{\zeta} - mg\hat{e}_3^I = F_\zeta \quad (2.18)$$

$$J\ddot{\eta} + \dot{J}\dot{\eta} - \frac{1}{2} \frac{\partial}{\partial \eta} \left(\dot{\eta}^\top J \dot{\eta} \right) = \tau \quad (2.19)$$

Defining the Coriolis/Centripetal vector

$$\bar{V}(\mu, \dot{\mu}) = \dot{J}\dot{\eta} - \frac{1}{2} \frac{\partial}{\partial \eta} \left(\dot{\eta}^\top J \dot{\eta} \right) \quad (2.20)$$

A new variable $C(\mu, \dot{\mu})$ referring to the Coriolis terms and gyroscopic and centrifugal terms associated with the η dependence of J can be introduced such that

$$C(\mu, \dot{\mu}) = \dot{J} - \frac{1}{2} \frac{\partial}{\partial \eta} \left(\dot{\eta}^\top J \right) \quad (2.21)$$

$$\bar{V}(\mu, \dot{\mu}) = C(\mu, \dot{\mu}) \dot{\eta} \quad (2.22)$$

Certainly, the mathematical model becomes,

$$\begin{aligned} \ddot{\zeta} &= -R^{(I,B)} \frac{u}{m} \hat{e}_3^B + g\hat{e}_3^I \\ \ddot{\eta} &= J^{-1} (-C(\mu, \dot{\mu}) \dot{\eta} + \tau) \end{aligned} \quad (2.23)$$

2.2.1 Forces and Moments

The generalized forces and moments acting on the quadrotor are explained in this section. Referring to Figure 2.5, the vertical thrust, u , can be written as the sum of forces generated by each rotor.

$$u = f_1 + f_2 + f_3 + f_4 \quad (2.24)$$

The rotors generate forces in a linear relationship with the power of the square of their angular velocities $\{\omega_i | i = 1, 2, 3, 4\}$.

$$f_i = b\omega_i^2 \quad (2.25)$$

where b is the thrust factor. The generalized moments are generated by torques from the rotation of the rotors. They are defined as,

$$\begin{aligned} \tau_x &= bl(\omega_4^2 - \omega_2^2) \\ \tau_y &= bl(\omega_1^2 - \omega_3^2) \\ \tau_z &= d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{aligned} \quad (2.26)$$

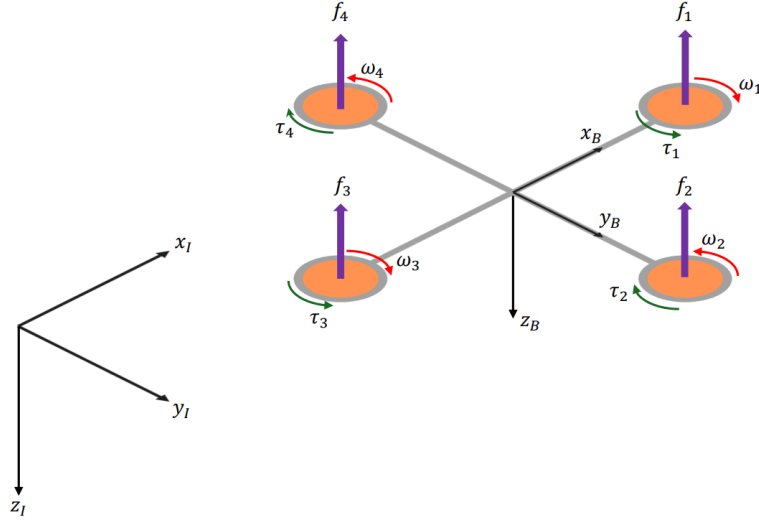


Figure 2.5: Scheme of the quadrotor mathematical model

where d is the drag factor, and l is the distance from the center of gravity to the center of rotors. It is assumed to be the same for each rotor.

2.2.2 Linear Model

This thesis employs data-driven control methods to control the quadrotor system. Although the quadrotor dynamics are inherently nonlinear, a linearized mathematical model is necessary for control. Thus, a nonlinear model is used only for the simulation from which the data is collected, while a linearized model is derived for implementation in MPC. This linearized form allows the controller to effectively operate using the data generated in this study.

To satisfy the requirements of MPC, the quadrotor system is approximated as a linear system around an equilibrium point. This linearization is essential because MPC relies on a linear system representation to predict future states and calculate control inputs efficiently. The control vector, $\mathbf{u}$, is defined to consist of the squared angular velocities of the four rotors:

$$\mathbf{u} = \begin{bmatrix} \omega_1^2 & \omega_2^2 & \omega_3^2 & \omega_4^2 \end{bmatrix}^T \in \mathbb{R}^4 \quad (2.27)$$

The state vector, $\mathbf{x}$, includes the position, orientation (Euler angles), and their deriva-

tives, represented as:

$$\mathbf{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \in \mathbb{R}^{12} \quad (2.28)$$

The system dynamics are described by the nonlinear function $f(\mathbf{x}, \mathbf{u})$ as shown in Equation 2.23. Linearization is performed around a steady-state equilibrium point $\bar{\mathbf{x}}$, with a corresponding fixed input $\bar{\mathbf{u}}$ that satisfies the condition:

$$f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = 0 \quad (2.29)$$

The equilibrium point $\bar{\mathbf{x}} = 0$ represents a stable hover state in which the quadrotor is at rest, with gravitational forces balanced by the thrust of the rotors. At this equilibrium, the thrust forces are required to counteract gravity, and the torques about each axis must sum to zero. Thus, the control input is defined to satisfy:

$$\sum_{i=1}^4 b\omega_i^2 = mg \quad (2.30)$$

Further, symmetry in rotor speeds is enforced to yield zero torque about the roll, pitch, and yaw axes:

$$\begin{aligned} \tau_x &= bl(\omega_4^2 - \omega_2^2) = 0 \\ \tau_y &= bl(\omega_1^2 - \omega_3^2) = 0 \\ \tau_z &= d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) = 0 \end{aligned} \quad (2.31)$$

By solving these conditions, the equilibrium control input is determined as:

$$\bar{\mathbf{u}} = \left\{ \omega_i^2 = \frac{mg}{4b} \mid i = 1, 2, 3, 4 \right\} \quad (2.32)$$

With the equilibrium point defined, the linearized state-space model is derived using a first-order Taylor expansion around $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$:

$$\dot{\Delta \mathbf{x}} = A\Delta \mathbf{x} + B\Delta \mathbf{u} \quad (2.33)$$

where $\Delta \mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ and $\Delta \mathbf{u} = \mathbf{u} - \bar{\mathbf{u}}$. The system matrices A and B are obtained as the Jacobians of f with respect to $\mathbf{x}$ and $\mathbf{u}$, evaluated at the equilibrium.

$$\begin{aligned} A &= \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}=\bar{\mathbf{x}}, \mathbf{u}=\bar{\mathbf{u}}} \\ B &= \left. \frac{\partial f}{\partial \mathbf{u}} \right|_{\mathbf{x}=\bar{\mathbf{x}}, \mathbf{u}=\bar{\mathbf{u}}} \end{aligned} \quad (2.34)$$

The state matrix A is expressed as follows:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -g/b & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g/b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.35)$$

The input matrix B is represented as:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{b}{m} & -\frac{b}{m} & -\frac{b}{m} & -\frac{b}{m} \\ 0 & -\frac{bl}{I_{xx}} & 0 & \frac{bl}{I_{xx}} \\ \frac{bl}{I_{yy}} & 0 & -\frac{bl}{I_{yy}} & 0 \\ -\frac{d}{I_{zz}} & \frac{d}{I_{zz}} & -\frac{d}{I_{zz}} & \frac{d}{I_{zz}} \end{bmatrix} \quad (2.36)$$



CHAPTER 3

THEORY

3.1 Preliminaries

3.1.1 Quadratic Programming

A convex optimization problem can be expressed as

$$\begin{aligned} \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_i(x) = 0, \quad i = 1, \dots, p \end{aligned} \tag{3.1}$$

where $f_0, \dots, f_m$ are convex functions. Inequality constraint functions are represented by $f_i(x)$. Equality constraints are defined by $h_i(x)$. If the objective function, f_0 , is quadratic and the constraint functions are affine, Equation 3.1 is called a quadratic programming (QP) problem. A QP problem can be expressed in the form

$$\begin{aligned} \min \quad & \frac{1}{2} x^\top P x + q^\top x \\ \text{s.t.} \quad & f_i(x) \leq 0 \quad i = 1, \dots, m \\ & h_i(x) = 0, \quad i = 1, \dots, p \end{aligned} \tag{3.2}$$

where:

- $P \in \mathbb{S}_+^n$: A symmetric positive semidefinite matrix representing the quadratic term. This ensures the convexity of the objective function.
- $q \in \mathbb{R}^n$: A vector representing the linear term of the objective

There are methods in the literature to solve QP problems. Since optimization problems in this study are large-scale QP problems, the interior-point methods are efficient

in solving the problem [32]. The following section overviews the theory.

3.1.2 Interior Point Methods

The interior point methods are powerful optimization algorithms used to solve QP problems, which involve minimizing a quadratic objective function subject to linear equality and inequality constraints. Due to its computational efficiency, robustness, and scalability, it is particularly effective for solving large-scale optimization problems. This section provides a detailed exploration of the barrier method, a kind of interior point method, including its theoretical foundations, practical implementation, and integration with Newton's method.

3.1.2.1 The Barrier Method

The interior point method incorporates inequality constraints $f_i(x) \leq 0$ into the objective function using a barrier method. This approach introduces a barrier function $\phi(x)$ that imposes a penalty as x approaches the boundary of the feasible region. For inequality constraints, the logarithmic barrier function is commonly used:

$$\phi(x) = - \sum_{i=1}^m \ln(-f_i(x)) \quad (3.3)$$

with $\text{dom } \phi = \{x \in \mathbb{R}^n | f_i(x) < 0, i = 1, \dots, m\}$. This transforms the original constrained problem into an unconstrained problem by augmenting the objective function with the barrier term. The new objective function becomes:

$$f_\mu(x) = \frac{1}{2} x^\top P x + q^\top x - \mu \sum_{i=1}^m \ln(-f_i(x)) \quad (3.4)$$

where $\mu > 0$ is the barrier parameter. The parameter μ controls the trade-off between the original objective and the barrier term:

- For large μ , the solution prioritizes minimizing the original objective $f_0(x)$.
- For small μ , the solution allows proximity to the constraint boundaries, approaching the true feasible optimum.

The barrier method iteratively solves this augmented problem, progressively reducing μ to converge to the solution of the original QP problem [6].

3.1.2.2 Newton's Method

Newton's method is employed to solve the unconstrained problem defined by $f_\mu(x)$. This iterative optimization technique relies on approximating the objective function locally using a second-order Taylor expansion. For the augmented function $f_\mu(x)$, the Taylor expansion at a point x is:

$$f_\mu(x + \Delta x) \approx f_\mu(x) + \nabla f_\mu(x)^\top \Delta x + \frac{1}{2} \Delta x^\top H(f_\mu(x)) \Delta x \quad (3.5)$$

where $\nabla f_\mu(x)$ is the gradient of $f_\mu(x)$, $H(f_\mu(x))$ is the Hessian matrix of $f_\mu(x)$, Δx is the Newton step, representing the adjustment to the decision variables.

From Equation 3.4, the gradient is computed as:

$$\nabla f_\mu(x) = Px + q - \sum_{i=1}^m \frac{\mu}{f_i(x)} \nabla f_i(x) \quad (3.6)$$

The Hessian is:

$$H(f_\mu(x)) = P + \sum_{i=1}^m \frac{\mu}{(f_i(x))^2} \nabla f_i(x) \nabla f_i(x)^\top - \sum_{i=1}^m \frac{\mu}{f_i(x)} \nabla^2 f_i(x) \quad (3.7)$$

These derivatives incorporate the curvature of the quadratic objective and the barrier term.

Furthermore, the Newton step is determined by solving the linear system:

$$H(f_\mu(x)) \Delta x = -\nabla f_\mu(x) \quad (3.8)$$

Once Δx is computed, the decision variables are updated as follows:

$$x \leftarrow x + \alpha \Delta x \quad (3.9)$$

where $\alpha \in (0, 1]$ is a step size chosen through a line search to ensure feasibility and sufficient decrease in the objective [32].

The interior point method iteratively solves the barrier problem for decreasing values of μ . The steps of the algorithm are as follows:

1. **Initialization:** Start with a feasible point x_0 , a large μ , and a stopping tolerance ϵ .
2. **Solve the Barrier Problem:** Use Newton's method to minimize $f_\mu(x)$.
3. **Update the Barrier Parameter:** Scale μ to smaller values (e.g. $\mu \leftarrow 0.1\mu$).
4. **Convergence Check:** Terminate when the solution satisfies the Karush-Kuhn-Tucker (KKT) conditions within the tolerance ϵ .

3.1.2.3 Karush-Kuhn-Tucker Conditions

Before introducing KKT conditions, the Lagrangian of the primal problem given in Equation 3.4 should be expressed.

$$L(x, \lambda, \nu) = f_\mu(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x) \quad (3.10)$$

$L(x, \lambda, \nu)$ is the Lagrangian, the weighted sum of objective and constraint functions. λ_i is Lagrange multiplier associated with $f_i(x) \leq 0$ and ν_i is Lagrange multiplier associated with $h_i(x) = 0$. Moreover, the dual function, $g(\lambda, \nu)$, can be expressed from the Lagrangian $L(x, \lambda, \nu)$ and indicates the lower bound of the objective function. The dual function is defined as

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) \quad (3.11)$$

The operator $\inf_x$ is the greatest lower bound of $L(x, \lambda, \nu)$ over all x . That indicates the dual function $g(\lambda, \nu)$ sets the greatest lower bound that is given by adjusting λ and ν .

The idea of duality originates from the connection between the primal problem and its corresponding dual problem. In the primal problem, the aim is to minimize the objective function $f_0(x)$ subject to constraints. Conversely, the dual problem maximizes the dual function $g(\lambda, \nu)$, which is expressed as:

$$\sup_{\lambda, \nu} g(\lambda, \nu) \quad (3.12)$$

This dual maximization problem ensures that the solution adheres to the feasibility conditions of the Lagrange multipliers $\lambda \geq 0$. The difference between the primal and dual solutions, if any, is referred to as the duality gap.

Duality consists of two concepts: weak duality and strong duality. Weak duality indicates that the value of the dual function provides a lower bound to the primal optimal value:

$$g(\lambda, \nu) \leq f^* \quad (3.13)$$

where f^* is the optimal value of the primal problem.

Strong duality indicates the duality gap is zero:

$$g^* = f^* \quad (3.14)$$

where g^* is the optimal value of the dual problem.

Keeping in mind the points explained above, the solution to the QP problem must satisfy the following KKT conditions, which are necessary for optimality:

1. Primal constraints:

$$f_i(x) \leq 0, \quad i = 1, \dots, m, \quad Ax = b \quad (3.15)$$

2. Dual constraints:

$$\lambda \geq 0 \quad (3.16)$$

3. Complementary slackness:

$$\lambda_i f_i(x) = 0, \quad i = 1, \dots, m \quad (3.17)$$

4. Gradient of the Lagrangian:

$$\nabla L = \nabla f_\mu(x) + \sum_{i=1}^m \lambda_i \nabla f_i(x) + \sum_{i=1}^p \nu_i \nabla h_i(x) = 0 \quad (3.18)$$

If the strong duality holds and x, λ, μ are optimal, they satisfy the KKT conditions. In other words, for any optimization problem with differentiable objective and constraint functions where strong duality is achievable, any pair of primal and dual optimal points will satisfy the KKT conditions [6].

In conclusion, the barrier method provides an effective approach to solving constrained optimization problems by iteratively moving toward the optimal solution

while respecting the constraints. Incorporating a barrier term into the objective function provides that infeasibility is penalized, guiding the solution along a feasible path. As the barrier term reduces over successive iterations, the solution converges to a point where the KKT conditions are satisfied, guaranteeing primal and dual optimality.

3.1.3 Model Predictive Control

Model predictive control is one of the most used optimal control strategies in the literature. The basic concept of MPC is to use a mathematical model to predict system behavior and optimize the prediction for the best decision, the control input at the current time. Therefore, models are central to every form of MPC.

In order to introduce MPC, a mathematical model given by a discrete-time linear system in state-space form is selected.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + Ke(k), \quad k \in \mathbb{N} \\ y(k) &= Cx(k) + Du(k) + e(k) \\ x(0) &= x_0 \end{aligned} \tag{3.19}$$

where x_0 is the initial condition of the system, $x(k) \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^m$ are the system's state and control input at current time k , respectively, $e(k) \in \mathbb{R}^p$ is the innovation at the same time instant, here assumed to be zero-mean white noise, $y(k) \in \mathbb{R}^p$ is the corresponding measured output. State space matrices are that $A \in \mathbb{R}^{n \times n}$ is the state transition matrix, $B \in \mathbb{R}^{n \times m}$ is the input matrix, $K \in \mathbb{R}^{n \times p}$ is the output matrix, $C \in \mathbb{R}^{p \times n}$ is the output matrix and $D \in \mathbb{R}^{p \times m}$ allows a direct coupling between $u(k)$ and $y(k)$. In this thesis and many practical applications, $D = 0$. It is assumed that (A, B) is controllable, (A, C) is observable, and the matrix $A - KC$ is strictly stable; all its eigenvalues lie inside the unit circle.

Over a finite horizon denoted by $N \in \mathbb{N}_{\geq 1}$, MPC searches for the optimal control

input to feed the system by solving the following optimization problem.

$$\begin{aligned}
& \min_{\mathbf{u}(k), \mathbf{y}(k)} \sum_{i=0}^{N-1} \|u(i|k) - u_r\|_R^2 + \|y(i|k) - y_r\|_Q^2 \\
& \text{s.t. } \mathbf{y}(k) = \Phi x(k) + \Gamma \mathbf{u}(k) \\
& (\mathbf{y}(k), \mathbf{u}(k)) \in \mathbb{Y}^N \times \mathbb{U}^N
\end{aligned} \tag{3.20}$$

where

$$\begin{aligned}
\mathbf{u}(k) &= \text{col}(u(0|k), \dots, u(N-1|k)) \\
\mathbf{y}(k) &= \text{col}(y(1|k), \dots, y(N|k))
\end{aligned}$$

denote the N -step-ahead input trajectory computed at time k and the vector of corresponding output predictions $y(i|k)$ at future time $k+i$, for $i = 1, \dots, N$, respectively. $Q \in \mathbb{S}_+^n$ and $R \in \mathbb{S}_{++}^n$ are output and input weighting matrices in the optimization problem. Moreover, as a notation, the norms of the form $\|\cdot\|_A^2$ represent the weighted Euclidean inner product $\|\nu\|_A^2 = \nu^T A \nu$. User-defined input and output references are $y_r \in \mathbb{R}^p$ and $u_r \in \mathbb{R}^n$. In the context of tracking, output reference tracking is a common goal for controller formulation. Conversely, input reference tracking can be incorporated into the objective function. The value of u_r should be associated with the process and the value of y_r . This additional reference term can enhance the controller's performance when the steady-state input value is known [31]. The feasible sets $\mathbb{Y}$ and $\mathbb{U}$ are convex compact sets that contain the desired references in their interior. By iterating the system given in Equation 3.19 and omitting the noise-dependent terms, the following system giving future input-output trajectory can be found in Equation 3.22. Firstly, the iteration steps are given below.

$$\begin{aligned}
y(1|k) &= C(Ax(k) + Bu(0|k)) \\
y(2|k) &= C(Ax(k) + Bu(0|k)) + CBu(1|k) \\
&\vdots \\
y(N|k) &= CA^{N-1}(Ax(k) + Bu(0|k)) + CA^{N-2}Bu(1|k) + \dots + CBu(N-1|k)
\end{aligned} \tag{3.21}$$

Then, the following system of equations is obtained.

$$\begin{aligned} \mathbf{y}(k) &= \Phi x(k) + \Gamma \mathbf{u}(k) \\ \text{where } \Phi &= \begin{bmatrix} CA & CA^2 & \cdots & CA^N \end{bmatrix}^\top \\ \Gamma &= \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} \end{aligned} \quad (3.22)$$

where Φ is the extended observability matrix, Γ is the block Toeplitz matrix. When Equation 3.20 is solved, the optimal control input sequence $\mathbf{u}^*(k)$ is found, then input at the beginning ($u(k) = u^*(0|k)$) is used to feed the system. Through the process, the solution of the constrained optimization problem is repeated. What makes MPC unique is that it directly deals with these limitations. As mentioned before, MPC is a model-based control approach. Therefore, the following section of data-driven methods switches state space matrices with Hankel matrices constructed by the collected empirical data. Indeed, while the advantages of predictive control are kept in terms of constraints and predictions, the subject is based on the system data instead of the system model.

3.1.4 Hankel Matrix

The Hankel matrix is a structured matrix widely used in signal processing, system identification, and control theory. It is constructed from a sequence of data values. This representation captures the system dynamics within the data, making it a powerful tool for analyzing dynamic systems. The definition of the Hankel matrix is given below.

Definition 3.1.1 (The Hankel matrix). The Hankel matrix is constructed using an arbitrary data set d , width depth a , length b and starting index i is denoted as:

$$\mathcal{H}_{[a,b]}^i(d) = \begin{bmatrix} d(i) & d(i+1) & \cdots & d(i+b-1) \\ d(i+1) & d(i+1) & \cdots & d(i+b-1) \\ \vdots & \vdots & & \vdots \\ d(i+a-1) & d(i+a) & \cdots & d(i+a+b-2) \end{bmatrix} \quad (3.23)$$

It is noted that if samples consist of a column vector, $d(i) \in \mathbb{R}^r$ corresponds to $[d_1(i) d_2(i) \dots d_r(i)]^\top$.

3.2 Data-driven Predictive Control

A data-driven strategy is precious in addressing the need for mathematical modeling in MPC. While linear MPC is based on a linearized mathematical model, the data-driven strategy is based on the collected data from the nonlinear system.

In data-driven control, mathematical modeling of the system is replaced with measured inputs and outputs, providing a more realistic representation of the actual system behavior. Since the mathematical model may remain incapable of covering the system's complexities, the data can close that gap.

When it comes to data-driven control, introducing a new parameter, the past horizon $T_{\text{ini}} \in \mathbb{R}_{\geq 1}$, becomes pivotal in specifying the initial conditions at time k . This parameter allows the incorporation of past information, enhancing the predictive capabilities of the control strategy.

To implement this data-driven approach, the following arrays, $\mathbf{u}_{\text{ini}}$ and $\mathbf{y}_{\text{ini}}$ are defined based on measured inputs and outputs.

$$\begin{aligned}\mathbf{u}_{\text{ini}}(k) &= \text{col}(u(k - T_{\text{ini}}), \dots, u(k - 1)) \\ \mathbf{y}_{\text{ini}}(k) &= \text{col}(y(k - T_{\text{ini}} + 1), \dots, y(k))\end{aligned}\tag{3.24}$$

These arrays play a critical role in capturing and employing the data to improve the predictive performance of the controller. The synergy of MPC principles with data-driven methods opens new possibilities for robust and effective control in complex and nonlinear systems. The above-defined variables are generated from the last T_{ini} measurements of inputs and outputs. Moreover, the data-driven approach is supposed to have an informative set of input/output trajectory $\{\bar{\mathbf{u}} \in \mathbb{R}^{m \times T}, \bar{\mathbf{y}} \in \mathbb{R}^{p \times T}\}$. That trajectory is used to construct Hankel matrices, which will be used in data-driven control methods.

Informativity of the input/output trajectory, $\{\bar{\mathbf{u}}, \bar{\mathbf{y}}\}$, is guaranteed by the fact that the available input sequence is persistently exciting of order $T_{\text{ini}} + N + n$ according to

the following definition, where n is the order of the system.

Definition 3.2.1 (Persistence of excitation). The input data trajectory $\bar{\mathbf{u}}$ is persistently exciting of order L if the rank of the Hankel matrix satisfies,

$$\text{rank}(\mathcal{H}_{[L,T]}^0(\bar{\mathbf{u}})) = Lm \quad (3.25)$$

The definition above indicates that T is long enough when replacing L with $(T_{\text{ini}} + N + n)$; that is,

$$T \geq m(T_{\text{ini}} + N + n) \quad (3.26)$$

These input/output trajectories are used to construct the Hankel matrices, which will be used in the following sections:

$$\begin{aligned} U_p &= \mathcal{H}_{[T_{\text{ini}}, T]}^0(\bar{\mathbf{u}}) \\ Y_p &= \mathcal{H}_{[T_{\text{ini}}, T]}^0(\bar{\mathbf{y}}) \\ U_f &= \mathcal{H}_{[N, T]}^{T_{\text{ini}}}(\bar{\mathbf{u}}) \\ Y_f &= \mathcal{H}_{[N, T]}^{T_{\text{ini}}}(\bar{\mathbf{y}}) \end{aligned} \quad (3.27)$$

The matrices U_p and Y_p are used to find a relation between the data and the past online measurements from the system, while U_f and Y_f are used to predict future inputs and outputs.

3.2.1 Subspace Predictive Control

SPC, initially proposed by [15], is a foundational step in evolving data-driven predictive control methods. It was among the earliest methods to integrate subspace identification outcomes [42] into MPC frameworks. SPC estimates prediction matrices derived from input-output data as an indirect data-driven strategy. This estimation process is grounded in a fundamental relationship that links data-driven senses to predictive control objectives, leading to the development of modern data-driven approaches.

$$\mathbf{y}(k) = \Theta \begin{bmatrix} \mathbf{u}_{\text{ini}}(k) \\ \mathbf{y}_{\text{ini}}(k) \\ \mathbf{u}(k) \end{bmatrix} \quad (3.28)$$

where $\Theta = \begin{bmatrix} P_1 & P_2 & \Gamma \end{bmatrix}$

In this framework, future outputs are represented as a linear combination of past inputs/outputs and predicted future inputs, with Γ defined as indicated in Equation 3.22. Employing this connection, the unknown matrix Θ can be estimated by formulating it as a least-squares problem [15, 45]. This approach allows the systematic computation of predictive matrices directly from input-output data, facilitating a data-driven predictive control scheme that aligns with MPC requirements, which makes the method an indirect data-driven method. The problem is defined as,

$$\min_{\Theta} \left\| Y_f - \Theta \begin{bmatrix} U_p \\ Y_p \\ U_f \end{bmatrix} \right\|_{\text{Fro}}^2 \quad (3.29)$$

The solution of the least-squares problem is given as,

$$\Theta^* = \begin{bmatrix} P_1 & P_2 & \hat{\Gamma} \end{bmatrix} = Y_f \begin{bmatrix} U_p \\ Y_p \\ U_f \end{bmatrix}^\dagger \quad (3.30)$$

where $\hat{\Gamma}$ is the data-driven estimation of Γ . Then, the estimation of the Θ constructs the equality constraint of the following optimization problem.

$$\begin{aligned} \min_{\mathbf{u}(k), \mathbf{y}(k)} & \sum_{i=0}^{N-1} \|u(i|k) - u_r\|_R^2 + \|y(i|k) - y_r\|_Q^2 \\ \text{s.t. } & \mathbf{y}(k) = P_1 \mathbf{u}_{\text{ini}}(k) + P_2 \mathbf{y}_{\text{ini}}(k) + \hat{\Gamma} \mathbf{u}(k) \\ & (\mathbf{y}(k), \mathbf{u}(k)) \in \mathbb{Y}^N \times \mathbb{U}^N \end{aligned} \quad (3.31)$$

The solution of the optimization problem in Equation 3.31 gives the optimal input sequence $\mathbf{u}^*(k)$ and resulting the future output sequence $\mathbf{y}(k)$, resulting in an unbiased estimate of the actual system matrices. Tunable parameters are only the weighting matrices Q and R as MPC. In other words, SPC has no additional tuning weighting parameters, unlike the following methods.

3.2.2 Data-enabled Predictive Control

In contrast to estimating the prediction matrices as in SPC, Data-enabled Predictive Control (DeePC) is a direct data-driven control method that predicts future outputs

Algorithm 1 SPC Algorithm

- 1: Obtain U_p, Y_p, U_f, Y_f
 - 2: Obtain Θ^* by Equation 3.30
 - 3: Construct the optimization problem in Equation 3.31
 - 4: **if** $k \leq \max_{\text{iter}}$ **then**
 - 5: Measure $y(k)$ and build $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$.
 - 6: Solve optimization problem in in Equation 3.31 and obtain $\mathbf{u}^*(k)$
 - 7: Apply $u(k) = u^*(0|k)$ to the system.
 - 8: $k = k + 1$
 - 9: **end if**
-

sequence $\mathbf{y}(k)$ directly from Hankel matrix data. This is possible based on the following deterministic result.

Lemma 3.2.1 (Willems' Fundamental Lemma). [43, 50] Consider a system as given in Equation 3.19 for $e(k) = 0$. For $\forall k \in \mathbb{N}$, and let $X_p = \begin{bmatrix} x(0) & \dots & x(T) \end{bmatrix}$. If the measured input sequence of $\bar{\mathbf{u}}$ is persistently exciting of order $T_{\text{ini}} + N + n$, then the following statements hold,

- (i) The matrix $\begin{bmatrix} X_p^\top & U_p^\top & U_f^\top \end{bmatrix}^\top$ has full row rank, i.e.,

$$\text{rank} \left(\begin{bmatrix} X_p \\ U_p \\ U_f \end{bmatrix} \right) = n + m(T_{\text{ini}} + N) \quad (3.32)$$

giving relation

$$\begin{aligned} \text{rank} \left(\begin{bmatrix} U_p \\ U_f \\ Y_f \end{bmatrix} \right) &= \text{rank} \left(\begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \\ \Phi A^{T_{\text{ini}}} & \Phi C_{\text{ini}} & \Gamma \end{bmatrix} \begin{bmatrix} X_p \\ U_p \\ U_f \end{bmatrix} \right) = n + m(T_{\text{ini}} + N) \\ \text{where } C_{\text{ini}} &= \begin{bmatrix} A^{T_{\text{ini}}-1}B & \dots & AB & B \end{bmatrix} \end{aligned} \quad (3.33)$$

Moreover, when Y_p is put instead of Y_f with the relation in Equation 3.22, it

results in

$$\text{rank} \begin{pmatrix} Y_p \\ U_p \\ U_f \end{pmatrix} = \text{rank} \begin{pmatrix} \Phi_{ini} & \Gamma_{ini} & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} X_p \\ U_p \\ U_f \end{pmatrix} = n + m(T_{ini} + N)$$

where $\Phi_{ini} = [CA \quad CA^2 \quad \dots \quad CA^{T_{ini}-1}]^\top$ (3.34)

$$\Gamma_{ini} = \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{T_{ini}-1}B & CA^{T_{ini}-2}B & \dots & CB \end{bmatrix}$$

(ii) Every input/output trajectory $\{\mathbf{u}_{ini}(k), \mathbf{y}_{ini}(k), \mathbf{u}(k), \mathbf{y}(k)\}$ is a trajectory of system given in Equation 3.19 if and only if

$$\begin{bmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{bmatrix} \boldsymbol{\alpha}(k) = \begin{bmatrix} \mathbf{u}_{ini}(k) \\ \mathbf{y}_{ini}(k) \\ \mathbf{u}(k) \\ \mathbf{y}(k) \end{bmatrix} \quad (3.35)$$

for some real vector $\boldsymbol{\alpha}(k) \in \mathbb{R}^\top$.

Proof. (i) Equation 3.36 defines the relationship between the future state block, X_p , and its delayed version, X_{p-n} , through the dynamics propagation matrix A^n and the controllability matrix $\mathcal{C}_n$:

$$X_p = A^n X_{p-n} + \mathcal{C}_n U_{p-n} \quad (3.36)$$

The system's controllability is assumed with $\text{rank}(\mathcal{C}_n)$, where n is the system order, and it is necessary that the input block U_{p-n} has full row rank. Additionally, the combined input matrix $\begin{bmatrix} U_{p-n}^\top & U_p^\top & U_f^\top \end{bmatrix}^\top$ must have a rank equal to $m(T_{ini} + N + n)$ to guarantee the persistent excitation of the input data over the relevant time horizon. Equation 3.37 establishes the rank condition needed for

predictive control:

$$\begin{aligned} \text{rank} \begin{pmatrix} \begin{bmatrix} X_p \\ U_p \\ U_f \end{bmatrix} \end{pmatrix} &= \text{rank} \begin{pmatrix} \begin{bmatrix} A^n & \mathcal{C}_n & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} X_{p-n} \\ U_{p-n} \\ U_p \\ U_f \end{bmatrix} \end{pmatrix} \\ \text{rank} \begin{pmatrix} \begin{bmatrix} X_p \\ U_p \\ U_f \end{bmatrix} \end{pmatrix} &= \text{rank} \begin{pmatrix} \begin{bmatrix} \mathcal{C}_n & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} U_{p-n} \\ U_p \\ U_f \end{bmatrix} \end{pmatrix} \end{aligned} \quad (3.37)$$

The exclusion of the A^n column and X_{p-n} term from the rank analysis is justified, as these components do not add independent information to the rank determination. The term $A^n X_{p-n}$ represents a transformation of past state data and, thus, does not influence the rank condition of the data matrix.

- (ii) Any input/output pair $\{\bar{\mathbf{u}}, \bar{\mathbf{y}}\}$ is a trajectory of the system if there exists a $\hat{x}_0$ such that:

$$\hat{\mathbf{y}} = \Phi \hat{x}_0 + \Gamma \hat{\mathbf{u}} \quad (3.38)$$

A critical requirement is that the matrix formed by $\begin{bmatrix} X_p^\top & U_p^\top & U_f^\top \end{bmatrix}^\top$ must have full row rank to ensure the existence of a vector $\boldsymbol{\alpha}(k)$ that satisfies:

$$\begin{bmatrix} X_p \\ U_p \\ U_f \end{bmatrix} \boldsymbol{\alpha}(k) = \begin{bmatrix} x(k - T_{\text{ini}}) \\ \mathbf{u}_{\text{ini}}(k) \\ \mathbf{u}(k) \end{bmatrix} \quad (3.39)$$

for any combination of inputs and states [43]. This rank condition guarantees the system can map past states and inputs to current and future states and outputs. Equation 3.40 further defines $\boldsymbol{\alpha}(k)$ in terms of U_p , Y_p , U_f , and Y_f as:

$$\begin{aligned} \begin{bmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{bmatrix} \boldsymbol{\alpha}(k) &= \begin{bmatrix} 0 & I & 0 \\ \Phi_{\text{ini}} & \Gamma_{\text{ini}} & 0 \\ 0 & 0 & I \\ \Phi A^{T_{\text{ini}}} & \Phi \mathcal{C}_{\text{ini}} & \Gamma \end{bmatrix} \begin{bmatrix} X_p \\ U_p \\ U_f \end{bmatrix} \boldsymbol{\alpha}(k) \\ &= \begin{bmatrix} 0 & I & 0 \\ \Phi_{\text{ini}} & \Gamma_{\text{ini}} & 0 \\ 0 & 0 & I \\ \Phi A^{T_{\text{ini}}} & \Phi \mathcal{C}_{\text{ini}} & \Gamma \end{bmatrix} \begin{bmatrix} x(k - T_{\text{ini}}) \\ \mathbf{u}_{\text{ini}}(k) \\ \mathbf{u}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{\text{ini}}(k) \\ \mathbf{y}_{\text{ini}}(k) \\ \mathbf{u}(k) \\ \mathbf{y}(k) \end{bmatrix} \end{aligned} \quad (3.40)$$

Moreover, because of the rank condition, it is clear that if $\{\mathbf{u}_{\text{ini}}(k), \mathbf{y}_{\text{ini}}(k)\}$ is a trajectory of the system, $x(k - T_{\text{ini}})$ can be determined. Then, for any $\mathbf{u}(\mathbf{k})$, a unique $\mathbf{y}(\mathbf{k})$ exists in the behavior of the system.

□

Algorithm 2 DeePC Algorithm

- 1: Obtain U_p, Y_p, U_f, Y_f
 - 2: Construct the optimization problem in Equation 3.41
 - 3: Tune λ_α
 - 4: **if** $k \leq \max_{\text{iter}}$ **then**
 - 5: Measure $y(k)$ and build $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$.
 - 6: Solve optimization problem in Equation 3.42 and obtain $\mathbf{u}^*(k)$
 - 7: Apply $u(k) = u^*(0|k)$ to the system.
 - 8: $k = k + 1$
 - 9: **end if**
-

Willems' Lemma provides a deterministic policy for designing data collection, ensuring that the input data sequence is persistently exciting for the unknown system. This condition is necessary for controlling and predicting future outputs directly from data, as DeePC utilizes. However, Willems' Lemma falls short in noisy environments because its deterministic nature does not account for corrupted data. When noise affects the data, the matrix $\begin{bmatrix} U_p^\top & U_f^\top & Y_f^\top \end{bmatrix}^\top$ loses its full row rank, potentially leading to inaccurate predictions by interpreting any input/output data pair as a system trajectory.

To address this limitation, DeePC includes a regularization term that encourages $\alpha(k)$ to remain small, resulting in the following optimization problem. This regularization helps to mitigate the impact of noise, enhancing the robustness and accuracy of the predictive model.

$$\begin{aligned}
& \min_{\substack{\boldsymbol{\alpha}(k), \sigma_y(k) \\ \mathbf{u}(k), \mathbf{y}(k)}}} \sum_{i=0}^{N-1} \|u(i|k) - u_r\|_R^2 + \|y(i|k) - y_r\|_Q^2 + \lambda_\alpha \|\boldsymbol{\alpha}(k)\|_2^2 + \lambda_\sigma \|\sigma_y(k)\|_2^2 \\
& \text{s.t.} \quad \begin{bmatrix} U_p \\ Y_p \\ U_f \\ Y_f \end{bmatrix} \boldsymbol{\alpha}(k) = \begin{bmatrix} \mathbf{u}_{\text{ini}}(k) \\ \mathbf{y}_{\text{ini}}(k) + \sigma_y(k) \\ \mathbf{u}(k) \\ \mathbf{y}(k) \end{bmatrix} \\
& (\mathbf{y}(k), \mathbf{u}(k)) \in \mathbb{Y}^N \times \mathbb{U}^N
\end{aligned} \tag{3.41}$$

where $\|\boldsymbol{\alpha}(k)\|_2^2$ is a regularization term that defines the relation between a pre-defined dataset and predicted states and inputs. Moreover, $\sigma_y(k)$ is a slack variable introduced to prevent the problem from turning infeasible.

The solution QP problem in Equation 3.42 gives the optimal input sequence $\mathbf{u}^*(k)$ and resulting the future output sequence $\mathbf{y}(k)$.

3.2.3 γ -data-driven Predictive Control

γ -DDPC [7, 8] combines SPC and DeePC. γ -DDPC relies on a preprocessing operation on the Hankel matrices to enhance prediction accuracy, while it does not need an estimate of the prediction matrices as SPC [44]. Before introducing that alternative data-driven control technique, the following variables classing the past data, Z_p , and the initial condition characterizing, $\mathbf{z}_{\text{ini}}(k)$ are defined.

$$Z_p = \begin{bmatrix} U_p \\ Y_p \end{bmatrix} \tag{3.42}$$

$$\mathbf{z}_{\text{ini}}(k) = \begin{bmatrix} \mathbf{u}_{\text{ini}}(k) \\ \mathbf{y}_{\text{ini}}(k) \end{bmatrix} \tag{3.43}$$

Furthermore, the following LQ decomposition of the Hankel matrices featured in the predictor of data-driven predictive control is defined below.

$$\begin{bmatrix} Z_p \\ U_f \\ Y_f \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{3.44}$$

where the matrices $\{L_{ii}\}_{i=1}^3$ are all non-singular (under the stochastic conditions of persistence of excitation [7]). Also, Q_i have orthonormal rows, i.e., $Q_i Q_i^T = I$, for $i = 1, \dots, 3$, $Q_i Q_j^T = 0$, $i \neq j$.

The last decomposition forms the essential step of the γ -DDPC scheme. Right-multiplying both sides of Equation 3.44 by $\alpha(k)$ gives

$$\begin{bmatrix} \mathbf{z}_{\text{ini}}(k) \\ \mathbf{u}_{\text{ini}}(k) \\ \mathbf{y}_{\text{ini}}(k) \end{bmatrix} = \begin{bmatrix} Z_p \\ U_f \\ Y_f \end{bmatrix} \alpha(k) = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} \gamma_1(k) \\ \gamma_2(k) \\ \gamma_3(k) \end{bmatrix} \quad (3.45)$$

provides to replace the predictor matrices (U_p, Y_p, U_f, Y_f) in DeePC with a new one in the new variables $\gamma_i(k) = Q_i \alpha_k$, for $i = 1, 2, 3$.

It is noted that the dimension of these new optimization variables does not depend on the length of collected data, T , anymore, making the control problem computationally governable even when large data are used. Moreover, the number of optimization variables can be reduced by pre-calculating $\gamma_1(k)$. That is, because $\mathbf{z}_{\text{ini}}(k)$ is fixed, $\gamma_1(k)$ can be found by

$$\gamma_1^*(k) = L_{11}^\dagger \mathbf{z}_{\text{ini}}(k) \quad (3.46)$$

Based on both the LQ-decomposition and the pre-computation of $\gamma_1(k)$, the γ -DDPC problem thus reduces to

$$\begin{aligned} & \min_{\substack{\gamma_2(k), \gamma_3(k) \\ \mathbf{u}(k), \mathbf{y}(k)}} \sum_{k=0}^{N-1} \|u(i|k) - u_r\|_R^2 + \|y(i|k) - y_r\|_Q^2 + \beta_2 \|\gamma_2(k)\|_2^2 + \beta_3 \|\gamma_3(k)\|_2^2 \\ & \text{s.t.} \quad \begin{bmatrix} \mathbf{u}(k) \\ \mathbf{y}(k) \end{bmatrix} = \begin{bmatrix} L_{22} & 0 \\ L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} \gamma_2(k) \\ \gamma_3(k) \end{bmatrix} + \begin{bmatrix} L_{21} \\ L_{31} \end{bmatrix} L_{11}^\dagger \mathbf{z}_{\text{ini}}(k) \\ & (\mathbf{y}(k), \mathbf{u}(k)) \in \mathbb{Y}^N \times \mathbb{U}^N \end{aligned} \quad (3.47)$$

where $\beta = \begin{bmatrix} \beta_2 & \beta_3 \end{bmatrix}^\top$ is a regularization term that can trade-off between bias and variance as in DeePC.

Algorithm 3 γ -DDPC Algorithm

- 1: Obtain U_p, Y_p, U_f, Y_f
 - 2: Compute and extract $L_{11}, L_{21}, L_{22}, L_{31}, L_{32}, L_{33}$ from Equation 3.44 by LQ-decomposition
 - 3: Calculate $\begin{bmatrix} L_{21} \\ L_{31} \end{bmatrix} L_{11}^\dagger$
 - 4: Construct the optimization problem in Equation 3.48
 - 5: Tune β_2, β_3
 - 6: **if** $k \leq \max_{\text{iter}}$ **then**
 - 7: Measure $y(k)$ and build $\mathbf{u}_{\text{ini}}(k)$ and $\mathbf{y}_{\text{ini}}(k)$ to get $\mathbf{z}_{\text{ini}}(k)$.
 - 8: Solve optimization problem in Equation 3.42 and obtain $\mathbf{u}^*(k)$
 - 9: Apply $u(k) = u^*(0|k)$ to the system.
 - 10: $k = k + 1$
 - 11: **end if**
-

3.2.4 Summary

Table 3.1: SPC summary

Parameter	Value
Problem size	$N(m + p)$
Memory requirement	$(Np)(T_{\text{ini}}(m + p) + Nm)$
Tunable bias-variance trade-off	No
Suitable for large data sets	Yes
Additional tuning weights	No

Tables 3.1, 3.2, and 3.3 present a comprehensive overview of the properties of various DDPC methods. It is important to note that the DeePC method is not suitable for handling large data sets, which limits its applicability in scenarios with extensive data requirements. In contrast, both the SPC and γ -DDPC methods are better equipped to manage larger datasets. This limitation of DeePC arises specifically from challenges associated with problem size.

Furthermore, the SPC algorithm stands out because it does not require additional

Table 3.2: DeePC summary

Parameter	Value
Problem size	$N(m + p) + T + T_{\text{ini}}p$
Memory requirement	$(T_{\text{ini}} + N)(m + p)T$
Tunable bias-variance trade-off	Yes
Suitable for large data sets	No
Additional tuning weights	$\lambda_\alpha, \lambda_\sigma$

Table 3.3: γ -DDPC summary

Parameter	Value
Problem size	$2N(m + p)$
Memory requirement	$N(m + p)^2(N + T_{\text{ini}})$
Tunable bias-variance trade-off	Yes
Suitable for large data sets	Yes
Additional tuning weights	β_2, β_3

tuning weights, unlike the other methods, which do necessitate such adjustments. These extra parameters contribute to a critical bias-variance trade-off, allowing the designer to emphasize either control tasks or reference tracking as necessary.

In conclusion, the hybrid method γ -DDPC is highly recommended for addressing complex problems due to its ability to efficiently tackle smaller problem sizes and the presence of additional tuning weights that enhance its flexibility and performance.

3.2.5 Implementation Guide

Data-driven predictive control methods are fundamentally based on the formulation and resolution of optimization problems, which often involve the careful tuning of various parameters. This tuning process is recognized as critical, as significant dependence on the chosen parameters can be noted regarding the performance of these control methods. Therefore, effective implementation is necessitated by a deep under-

standing of the underlying engineering principles and insights related to the specific system being controlled.

This section provides detailed recommendations to guide practitioners in the effective application of a data-driven approach within complex systems. These guidelines will cover aspects such as data collection and parameter tuning strategies, ensuring robust and efficient implementation. By following these recommendations, the reliability and accuracy of predictive control strategies in complex environments can be enhanced.

3.2.5.1 Data Collection

As outlined in Definition 3.2.1, it is essential that the input signal employed in the construction of the Hankel matrices, as detailed in Definition 3.1.1, is persistently exciting of sufficient order. This means that the input signal should contain a wide range of frequencies and variations to effectively characterize the system being analyzed.

Two primary methods can be utilized to gather this necessary data: one approach involves injecting a carefully designed random input sequence directly into the system, while the other entails executing a manual flight experiment, in which a human operator actively assumes the role of the stabilizing controller [12].

However, since the benchmarking process is carried out through computer-based simulations, injecting a random input sequence is chosen as the most suitable option. This ensures that the data collected is both repeatable and representative of the system's dynamic behavior under diverse conditions.

Due to the unstable behavior of the quadrotor in an open-loop system [35], it is essential to add random noise to the input signal generated by the precontroller. Additionally, it is advisable to increase the amount of noise added to the input signal as much as possible.

While a noisy input signal can effectively promote persistent excitation, it is important to note that measurements may also be noisy. This adds realism to the simulation and helps prevent overfitting in the data.

3.2.5.2 Tuning

This section provides recommendations for practitioners on optimizing data-driven predictive settings.

1. **Tuning of SPC:** After ensuring that there is sufficient excitation and an adequate length of data, the SPC algorithm can be utilized by adjusting the Q and R weighting matrices. Identifying the appropriate values for these matrices establishes the baseline settings for other controllers, including DeePC and γ -DDPC. Additionally, while tuning the SPC, the data length can be as long as possible due to its low computing time requirements.
2. **Tuning of DeePC Parameters:** Once the Q and R weighting matrices are defined, a more in-depth analysis of the DeePC method can be conducted. This algorithm requires tuning two additional parameters to optimize its performance. The first parameter, λ_α , is responsible for regulating the influence of the regularization term, which helps prevent overfitting by penalizing overly complex models. The second parameter, λ_σ , is associated with the slack variable, which allows for some flexibility in the model to accommodate potential deviations from the desired performance. Proper adjustment of these parameters is crucial for achieving the best results with the DeePC method.

According to [25], the DeePC cost function can be reformulated as a Tikhonov regularization problem, as follows:

$$\begin{aligned}
 & \min_{\lambda_\alpha, \boldsymbol{\alpha}(k)} \|A\boldsymbol{\alpha}(k) - b\| + \lambda_\alpha \|\boldsymbol{\alpha}(k)\|_2^2 \\
 & \text{where } A = \text{diag}\left(\bar{Q}^{\frac{1}{2}}, \bar{R}^{\frac{1}{2}}\right) \begin{bmatrix} Y_f \\ U_f \end{bmatrix} \\
 & \quad b = \text{diag}\left(\bar{Q}^{\frac{1}{2}}, \bar{R}^{\frac{1}{2}}\right) \begin{bmatrix} \mathbf{r}_y \\ \mathbf{r}_y \end{bmatrix} \\
 & \quad \bar{Q} = Q \otimes I_N, \quad \bar{R} = R \otimes I_N
 \end{aligned} \tag{3.48}$$

This allows practitioners to utilize existing tuning strategies for DeePC. The regularization weight, λ_α , can be adjusted using the Hanke-Raus rule, which

involves evaluating the following score function of λ_α :

$$\psi(\lambda_\alpha) = \left\langle b, \lambda_\alpha^2 \left(AA^\top + \lambda_\alpha I \right)^{-3} b \right\rangle^{\frac{1}{2}} \quad (3.49)$$

Additionally, $\mathbf{r}_y$ and $\mathbf{r}_u$ represent the reference values for the output and input vectors, respectively. It is important to note that if these values are both set to zero, the score function $\psi(\lambda_\alpha)$ becomes undefined because $\lambda_\alpha = 0$ minimizes the cost function. Therefore, this approach is valid as long as both $\mathbf{r}_y$ and $\mathbf{r}_u$ are not zero simultaneously.

As the parameter λ_α decreases toward zero, the method increasingly overlooks the primary control objective. Conversely, as λ_α approaches infinity, the approach may yield results that are not only ineffective but could also lead to unintended consequences. Additionally, as previously mentioned, λ_y should be very high to restrict the estimator's freedom [44].

3. **Tuning of γ -DDPC Parameters:** Unlike DeePC, the γ -DDPC method requires the user to adjust two parameters: β_2 and β_3 . Ideally, these parameters should be set to $\beta_2 = 0$ and $\beta_3 \rightarrow \infty$ if an infinite dataset is available [7]. However, selecting the appropriate values for these parameters can be crucial when working with a limited amount of data. Since the data available for designing the controller is typically finite, we will summarize potential tuning strategies for β_2 and β_3 to address this practical limitation.

As tuning parameters of the DeePC approach, the Hanke-Raus rule can be used for both γ -DDPC tuning parameters [44]. Firstly, the score function is introduced for β_2 as follows:

$$\begin{aligned} \psi(\beta_2) &= \left\langle b(\gamma_1^*), \beta_2^2 \left(AA^\top + \beta_2 I \right)^{-3} b(\gamma_1^*) \right\rangle^{\frac{1}{2}} \\ \text{where } A &= \text{diag} \left(\bar{Q}^{\frac{1}{2}}, \bar{R}^{\frac{1}{2}} \right) \begin{bmatrix} L_{32} \\ L_{22} \end{bmatrix} \\ b(\gamma_1^*) &= \text{diag} \left(\bar{Q}^{\frac{1}{2}}, \bar{R}^{\frac{1}{2}} \right) \begin{bmatrix} \mathbf{r}_y - L_{31}\gamma_1^* \\ \mathbf{r}_u - L_{21}\gamma_1^* \end{bmatrix} \end{aligned} \quad (3.50)$$

It is noted that γ_1^* is defined in Equation 3.46. For the parameter β_3 , the same

procedure can be followed.

$$\begin{aligned}\psi(\beta_3) &= \left\langle b(\gamma_1^*, \gamma_2), \beta_3^2 \left(AA^\top + \beta_3 I \right)^{-3} b(\gamma_1^*, \gamma_2) \right\rangle^{\frac{1}{2}} \\ \text{where } A &= \bar{Q}^{\frac{1}{2}} L_{33} \\ b(\gamma_1^*, \gamma_2) &= \bar{Q}^{\frac{1}{2}} (\mathbf{r}_y - L_{31} \gamma_1^* - L_{31} \gamma_2)\end{aligned}\tag{3.51}$$

As a result of the recommendations above, data-driven predictive control methods can be employed. After having sufficient data from the system, the Q and R weighting matrices are tuned first. Other fixed parameters consist of N and T_{ini} . N is selected as sufficiently large such that the chosen value can provide stabilizing. T_{ini} is recommended such that it should be larger than the order of the system, n .



CHAPTER 4

SIMULATION RESULTS

4.1 Quadrotor Parameters

Fixed physical and inertial parameters define the quadrotor model in this study, which is held constant across all simulations to allow for consistent evaluation of control methods. Table 4.1 lists the parameters used, as taken from [20].

Table 4.1: Fixed model parameters for the quadrotor system

Parameter	Value
Mass (m) [kg]	2
Arm length (l) [m]	0.25
Moment of inertia (x -axis) (I_{xx}) [kg·m ²]	1.2
Moment of inertia (y -axis) (I_{yy}) [kg·m ²]	1.2
Moment of inertia (z -axis) (I_{zz}) [kg·m ²]	2.3
Gravitational acceleration (g) [m/s ²]	9.81
Thrust factor (b) [N/(rad/s) ²]	1
Drag factor (d) [N·m/(rad/s) ²]	0.2
Maximum motor speed ($\omega_{\max}$) [rad/s]	3.16

These parameters are used in the equations of motion presented in Chapter 2, forming the basis for simulations.

4.2 Data Collection

The system data has to be collected, with the use of which Hankel matrices needed for data-driven methods will be constructed. Input and output trajectories are generated using the mathematical model that was introduced in Chapter 2. Since instabilities occur on the quadrotor whenever it is in open-loop operation, an essential strategy is the implementation of a model-based pre-controller for gathering this data in a stable way. In this regard, MPC is utilized to get the required amount of data using a mathematical model of an underlying system so as to obtain the control strategy. In turn, the MPC itself is utilized so that there is a sound basis of stability during data collection, enabling the system to stay within its operational bounds while collecting useful trajectory data.. The specific MPC configuration used for this purpose is detailed in Table 4.2, highlighting the key settings and parameters involved in the process.

Table 4.2: MPC settings to generate data

Parameter	Value
Sampling rate (T_s) [s]	0.1
Prediction horizon (N) [-]	30
State weight matrix (Q)	$\begin{bmatrix} I_3 & 0_3 & \dots & 0_3 \\ 0_3 & 500 \otimes I_3 & \ddots & \vdots \\ \vdots & \ddots & I_3 & 0_3 \\ 0_3 & \dots & 0_3 & 500 \otimes I_3 \end{bmatrix}$
Input weight matrix (R)	$10 \times I_4$

The MPC is implemented to enable the quadrotor to follow a predefined trajectory accurately. The subsequent expressions detail this trajectory.

$$\begin{aligned}
 x &= p \sin(2\pi\omega_d) \\
 y &= p \sin(2\pi\omega_d) \\
 z &= p \sin(2\pi\omega_d)
 \end{aligned} \tag{4.1}$$

Table 4.3 indicates the quadrotor should track trajectory parameters. Amplitude and frequency are selected so that the quadrotor can follow the reference and the data can cover sufficient information for data-driven methods.

Table 4.3: Trajectory parameters to generate the data

Parameter	Value
Amplitude of sinusoidal position reference (p) [m]	10
Frequency of sinusoidal position reference (ω_d) [Hz]	0.01

The simulation of the quadrotor controlled by the MPC proceeded for 254.9 seconds, corresponding to 2550 data points for each output. The length of the collected data follows the condition given in Equation 3.26. Furthermore, the parameter of T is used as 2500, whereas the length of the collected data is 2550. Since the Hankel matrix structure in Equation 3.23 needs an offset, the sum of $T_{\text{ini}} = 20$ and $N = 30$, which are used in the simulation of the data-driven methods should be added into T . Throughout the data collection phase and closing the loop, it is assumed that only the system outputs are corrupted by noise, with the measurement noise. The noise is specified as zero-mean Gaussian white noise.

Furthermore, the control input is intentionally subjected to Gaussian white noise to enhance the persistency of excitation, which is crucial for reliable system performance. The specific standard deviations of this noise are detailed in Table 4.4. It is worth mentioning that while measurement noise on the output signals plays a significant role in various other simulations—particularly those involving both data-driven and model-based benchmarks—noise is deliberately omitted from the control inputs. This exclusion is strategic, as it ensures that the control inputs maintain their effectiveness in promoting the necessary persistency of excitation.

Table 4.4: Standard deviation of the noises

Parameter	Value
Std of the noise on position measurements (σ_p) [m]	0.01
Std of the noise on Euler angle measurements (σ_e) [$^\circ$]	0.5
Std of the noise on velocity measurements (σ_v) [rad]	0.01
Std of the noise on Euler angle rates measurements (σ_ω) [$^\circ/\text{s}$]	0.5
Std of the noise on control inputs (σ_ω) [rad^2/s^2]	1.5

Figures 4.1, 4.2, 4.3, and 4.4 present a comprehensive visualization of the output trajectories derived from 100 randomly generated datasets. These figures illuminate the time-dependent variation of various system variables, showcasing how each state responds to the induced disturbances. The data clearly demonstrates that the controller adeptly manages these random disturbances, ensuring stable and controlled flight operations throughout the experiments. This consistent performance suggests that the collected data accurately captures the intricate dynamics of the quadrotor, providing valuable insights for a detailed analysis of its behavior across a spectrum of conditions.

The extreme values associated with each state serve as essential constraints that inform the design and implementation of data-driven predictive controllers. These constraints define the operational boundaries within which the controllers must function, ensuring optimal performance and stability in various scenarios.

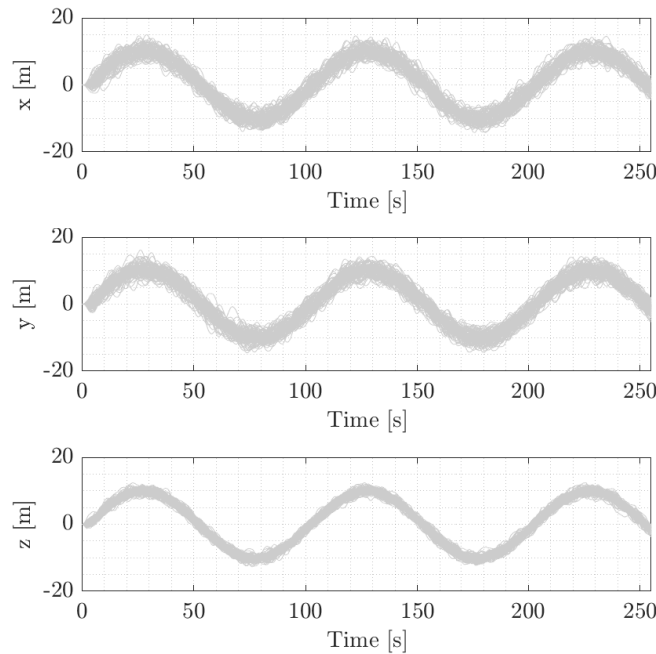


Figure 4.1: Position trajectories for 100 randomly generated datasets

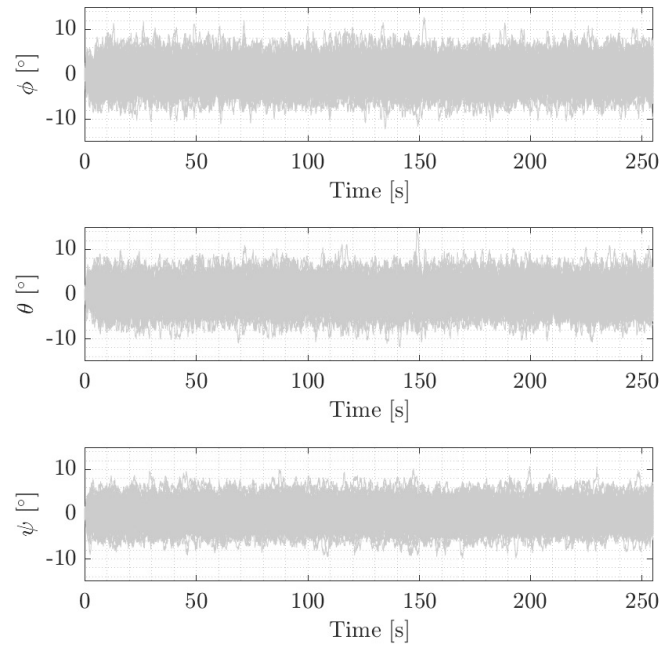


Figure 4.2: Euler angles trajectories for 100 randomly generated datasets

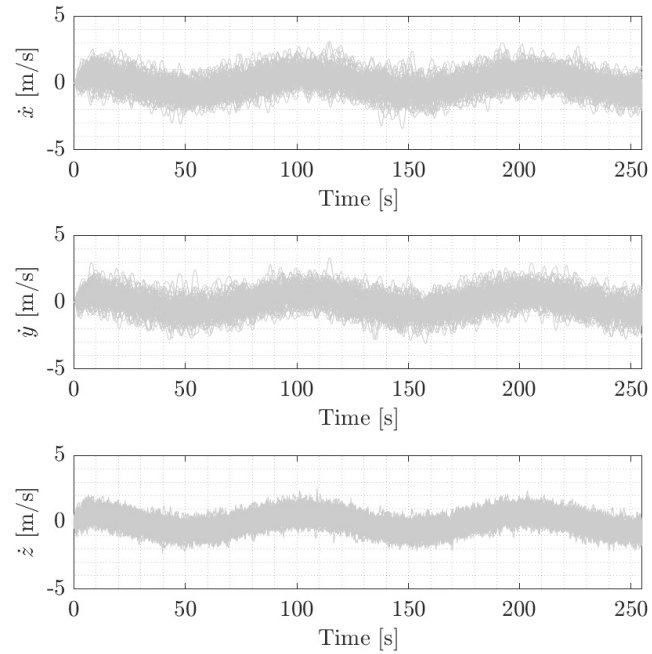


Figure 4.3: Velocity trajectories for 100 randomly generated datasets

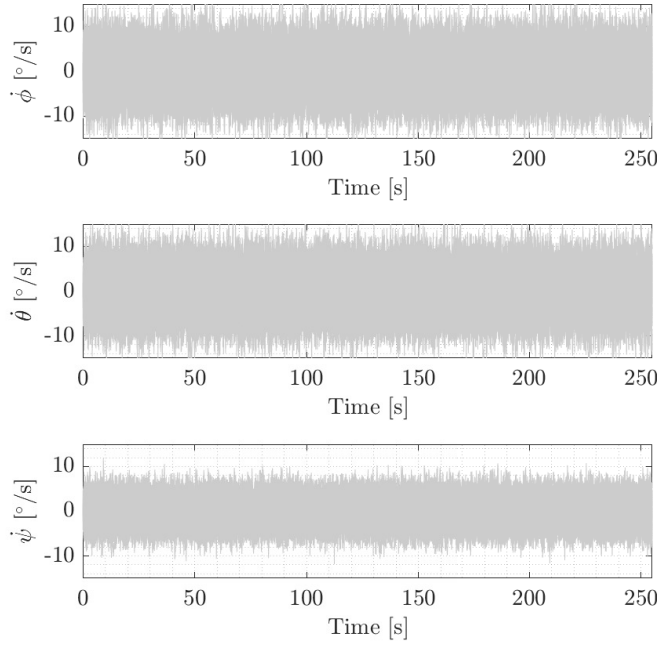


Figure 4.4: Euler angle rates trajectories for 100 randomly generated datasets

Figure 4.5 presents the input trajectories for 100 randomly generated datasets, illustrating how the control inputs for each rotor are changed over time. Various input variations are shown in each dataset, providing insight into rotor behavior under different circumstances. The square of the angular velocities of the rotors is constrained to fall between 0 and 10, establishing the necessary limits for the input values. These constraints are incorporated into the MPC to ensure that the rotors are operated safely and effectively within the defined limits. This approach allows for the management and adjustment of the control inputs to be carried out, enabling optimal performance of the rotors in noisy conditions.

Added noise in control inputs ensures the persistency of the excitation, which is a very important fact in data-driven control, widening the range of dynamic responses. Persistency of excitation gives comprehensive data that enriches the system's information and allows us to precisely model the behavior of the quadrotor under various operational conditions. The control inputs remain within the specified limits; thus, the controller operates normally without overstepping from the values that can pro-

vide stability in the operation of the quadrotor. These input trajectories, therefore, contribute to the reliability of the data collected for analysis of various conditions pertinent to control effectiveness and stability.

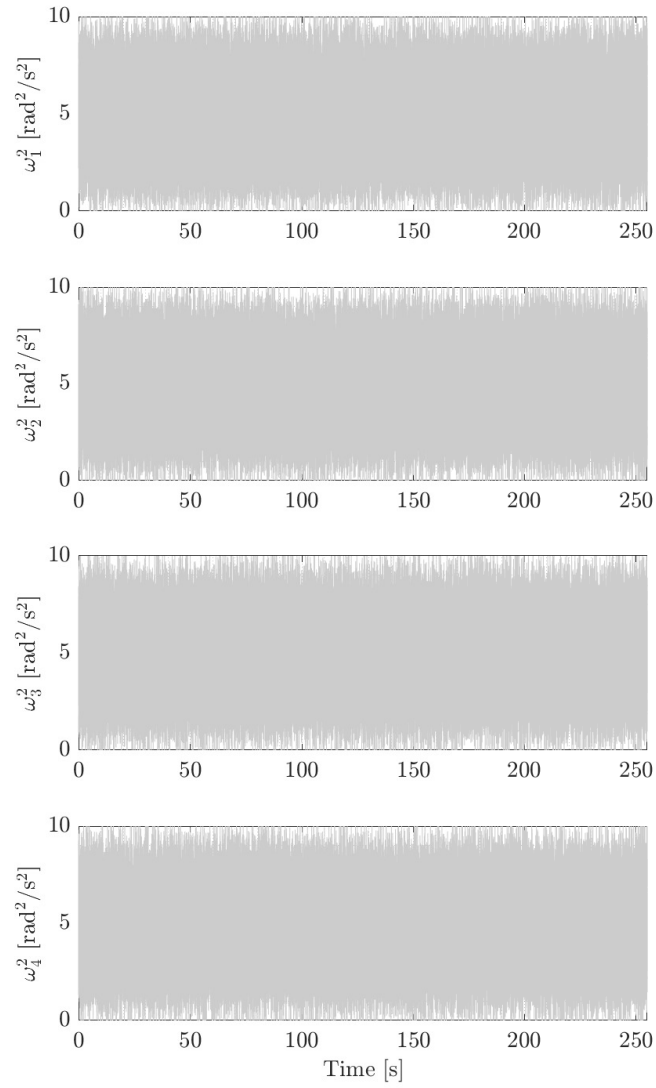


Figure 4.5: Input trajectories for 100 randomly generated datasets

4.3 Choice of Prediction Horizon and Sampling Rate

For the DDPC methods, the prediction horizon and sampling rate are the primary parameters in the optimization problem. Selecting these values is not trivial in this context since the problem is highly nonlinear and not noise-free. Additionally, there are constraints on the states and inputs. Thus, an infinite prediction horizon corresponding to an LQR-like method is not feasible. Therefore, a practical approach should be followed to determine these values. This approach examines the impact of these parameters on position tracking performance, quantified by the position tracking error $\int_{t=T_{ini}}^{t=t_f} \|\zeta(t) - \zeta_r(t)\|_2^2 dt$, where ζ represents the position variables of (x, y, z) and ζ_r denotes their reference values.

Table 4.5: Error Comparison for Different N and T_s Values

T_s [s] \ N [-]	0.005	0.01	0.02	0.05	0.1	0.2
30	-	-	24.975	13.419	9.5937	-
50	-	41.331	21.431	13.427	9.5949	-
100	-	30.848	21.227	13.454	-	-

Table 4.5 presents error calculations for each N selection with varying T_s . Firstly, the error values of some selections are written as (-) in order to identify them as improper selections. Improper selections mean that the optimization process is not completed, and the simulation is interrupted. It is because the NT_s multiplication is either too small to capture dynamics or too high to withstand noises in measurements.

Furthermore, they are related to the state and input weight matrices selection, whose values will be introduced in the following pages. The matrices Q and R determine the settling and rise time metrics of the responses. So, the result of this investigation depends also on Q and R matrices.

Based on the results presented in Table 4.5, the selection of T_s and N significantly impacts the error values. As seen in the table, smaller sampling times ($T_s = 0.005$ and 0.02) led to instability or excessive computational requirements, as indicated by the missing error values. Conversely, for $T_s = 0.1$, the error values are consistently

low across different horizon lengths, particularly for $N = 30$ and $N = 50$, where the errors are 9.5937 and 9.5949, respectively. Although increasing the prediction horizon can slightly reduce the error, as observed in the cases of $N = 50$ and $N = 100$, it comes at the cost of higher computational complexity.

To balance accuracy and computational efficiency, selecting $T_s = 0.1$ and $N = 30$ is a reasonable choice, as it provides one of the lowest error values while minimizing computational demand. This selection ensures stable simulation performance and adequate dynamic capture without excessive processing time.

4.4 Choice of Regularization Parameters

In data-driven control techniques, selecting appropriate regularization parameters is essential to optimize model performance and prevent overfitting. Unlike SPC, which does not require tuning such parameters, DeePC and γ -DDPC contain regularization terms in their optimization formulations.

This section investigates the impact of these parameters on position tracking performance, quantified by the position tracking error $\int_{t=T_{\text{ini}}}^{t=t_f} \|\zeta(t) - \zeta_r(t)\|_2^2 dt$, where ζ are position variables of (x, y, z) and ζ_r defines their reference values.

Through grid analyses presented in Figures 4.6 and 4.7, it is explored that the effect of different combinations of parameters $\lambda_\alpha, \lambda_\sigma$ for DeePC and β_2, β_3 for γ -DDPC on the tracking error. The heatmaps point to the regions of these parameters, which have minimum tracking errors and provide insight into the tuning of parameters for better control accuracy. This analysis gives importance to the selection of parameters in order to reach better robustness and applicability of the data-driven control methods such as DeePC and γ -DDPC.

Figure 4.6 shows a heatmap that visualizes the position tracking error for different combinations of regularization parameters λ_α and λ_σ in DeePC. The map ranges from low tracking errors (white and light yellow) to high tracking errors (red and black). The white and light yellow regions are the optimal sets of regions where the tracking position error is minimal, which means that combinations of parameters enhance

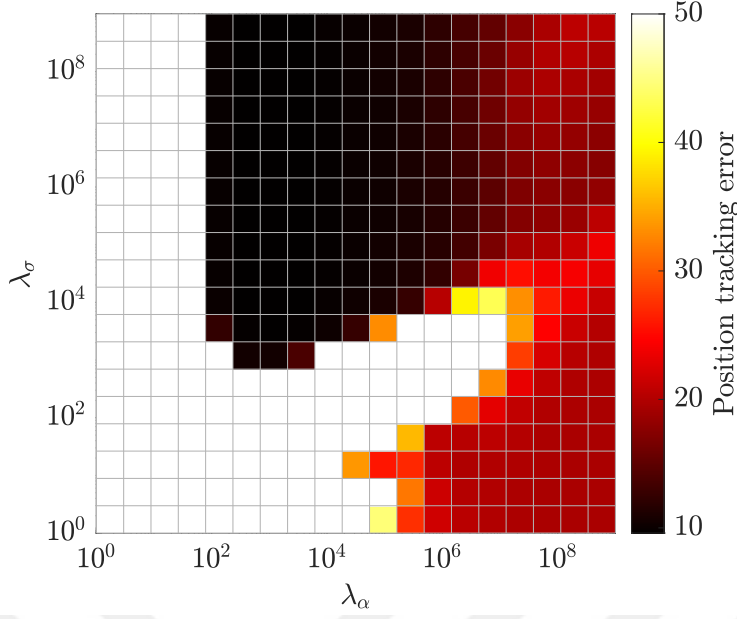


Figure 4.6: Position tracking error with various λ_α and λ_σ

control performance. On the contrary, black and dark red regions show high tracking errors, signifying the settings of parameters that lead to poor tracking performance because of over-regularization or under-regularization.

Upon examining the map, a suitable combination of parameters for effective performance falls within the white and light yellow zones. Selecting $\lambda_\alpha = 10^3$ and $\lambda_\sigma = 10^8$ is justified as these values lie within an area showing minimal tracking error. The selected λ_α value provides sufficient regularization to control model complexity and prevent underfitting, ensuring the control model maintains adaptability to system dynamics. The higher value of λ_σ offers robust noise and disturbance attenuation, helping to stabilize the control output and maintain low variability in tracking performance. This combination balances adaptability and robustness, leading to more reliable and accurate position tracking under various operating conditions.

Figure 4.7 presents a heatmap illustrating the position tracking error across different combinations of regularization parameters β_2 and β_3 for γ -DDPC. The chosen parameters, $\beta_2 = 0$ and $\beta_3 = 2 \times 10^3$, all within a region on the map that aligns with low tracking error, as shown by the light yellow areas. Setting $\beta_2 = 0$ implies that this specific regularization term is not contributing to the cost function, which could indi-

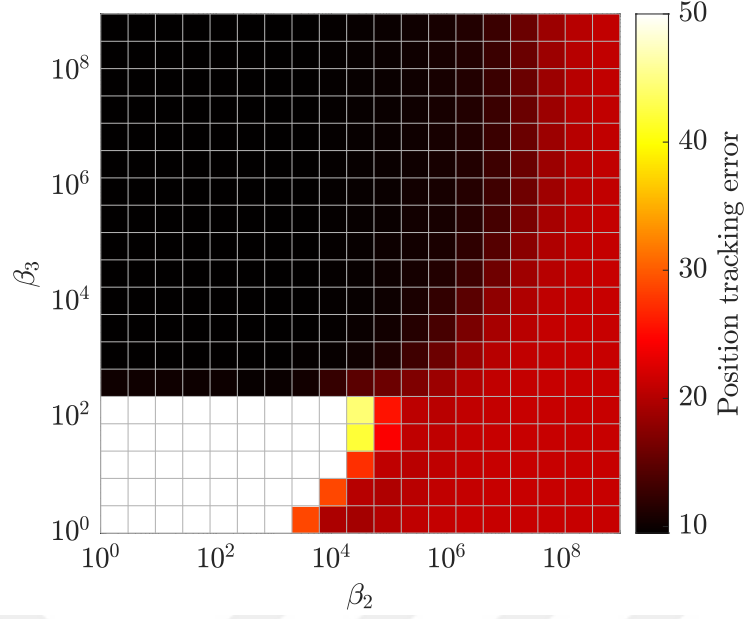


Figure 4.7: Position tracking error with various β_2 and β_3

cate that the parameter is not needed or that including it may not significantly improve the performance for this specific case. The value of $\beta_3 = 2 \times 10^3$ is a moderately regularized value that keeps the balance in fitting the data on one hand while keeping it far enough from overfitting on the other. Justification: this choice will maintain accurate position tracking against noise or disturbances right from model instability effects. The map confirms that this combination is suitable for minimizing tracking errors and achieving reliable performance with γ -DDPC.

Following a thorough analysis of the grid map, the parameters necessary for the DeePC and γ -DDPC methods have been identified. Two distinct approaches can be used to define these parameters: analytical expressions and grid map searches.

Analytical expressions provide a mathematical framework to guide parameter selection, offering valuable insights into the rationale behind the choices made. However, this method may not always fully capture the complexity of the parameter space.

In contrast, the grid map search method involves systematically exploring a range of parameter values across the grid. This approach allows for a more comprehensive evaluation of potential outcomes, often resulting in more optimal parameter configu-

rations. By utilizing the grid map search, the most effective settings for the methods can be identified, ensuring better performance in practical applications.

4.5 Parameter Settings of Data-driven Predictive Control Methods

As discussed in Chapter 3, the weighting matrices used in optimization problems, Q and R , can be adjusted in the SPC problem. This adjustment is possible because the SPC method relies solely on these tuning variables. Once the tuning procedure is complete, these variables are applied across all DDPC methods.

Table 4.6: SPC settings

Parameter	Value
Sampling rate (T_s) [s]	0.1
Prediction horizon (N) [-]	30
Past horizon (T_{ini}) [-]	20
Data length (T) [-]	2500
State weight matrix (Q)	$\begin{bmatrix} 100 & 0 & \dots & \dots & \dots & 0 \\ 0 & 100 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 20 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & \ddots & \vdots \\ \vdots & \dots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 1 \end{bmatrix}$
Input weight matrix (R)	$10^{-2} \times I_4$
Input lower BC (u_l)	$0 \times 1_4$
Input upper BC (u_u)	$8 \times 1_4$
Output lower BC (y_l)	$-[10 \times 1_3 \ 0.09 \times 1_3 \ 2 \times 1_3 \ 0.18 \times 1_3 \ 0.09]^\top$
Output upper BC (y_u)	$[10 \times 1_3 \ 0.09 \times 1_3 \ 2 \times 1_3 \ 0.18 \times 1_3 \ 0.09]^\top$

Table 4.6 outlines the settings used for the SPC method. The Q and R matrices serve as the foundation for the DDPC methods, alongside other variables such as T_{ini} and N . Additionally, lower and upper-bound constraints are consistent across all DDPC

methods.

The selection of bounds for inputs and outputs is based on the data collection process. To ensure the solution is feasible, the optimal input must remain within the bounds established by the collected data. Therefore, it is important to thoroughly investigate the data when specifying these values.

From Section 4.4, the parameters to be tuned are selected. Tables 4.7 and 4.8 indicate the settings for DeePC γ -DDPC.

Table 4.7: DeePC settings

Parameter	Value
Sampling rate (T_s) [s]	0.1
Prediction horizon (N) [-]	30
Past horizon (T_{ini}) [-]	20
Data length (T) [-]	1200
State weight matrix (Q)	$\begin{bmatrix} 100 & 0 & \dots & \dots & \dots & 0 \\ 0 & 100 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 20 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & \ddots & \vdots \\ \vdots & \dots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 1 \end{bmatrix}$
Input weight matrix (R)	$10^{-2} \times I_4$
Regular. term weighting (λ_α)	10^3
Slack variable weighting (λ_σ)	10^8
Input lower BC (u_l)	$0 \times 1_4$
Input upper BC (u_u)	$8 \times 1_4$
Output lower BC (y_l)	$-[10 \times 1_3 \ 0.09 \times 1_3 \ 2 \times 1_3 \ 0.18 \times 1_3 \ 0.09]^\top$
Output upper BC (y_u)	$[10 \times 1_3 \ 0.09 \times 1_3 \ 2 \times 1_3 \ 0.18 \times 1_3 \ 0.09]^\top$

It is noted that the data length used in the DeePC method, which is 1200, is shorter than that of the SPC and γ -DDPC methods, which use a data length of 2500. This difference is because the DeePC method does not require precomputed terms and

operations, which helps to reduce computing time. As previously mentioned, a longer data length generally makes the optimization problem easier to solve. In the following pages, we will present a comparison of computing times for the DDPC methods. This analysis will indicate that the DeePC algorithm takes longer to compute than the other methods, even though its data length is shorter.

Table 4.8: γ -DDPC settings

Parameter	Value
Sampling rate (T_s) [s]	0.1
Prediction horizon (N) [-]	30
Past horizon (T_{ini}) [-]	20
Data length (T) [-]	2500
State weight matrix (Q)	$\begin{bmatrix} 100 & 0 & \dots & \dots & \dots & 0 \\ 0 & 100 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 20 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & \ddots & \vdots \\ \vdots & \dots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 1 \end{bmatrix}$
Input weight matrix (R)	$10^{-2} \times I_4$
γ_2 term weighting (β_2)	0
γ_3 term weighting (β_2)	2×10^3
Input lower BC (u_l)	$0 \times 1_4$
Input upper BC (u_u)	$8 \times 1_4$
Output lower BC (y_l)	$-[10 \times 1_3 \ 0.09 \times 1_3 \ 2 \times 1_3 \ 0.18 \times 1_3 \ 0.09]^\top$
Output upper BC (y_u)	$[10 \times 1_3 \ 0.09 \times 1_3 \ 2 \times 1_3 \ 0.18 \times 1_3 \ 0.09]^\top$

4.6 Comparison of Data-driven Predictive Control Methods

Figure 4.8 provides a comparative analysis of position states (x, y, z) over time for three different control methods: SPC, DeePC, and γ -DDPC, and reveals apparent differences in the response characteristics between the three methods. While grey

lines represent each run of Monte Carlo simulations by 100 datasets, red lines show the mean of these runs.

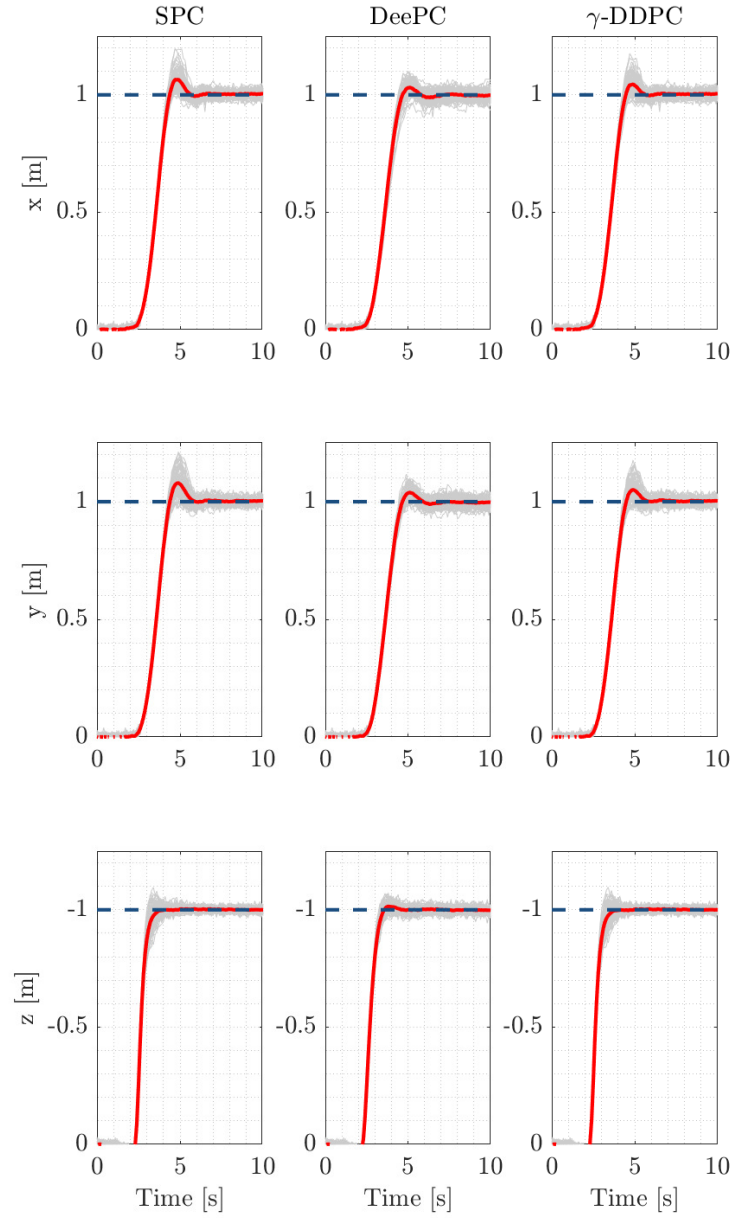


Figure 4.8: Responses of the x -position for the data-driven methods with different datasets

To conduct a detailed investigation of position step responses for all methods, the

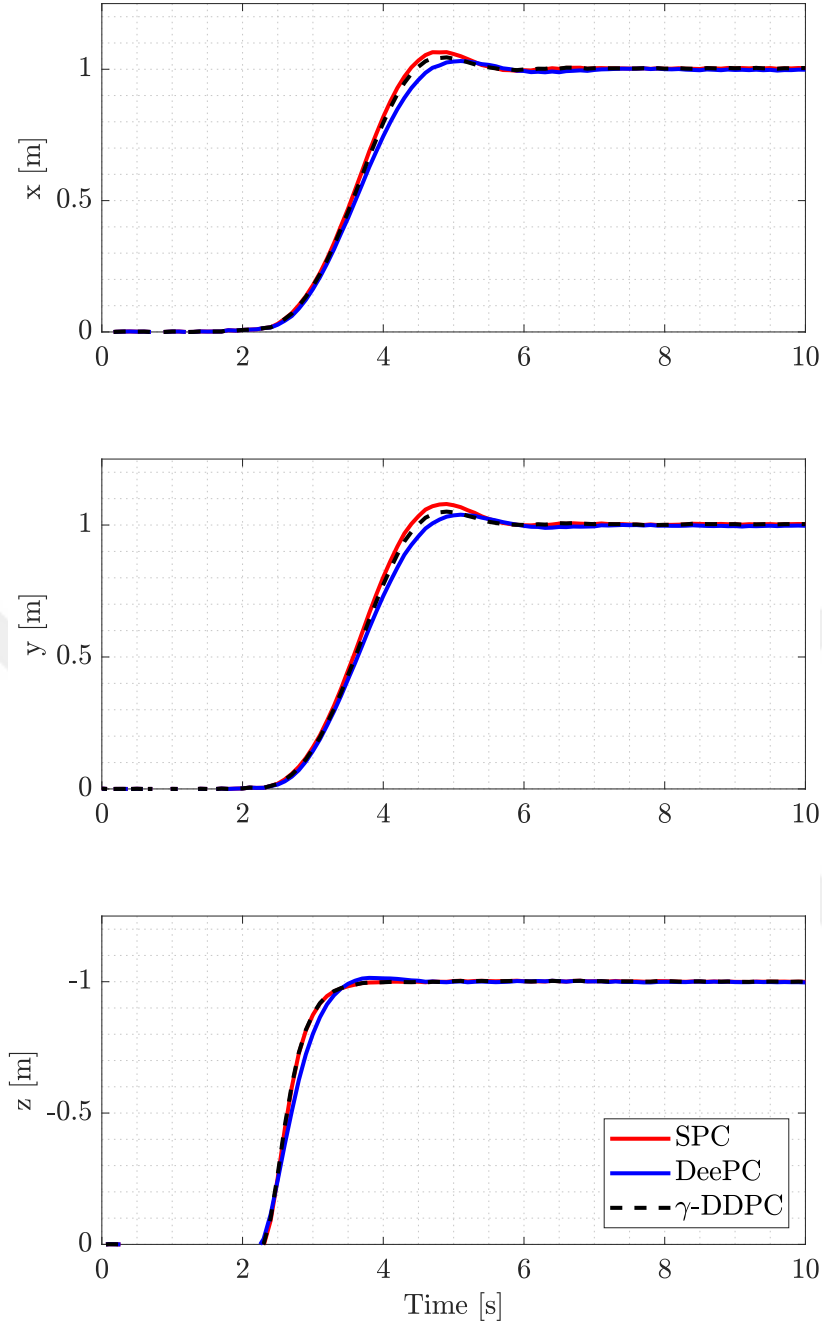


Figure 4.9: Position mean responses for the methods with different datasets

mean and standard deviations of the responses should be considered. Figure 4.9 illustrates the mean position responses along the x , y , and z axes for the three control methods: SPC, DeePC, and γ -DDPC. Regarding the z axis response, both SPC and γ -DDPC exhibit slightly better-rising behavior than the DeePC method. In contrast, for the x and y axes, γ -DDPC begins to diverge from SPC in terms of overshoot value,

as shown graphically. Furthermore, although DeePC displays a lower overshoot value in the mean response, it has a slower response compared to γ -DDPC.

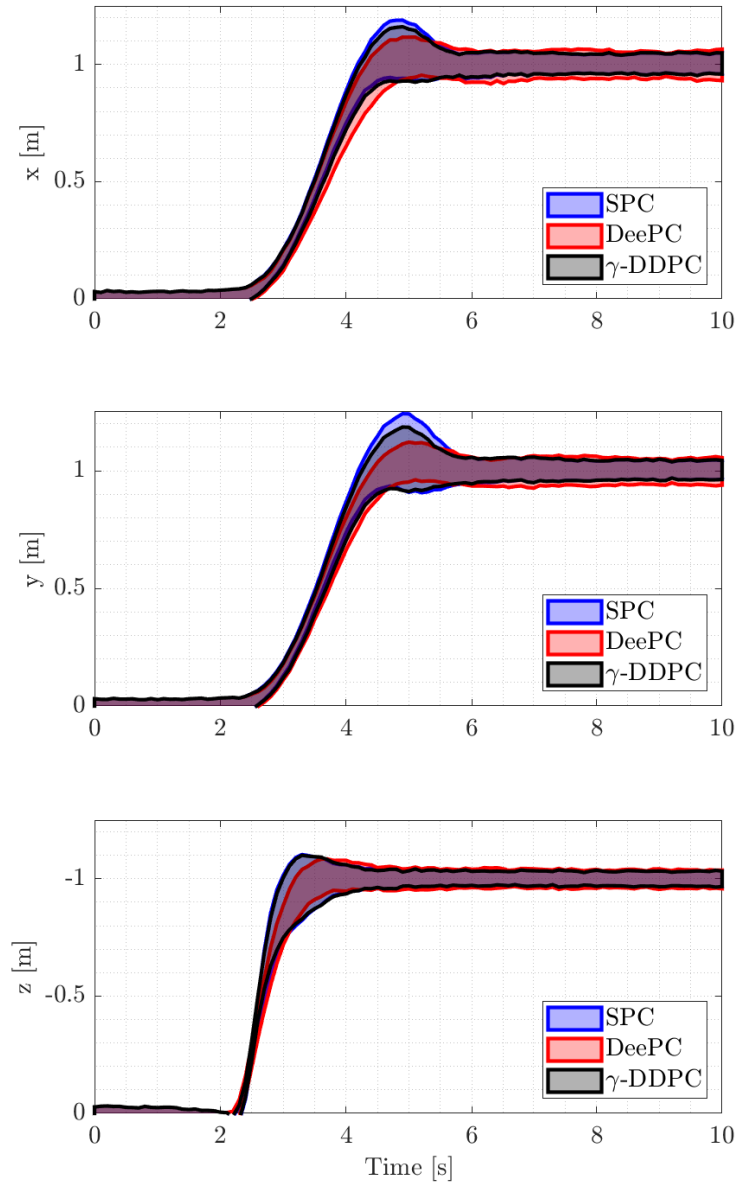


Figure 4.10: Standard deviations of position responses for the methods with different datasets

Next, the standard deviations of the position responses are calculated, as shown in

Figure 4.10. It can be seen that SPC has the highest peak values on the x and y axes, while DeePC excels in peak value. γ -DDPC exhibits an average response among these methods. Additionally, for the z axis, SPC and γ -DDPC demonstrate the same rising performance visually, while DeePC shows a smaller distribution around the mean. Furthermore, all methods exhibit nearly identical distributions around the step reference after settling.

When specifying the transient response characteristics of a control system responding to a unit-step input, the following specifications are outlined [33].

Definition 4.6.1 (Rise time (t_r)). The rise time t_r is the duration needed for the response to increase from 10% to 90% of its final value.

Definition 4.6.2 (Settling time (t_s)). The settling time t_s refers to the duration necessary for the response curve to reach and remain within 2% of the final value. In some instances, 5%, rather than 2%, is used as the percentage of the final value.

Definition 4.6.3 (Maximum (percent) overshoot (M_p)). The maximum overshoot M_p is the maximum peak value of the response curve, measured from the final steady-state value. If the final steady-state value is equal to 1, the maximum percent overshoot is $M_p \times 100\%$. If the final steady-state value of the response differs from unity, then it is common practice to use the following definition of the maximum percent overshoot:

$$M_p = \frac{y(t_p) - y(\infty)}{y(\infty)} \times 100\%$$

The amount of the maximum (percent) overshoot directly indicates the relative stability of the system.

Table 4.9: Comparison of settling times

	SPC	DeePC	γ -DDPC
$\bar{t}_{s,x} \pm \sigma_{t_{s,x}}$ [s]	3.0002 ± 0.5856	3.3147 ± 1.1806	2.7281 ± 0.3668
$\bar{t}_{s,y} \pm \sigma_{t_{s,y}}$ [s]	3.1301 ± 0.5033	3.2342 ± 0.9233	2.9152 ± 0.4155
$\bar{t}_{s,z} \pm \sigma_{t_{s,z}}$ [s]	1.4270 ± 0.2481	1.4665 ± 0.1911	1.4574 ± 0.2594

Table 4.9 presents the settling time means and standard deviations in Monte Carlo simulations. In terms of mean values, γ -DDPC shows the best performance on the x

and y axes. While SPC delivers the best performance on the z axis, γ -DDPC achieves results that are close to it. When considering the standard deviations of the settling times, γ -DDPC exhibits the smallest deviations on the x and y axes. Additionally, it is noted that DeePC shows the smallest deviation on the z axis by a significant margin.

The rise time values along the z axis are better than those along the x and y axes because the quadrotor experiences significant coupling in the xy plane. This means that vertical motion (along the z axis) can be achieved by simply adjusting the total thrust generated by all four rotors.

In contrast, movement along the x or y axes is conducted by tilting. This tilting projects a portion of the thrust vector into the horizontal plane, resulting in translational motion on xy plane. However, this process is slower and more coupled, as it requires adjusting the quadrotor's attitude while simultaneously maintaining balance during motion.

Table 4.10: Comparison of rise times

	SPC	DeePC	γ-DDPC
$\bar{t}_{r,x} \pm \sigma_{t_{r,x}}$ [s]	1.3588 ± 0.0468	1.4925 ± 0.0903	1.3668 ± 0.0444
$\bar{t}_{r,y} \pm \sigma_{t_{r,y}}$ [s]	1.3292 ± 0.0505	1.4498 ± 0.0593	1.3149 ± 0.0490
$\bar{t}_{r,z} \pm \sigma_{t_{r,z}}$ [s]	0.6835 ± 0.1361	0.7920 ± 0.0624	0.7046 ± 0.1480

Table 4.10 shows the means and standard deviations of rise times in Monte Carlo simulations. In terms of mean values, both γ -DDPC and SPC exhibit nearly identical performance across all axes, whereas DeePC deviates significantly in a negative direction. Regarding the standard deviations of rise times, γ -DDPC and SPC again display similar performance across all axes. Furthermore, it is observed that DeePC has the smallest deviation on the z axis by a considerable margin.

Additionally, the explanation for the rise time comparison applies similarly to settling time, highlighting better performance on the z axis compared to the x and y axes.

Table 4.11: Comparison of overshoot values

	SPC	DeePC	γ-DDPC
$\bar{M}_{p,x} \pm \sigma_{M_{p,x}} [\%]$	6.3411 ± 3.8754	3.9757 ± 2.7278	4.2637 ± 2.5680
$\bar{M}_{p,y} \pm \sigma_{M_{p,y}} [\%]$	7.9452 ± 4.9343	4.6502 ± 2.6758	4.3297 ± 3.6324
$\bar{M}_{p,z} \pm \sigma_{M_{p,z}} [\%]$	1.6949 ± 1.5500	2.9915 ± 1.4012	1.2981 ± 1.6578

Table 4.11 presents the mean and standard deviation of overshoots in the Monte Carlo simulations. For the mean values, DeePC performs best along the x axis, while γ -DDPC achieves the best results along the y and z axes. Regarding standard deviations, γ -DDPC outperforms on the x axis, whereas DeePC shows the best performance on the y and z axes.

It is observed that, across all axes, the standard deviations are relatively close to the mean values. This indicates significant variability in the system's response and suggests that DDPC methods may struggle with overshoot control. A key reason for this is that each dataset used in DDPC contains varying levels of noise in both inputs and outputs. Unlike model-based controllers, which rely on a well-defined mathematical model, DDPC methods are directly influenced by the quality of the data. As a result, the presence of random noise can degrade performance, especially in terms of transient behavior such as overshoot.

Table 4.12: The number of acceptable overshoot responses on position states

	SPC	DeePC	γ-DDPC
x	39	67	68
y	33	59	54
z	93	93	94

Furthermore, the comparative analysis of Table 4.12 highlights the performance differences among SPC, DeePC, and γ -DDPC in maintaining acceptable overshoot responses, defined by a %5 threshold. The value of the threshold can be defined by the damping ratio, ζ . For that case, if the damping ratio is selected as 0.7, which is a suit-

able selection for aerospace applications generally, the percentage of the overshoot value can be calculated by following the formula [33],

$$M_p = e^{-\zeta\pi/\sqrt{1-\zeta^2}}$$

$$M_p = e^{-0.7\pi/\sqrt{1-0.7^2}} = 0.046 \approx 0.05 = \%5$$

The Monte Carlo simulations indicate that all tested methods exhibit nearly identical performance concerning the overshoot threshold along the z axis. However, when considering the y and z axes, the DeePC and γ -DDPC methods outperform the others, suggesting that they are more effective at managing coupled dynamics within the xy plane compared to the SPC method. This highlights the capability of DeePC and γ -DDPC in scenarios where interactions between axes are critical, allowing for more precise control and response under such conditions.

The analysis indicates that the number of acceptable overshoot responses on the z axis is superior to those on the x and y axes. As previously mentioned, the quadrotor experiences significant coupling in the xy plane, whereas adjusting thrust against gravity primarily governs translational motion on the z axis. This distinction explains why the overshoot performance on the z axis is better than on the others.

Additionally, even though it is assumed that the quadrotor is symmetric in the xy plane, the overshoot performance differs between the x and y axes. This discrepancy arises because the Monte Carlo simulation utilizes various datasets that include different noise seeds. In other words, although the model itself is symmetric, the datasets used in DDPC methods may not exhibit such symmetry.

Based on the provided results, γ -DDPC demonstrates strong performance across multiple metrics. It achieves the smallest deviations in settling times along the x and y axes while maintaining performance close to SPC on the z axis. In terms of rise times, both γ -DDPC and SPC exhibit nearly identical mean values, with γ -DDPC showing slightly lower standard deviations, indicating more consistent performance. Regarding overshoot, γ -DDPC achieves the best results along the y and z axes. However, DeePC outperforms the others along the x axis and shows the lowest overshoot deviation along the z axis, indicating its ability to regulate specific dynamics effectively. Furthermore, the comparison of acceptable overshoot responses suggests that both

DeePC and γ -DDPC are better suited for handling coupled dynamics in the xy plane compared to SPC. Overall, while γ -DDPC demonstrates the most balanced and stable performance across all axes, DeePC exhibits strengths in specific aspects, particularly in overshoot regulation.

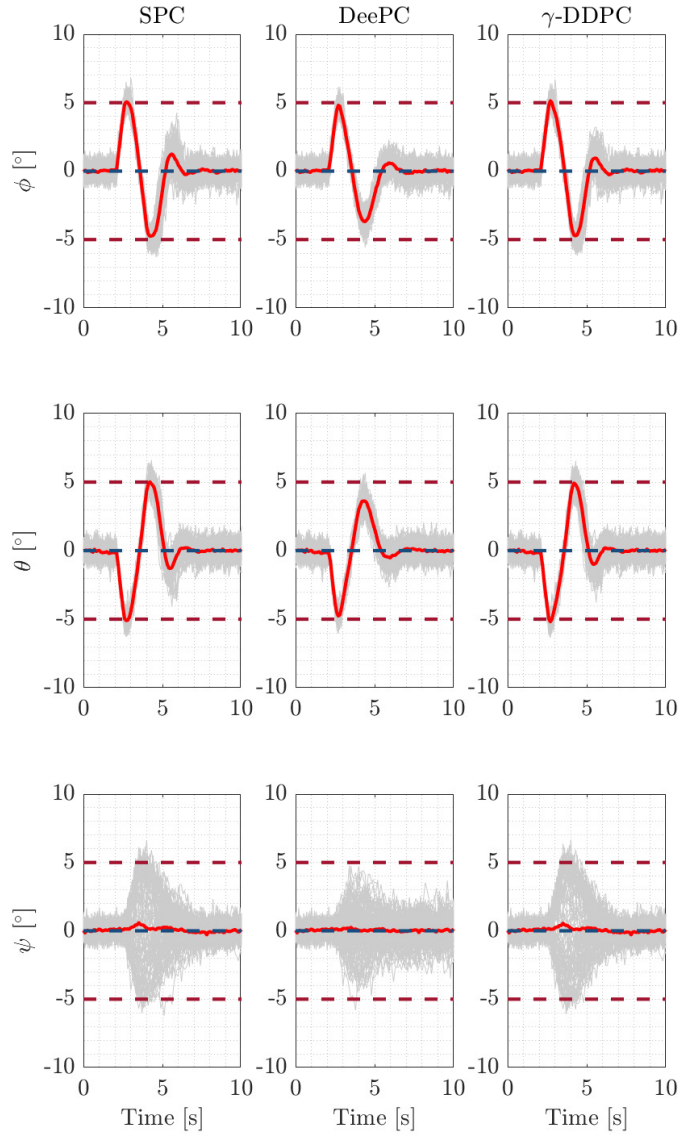


Figure 4.11: Euler angles responses for the methods with different datasets

The position of the quadrotor is intimately tied to its orientation, which is represented by Euler angles: roll, pitch, and yaw. These angles, serving as the quadrotor's control

system, dictate its tilt and turn, enabling it to adjust its thrust and move to desired positions. Comparing the Euler angles responses is the key to achieving stable and accurate positioning, giving control over the quadrotor's movement.

Figure 4.11 provides a comparative analysis of Euler angle states (ϕ, θ, ψ) over time for three different control methods: SPC, DeePC, and γ -DDPC, and reveals apparent differences in the response characteristics between the three methods.

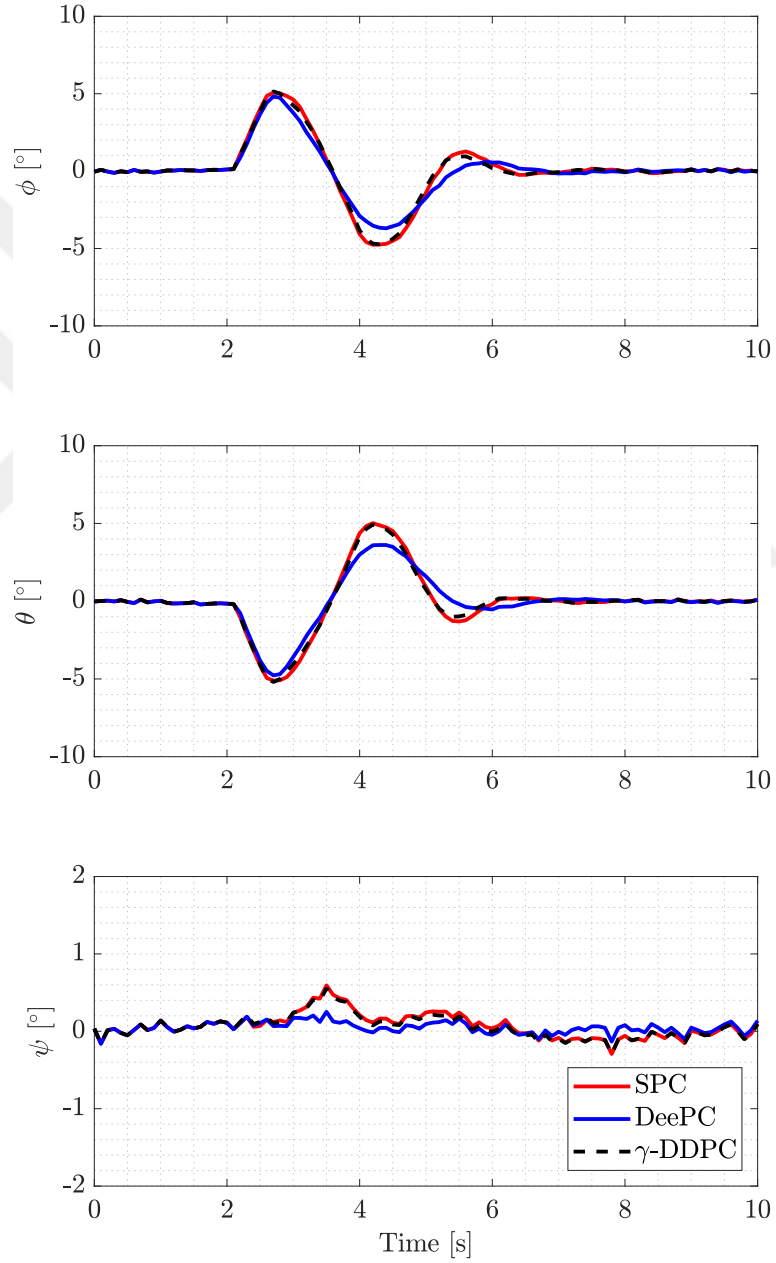


Figure 4.12: Euler angles mean responses for the methods with different datasets

To conduct a detailed investigation of changes in Euler angles for all methods, the means and standard deviations of the responses should be considered as position states. Figure 4.12 illustrates the mean Euler angles for the three control methods: SPC, DeePC, and γ -DDPC. Firstly, it can be noted that the yaw angle, ψ , remains around zero for all methods. Furthermore, for the roll and pitch angles, SPC and γ -DDPC have nearly identical values. On the other hand, DeePC exhibits lower mean amplitudes compared to the other DDPC methods.

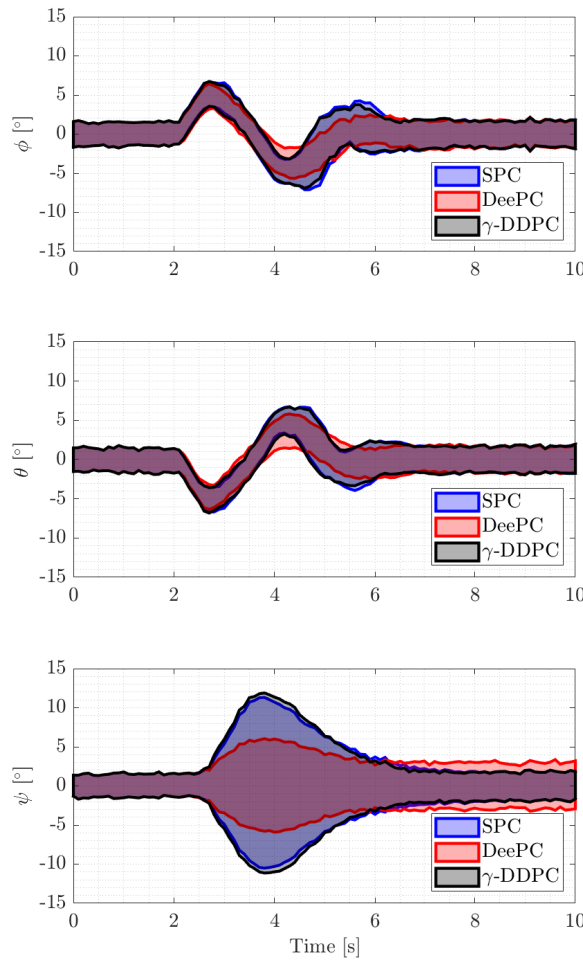


Figure 4.13: Standart deviations of Euler angles responses for the methods with different datasets

Next, the standard deviations of the Euler angles are calculated, as shown in Figure 4.13. Like the mean Euler angle plots, the standard deviation plots of the Euler angles

yield nearly the same results. Specifically, for all Euler angles, SPC and γ -DDPC show similar distributions. DeePC demonstrates a smaller variance compared to the other DDPC methods.

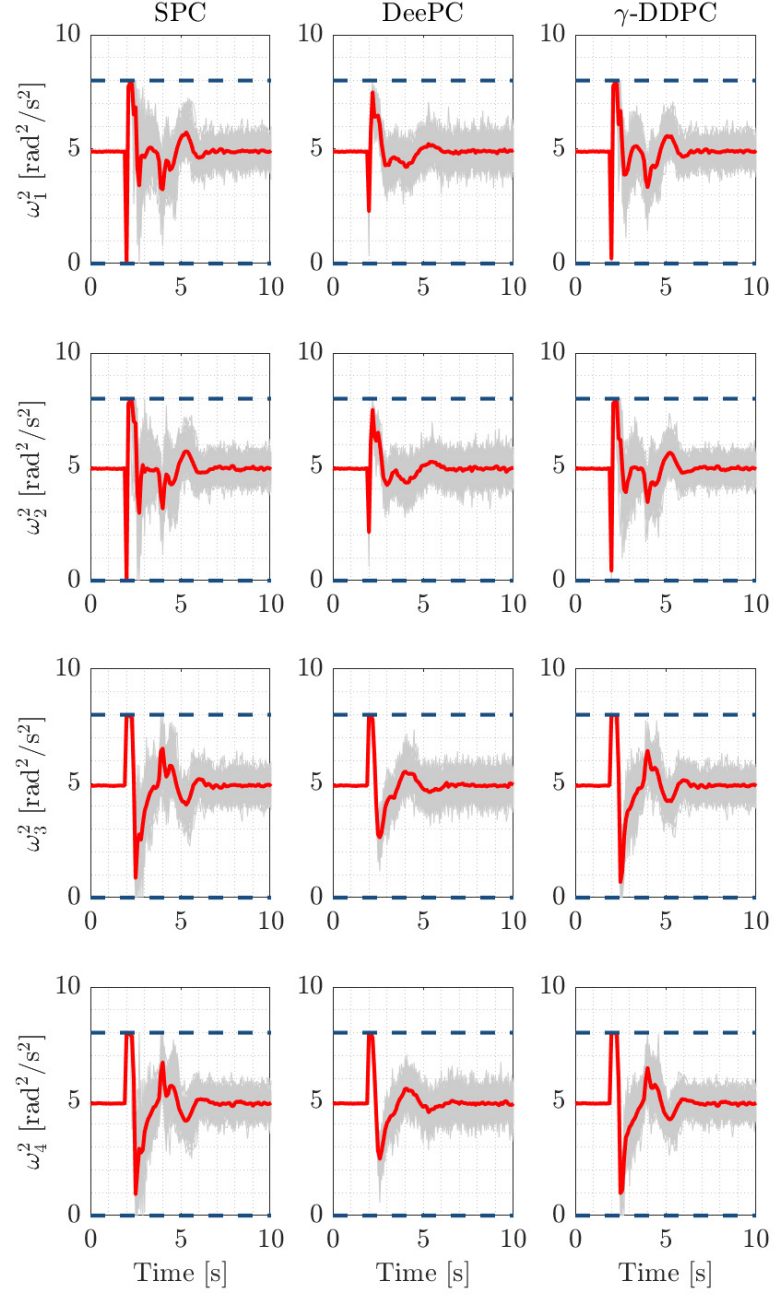


Figure 4.14: Comparative analysis of control inputs with different datasets

Lastly, for the comparison section, Figure 4.14 presents a comparative analysis of control inputs using three different control strategies: SPC, DeePC, and γ -DDPC. Each row in the figure corresponds to control inputs denoted by ω_i^2 , while the columns compare the performance of the three control methods over time.

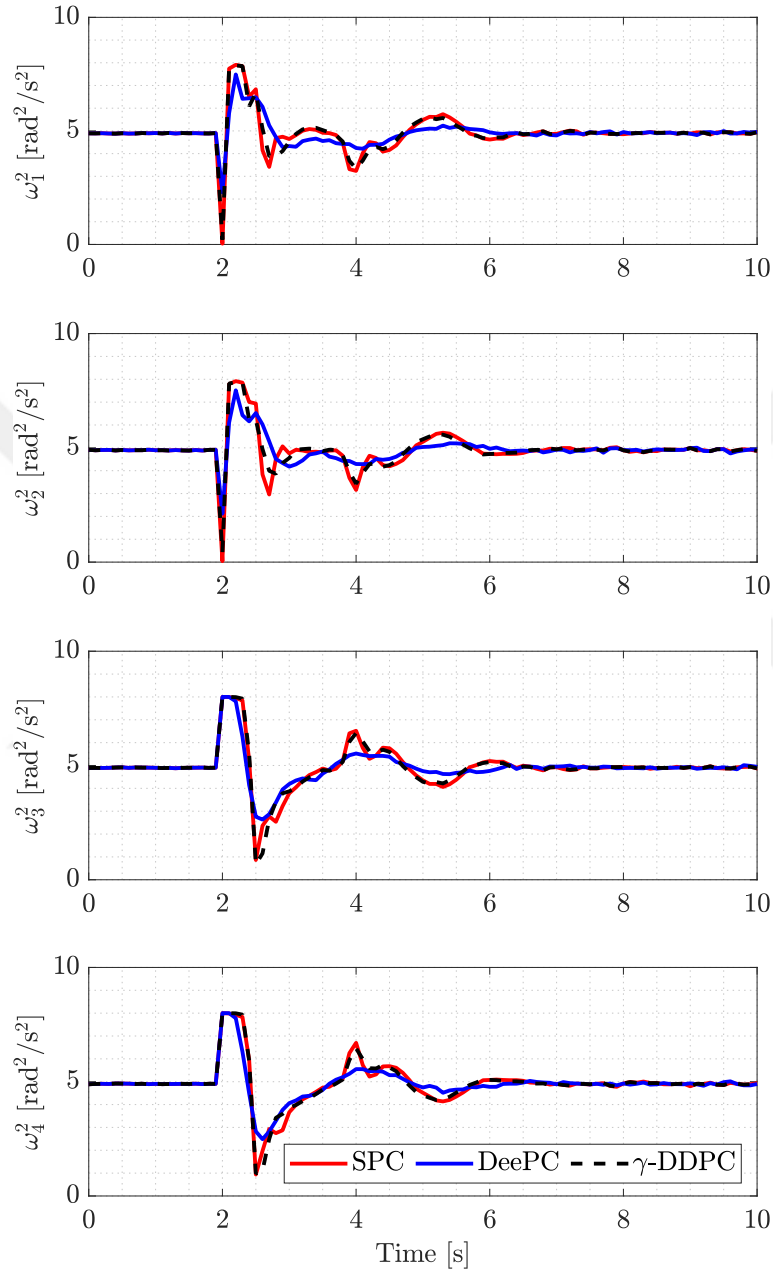


Figure 4.15: Mean control inputs for the methods with different datasets

To conduct a detailed investigation of control inputs for all methods, the means and standard deviations of the control inputs should be considered as position states and

Euler angles. Figure 4.15 illustrates the mean control inputs for the three control methods: SPC, DeePC, and γ -DDPC. As expected from responses of position and Euler angles, DeePC has lower amplitudes in control inputs with respect to SPC and γ -DDPC.

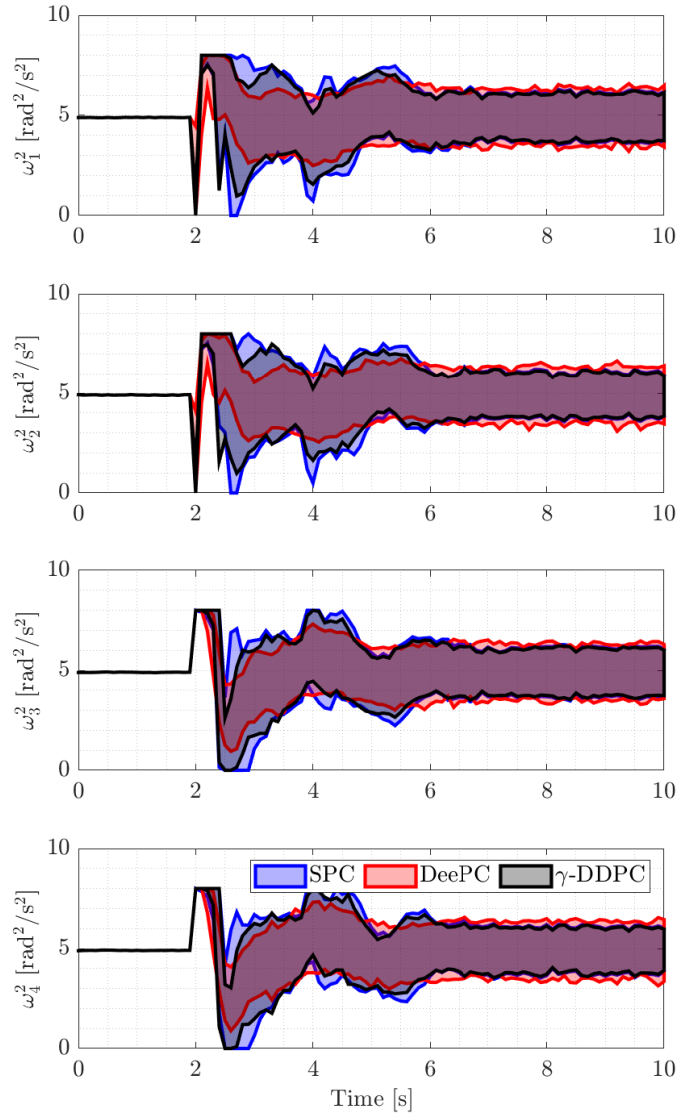


Figure 4.16: Standard deviations of control inputs for the methods with different datasets

Next, the standard deviations of the Euler angles are calculated, as shown in Figure

4.16. Like the mean Euler angle plots, the standard deviation plots of the Euler angles yield nearly the same results. Specifically, for all Euler angles, SPC and γ -DDPC show similar distributions. DeePC demonstrates smaller variance compared to the other DDPC methods.

4.7 Comparison of Computing Times of Data-driven Predictive Controllers

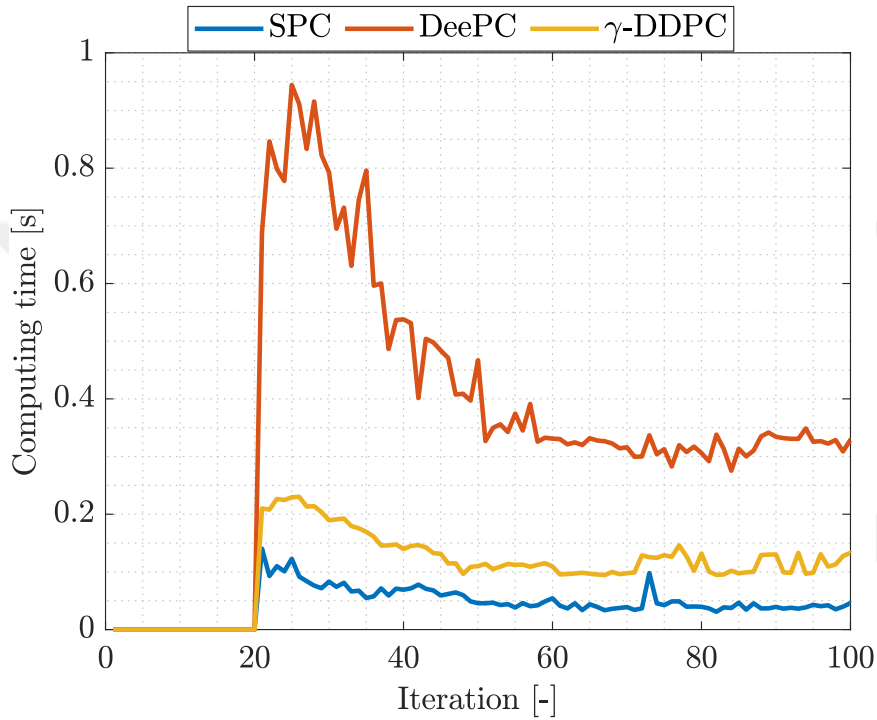


Figure 4.17: Computing time comparison

Figure 4.17 compares computing times across iterations for three control methods: SPC, DeePC, and γ -DDPC. The data was collected on a system equipped with an Intel(R) Core(TM) i5-12400F processor, a six-core CPU typical of mid-range performance capabilities. DeePC exhibits the highest computing time, particularly in the initial iterations, which peaks at around 0.9 seconds. Although DeePC's computing time gradually decreases and stabilizes around 0.3 to 0.4 seconds per iteration, it remains consistently higher than the other two methods. γ -DDPC, on the other hand, demonstrates more moderate computational demands, stabilizing at approximately 0.2 seconds per iteration after an initial transient phase. SPC outperforms both in terms of computational efficiency, maintaining a stable computing time below 0.1

seconds throughout the iterations. This comparison suggests that SPC is the most computationally efficient of the three methods, making it a suitable choice for applications where rapid computations are critical. At the same time, DeePC may require more resources due to its higher computational load.

4.8 Comparison of Predictive Controllers with Data-driven and Model-based Methods

In this section, a comparison is made between data-driven and model-based predictive controllers, specifically MPC and Nonlinear MPC (NLMPC). Firstly, an ideal scenario is examined. Secondly, a model mismatch scenario is considered, where some modeling parameters contain uncertainties.

4.8.1 Ideal Case

Differently from the previous section of the comparison, the trajectory is defined by trigonometric functions instead of a constant value.

$$\begin{aligned}x &= p \sin(2\pi\omega_d) \\y &= -p \sin(2\pi\omega_d) \cos(2\pi\omega_d) \\z &= p \cos(2\pi\omega_d)\end{aligned}\tag{4.2}$$

Table 4.13 indicates the quadrotor should track trajectory parameters. Amplitude and frequency are selected so that the quadrotor can follow the reference by the controller.

Table 4.13: Trajectory parameters to be followed

Parameter	Value
Amplitude of trigonometric functions (p) [m]	6
Frequency of trigonometric functions (ω_d) [Hz]	$\frac{1}{2\pi \cdot 3}$

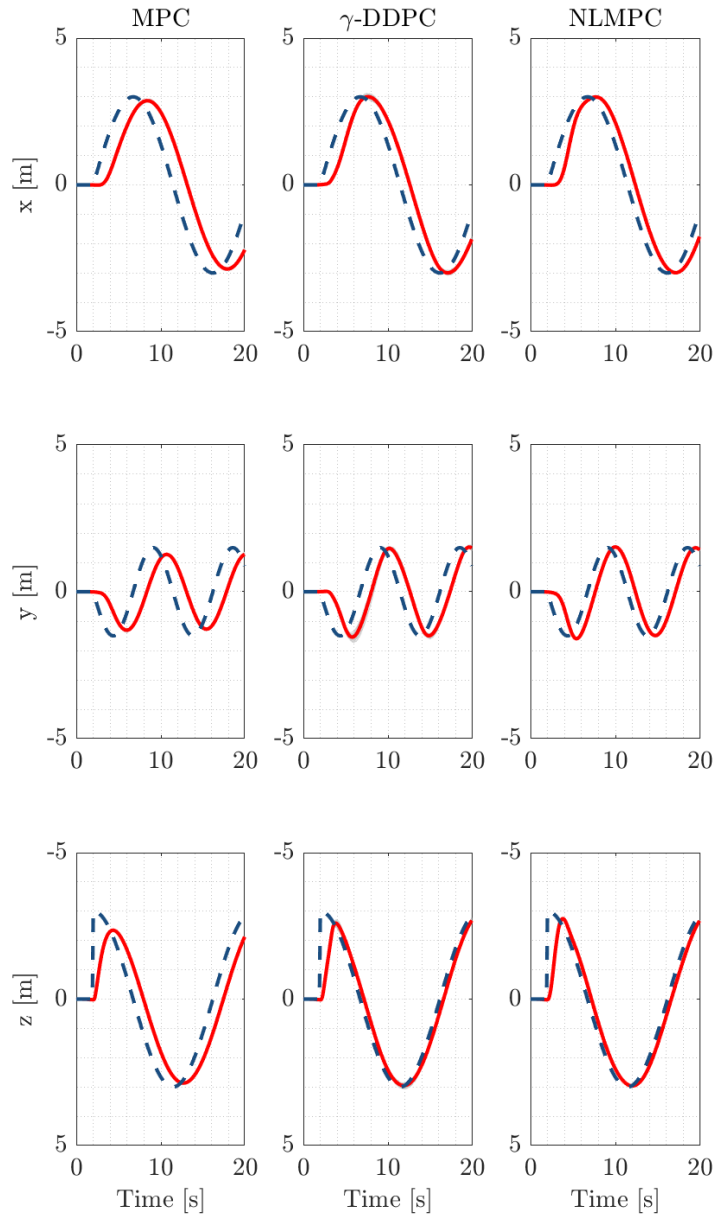


Figure 4.18: Position outputs comparison of data-driven and model-based controller on the ideal case

Figure 4.18 compares the position outputs along the (x,y,z) axes for γ -DDPC, MPC, and NLMPC under ideal conditions with no uncertainty. The reference trajectory, shown in purple, represents the desired position on each axis. It can be concluded

that NLMPC successfully follows the defined trajectory. In contrast, neither γ -DDPC nor MPC performs as well as NLMPC. To provide a detailed analysis, the following paragraphs offer further insight into the tracking performance.

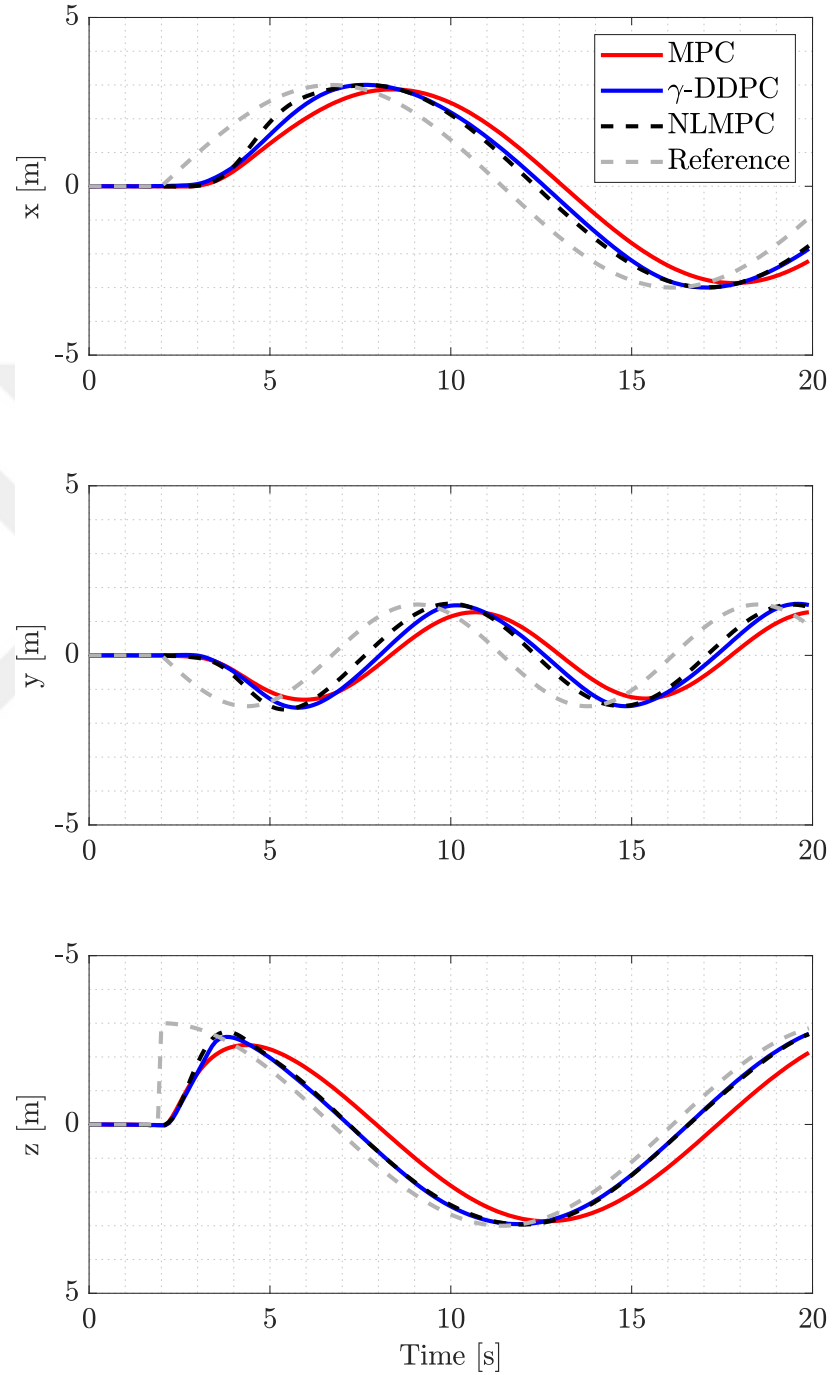


Figure 4.19: Mean position outputs comparison of data-driven and model-based controller on the ideal case

Figure 4.19 illustrates the mean position outputs for each controller. This figure indicates that NLMPC outperforms the others. Additionally, it can be noted that γ -DDPC is quicker than the MPC, as it aligns more closely with the NLMPC response and the reference.

Additionally, to compare γ -DDPC with other data-driven methods over the reference tracking performance, Integral Absolute Error can be utilized. Integral Absolute Error (IAE) is defined as follows:

$$\text{IAE} = \int_{t=T_{\text{ini}}}^{t_f} \|\zeta_r(t) - \zeta(t)\|_1 dt$$

where ζ represents the positional variables of (x, y, z) , and ζ_r denotes its reference value. Also, the lower bound of the integral is taken as T_{ini} since the controllers are not active until $t = T_{\text{ini}}$. Then, the IAE values for each method is given in Table 4.14 where $\overline{\text{IAE}}$ is the mean of IAE and σ_{IAE} is the standard deviation of IAE.

Table 4.14: Integral Absolute Error (IAE) Comparison for the Ideal Case

MPC		γ -DDPC		NLMPC	
$\overline{\text{IAE}}$	σ_{IAE}	$\overline{\text{IAE}}$	σ_{IAE}	$\overline{\text{IAE}}$	σ_{IAE}
482	1.38	332	4.89	290	1.08

Table 4.14 shows that NLMPC achieves the lowest IAE, significantly outperforming MPC and γ -DDPC, indicating superior accuracy. Additionally, NLMPC exhibits the lowest variability, making it the most stable method. γ -DDPC, while more accurate than MPC, has the highest variability, indicating potential sensitivity to external disturbances. These results highlight that NLMPC is the best-performing approach in terms of accuracy and stability, though further investigation into its computational demands and robustness is necessary.

Figure 4.20 illustrates the control inputs generated by the γ -DDPC, the MPC, and the NLMPC under conditions of no uncertainty across 100 different noise sets in measurements. It is noteworthy that the MPC and the NLMPC are constrained such that the bounds of the generated control inputs range from 0 to 10. In contrast, the

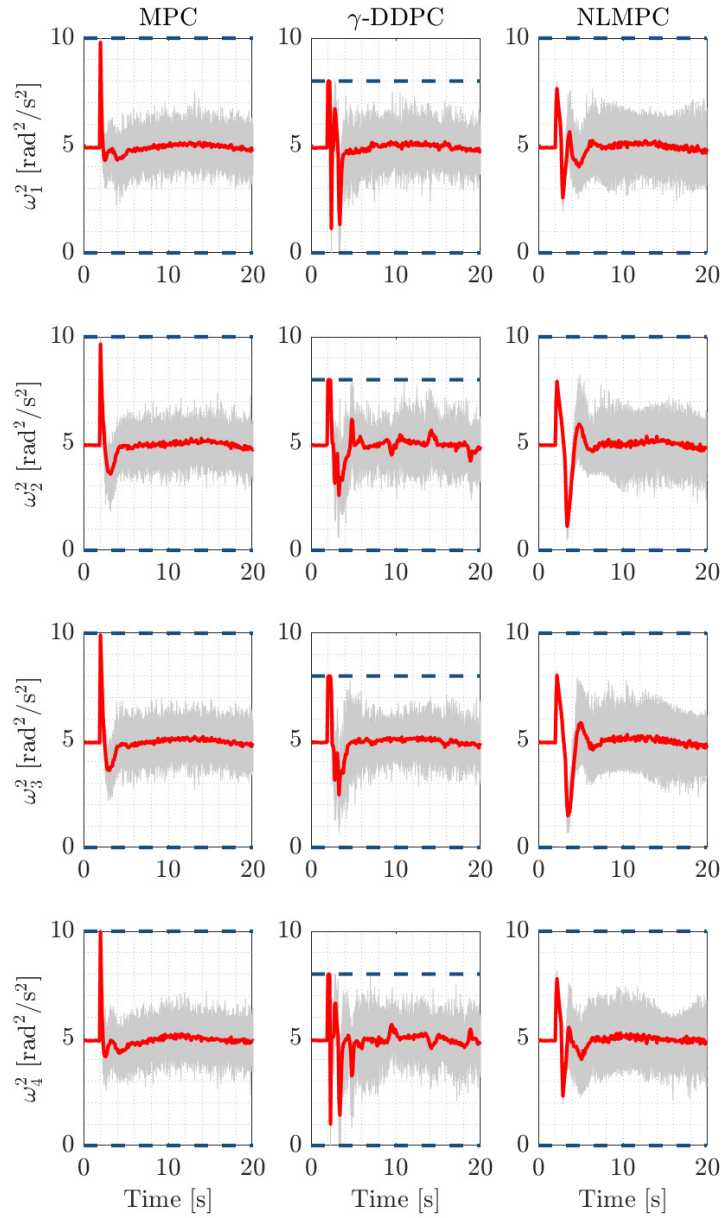


Figure 4.20: Control inputs comparison of data-driven and model-based controller on the ideal case

γ -DDPC has an upper bound of 8. This discrepancy arises because the DDPC is data-driven, whereas the MPC methods rely on model information. In other words, the data is valid within the range of 0 to 8, while the model is applicable between 0 and 10.

First of all, Figure 4.21 displays the mean control inputs for each method. It can be stated that NLMPC gives more aggressive rotor inputs initially with respect to MPC and γ -DDPC. Also, it can be said that γ -DDPC oscillates more after settling due to noisy measurements.

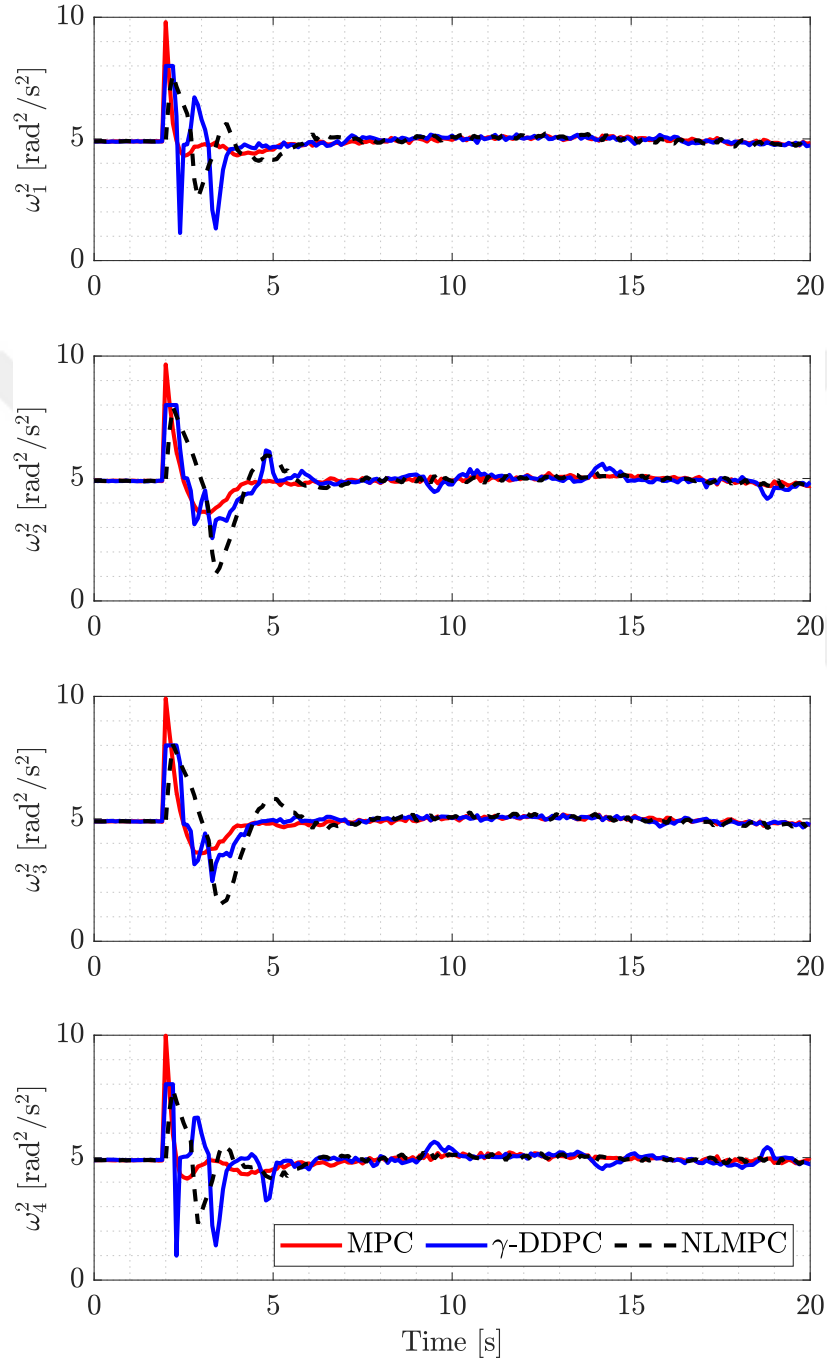


Figure 4.21: Mean control inputs comparison of data-driven and model-based controller on the ideal case

Furthermore, Figure 4.22 shows that γ -DDPC has higher variance with respect to other methods in terms of control inputs. It is because of γ -DDPC is based on noisy measurements.

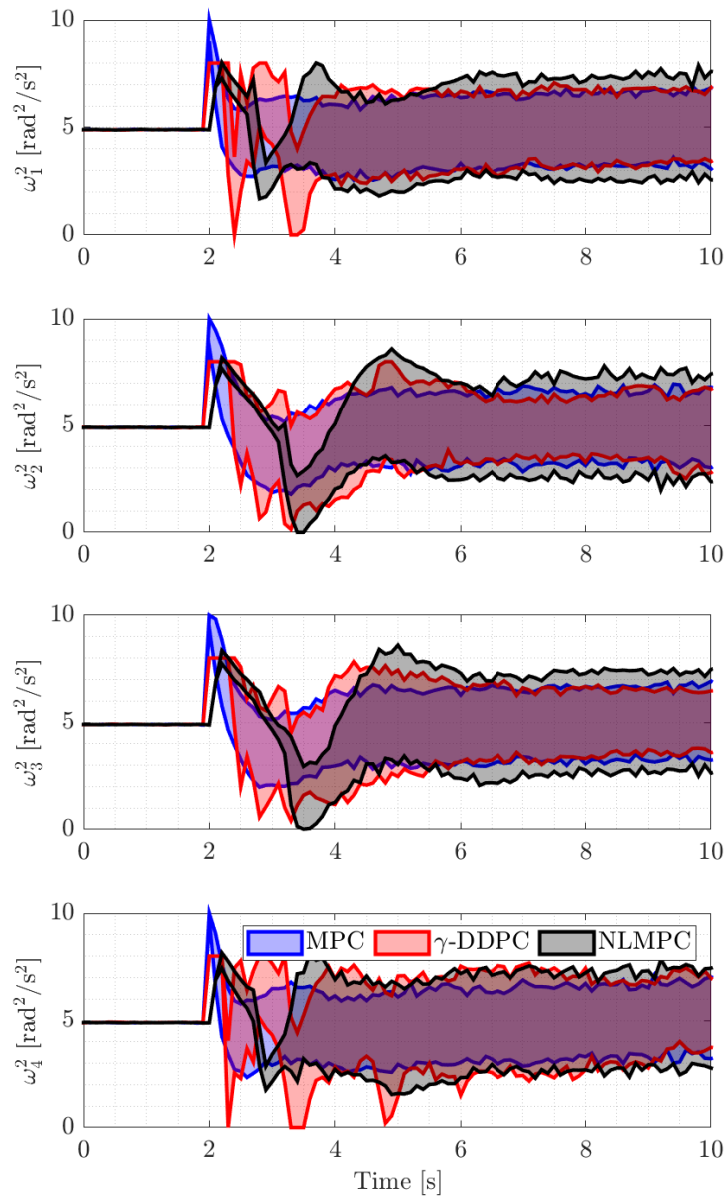


Figure 4.22: Standard deviations of control inputs comparison of data-driven and model-based controller on the ideal case

Also, in order to investigate the controllers numerically, Input Energy (IE) can be defined as follows:

$$\text{IE} = \int_{t=T_{\text{ini}}}^{t_f} \|u(t)\|_2^2 dt$$

where u represents the control input vector. IAE indicates the tracking performance, while IE reflects the control effort involved in the tracking task. Then,

Table 4.15: Input Energy (IE) Comparison the Ideal Case

MPC		γ -DDPC		NLMP	
$\bar{\text{IE}}$	σ_{IE}	$\bar{\text{IE}}$	σ_{IE}	$\bar{\text{IE}}$	σ_{IE}
18015	19.79	18175	38.27	18320	27.76

Table 4.15 indicates that MPC is the most energy-efficient method, with the lowest IE and the smallest variability. γ -DDPC, while consuming slightly more energy, exhibits the highest variability, implying potential sensitivity to disturbances. NLMP, on the other hand, has the highest energy consumption but a lower standard deviation compared to γ -DDPC, indicating more consistency but less efficiency than MPC. These findings suggest that while MPC provides the most stable and energy-efficient solution, NLMP and γ -DDPC may offer trade-offs in terms of robustness or performance.

4.8.2 Model Mismatch Case

Alongside the ideal case, a model mismatch case is introduced. In mathematical modeling, there may be some discrepancies in model parameters, which can negatively affect controller performance. Furthermore, the trajectory is the same as in the previous case.

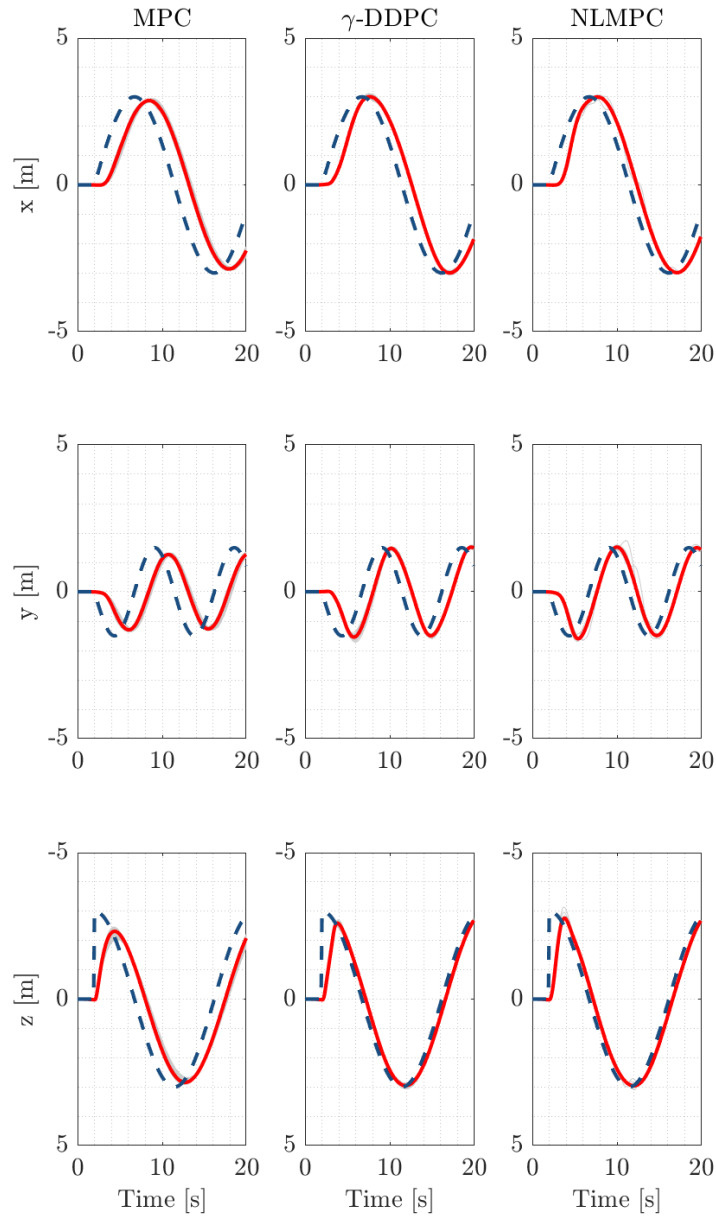


Figure 4.23: Position outputs comparison of data-driven and model-based controller on the model mismatch case

Figure 4.23 compares the positional outputs along the x , y , and z axes for γ -DDPC, MPC, and NLMPC under conditions of parameter uncertainty. Specifically, a 25% model mismatch was introduced in both the thrust factor (b) and the drag coefficient

(d) to evaluate the robustness of each control method. It is important to note that the results of γ -DDPC remain the same as in the ideal case, as the parameter model mismatch does not impact the system's measurements. Instead, the uncertainty affects the mathematical model.

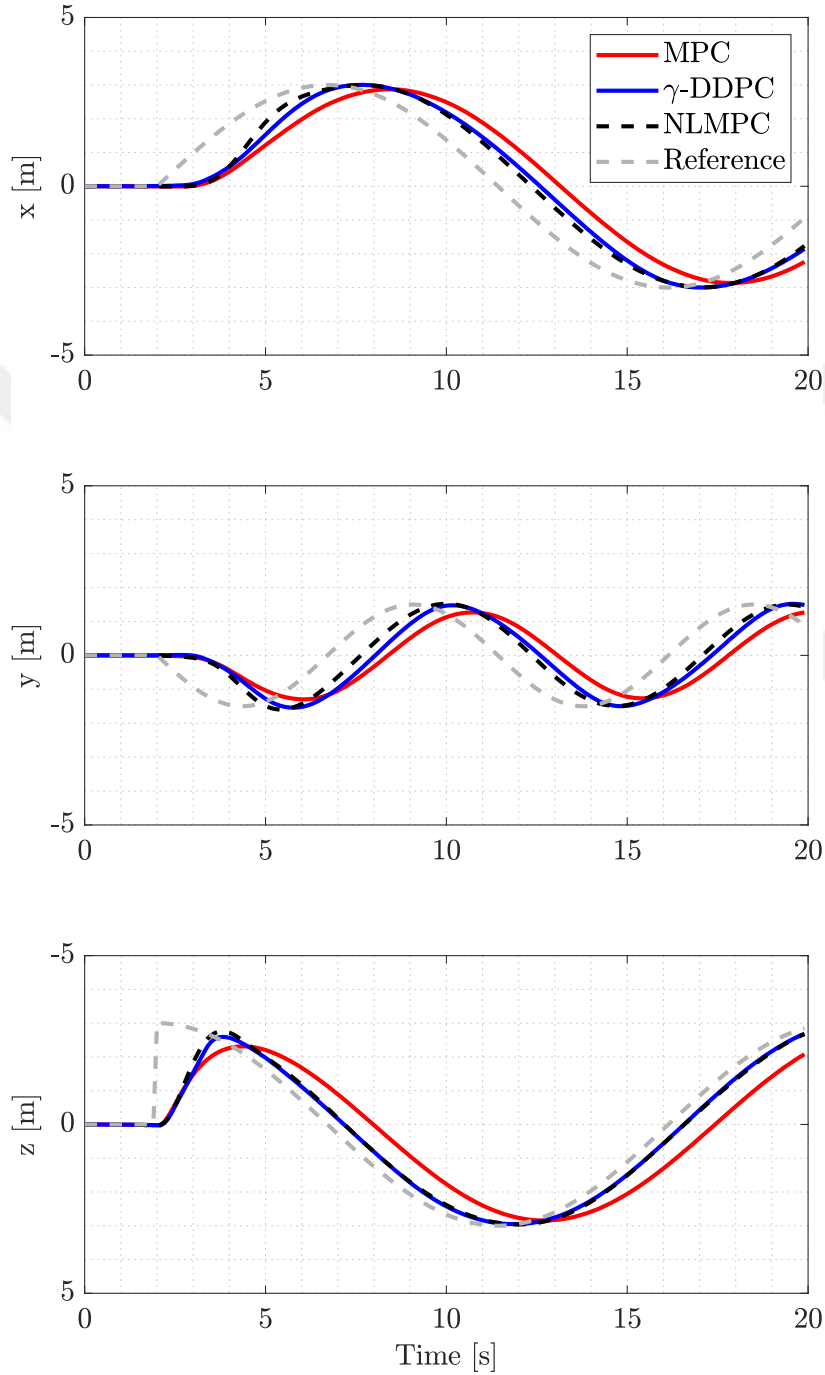


Figure 4.24: Mean position outputs comparison of data-driven and model-based controller on the model mismatch case

Figure 4.24 illustrates the mean position outputs for each controller. This figure indicates that NLMPC outperforms the others. Additionally, it can be noted that γ -DDPC is quicker than the MPC, as it aligns more closely with the NLMPC response and the reference.

Additionally, to compare γ -DDPC with other data-driven methods, Integral Absolute Error and Input Energy can be utilized for also the model mismatch case.

Table 4.16: Integral Absolute Error (IAE) Comparison for the Model Mismatch Case

MPC		γ -DDPC		NLMPC	
$\bar{\text{IAE}}$	σ_{IAE}	$\bar{\text{IAE}}$	σ_{IAE}	$\bar{\text{IAE}}$	σ_{IAE}
495	34.11	332	4.89	302	3.78

For the model mismatch case, Table 4.16 indicates that NLMPC maintains the lowest IAE (302) and the lowest variability ($\sigma_{\text{IAE}} = 3.78$), making it the most robust approach. γ -DDPC, while slightly less accurate, remains a stable alternative with moderate IAE (332) and a low standard deviation (4.89). MPC, in contrast, struggles the most, exhibiting the highest error (495) and the most performance variability ($\sigma_{\text{IAE}} = 34.11$). These findings suggest that NLMPC is the most reliable method under model uncertainty, while MPC may require additional adaptation mechanisms to improve robustness.

Figure 4.25 illustrates the control inputs generated by the γ -DDPC, the MPC, and the NLMPC under conditions of uncertainty across 100 different noise sets in measurements. It is noteworthy that the MPC and the NLMPC are constrained such that the bounds of the generated control inputs range from 0 to 10. In contrast, the γ -DDPC has an upper bound of 8. This discrepancy arises because the DDPC is data-driven, whereas the MPC methods rely on model information. In other words, the data is valid within the range of 0 to 8, while the model is applicable between 0 and 10.

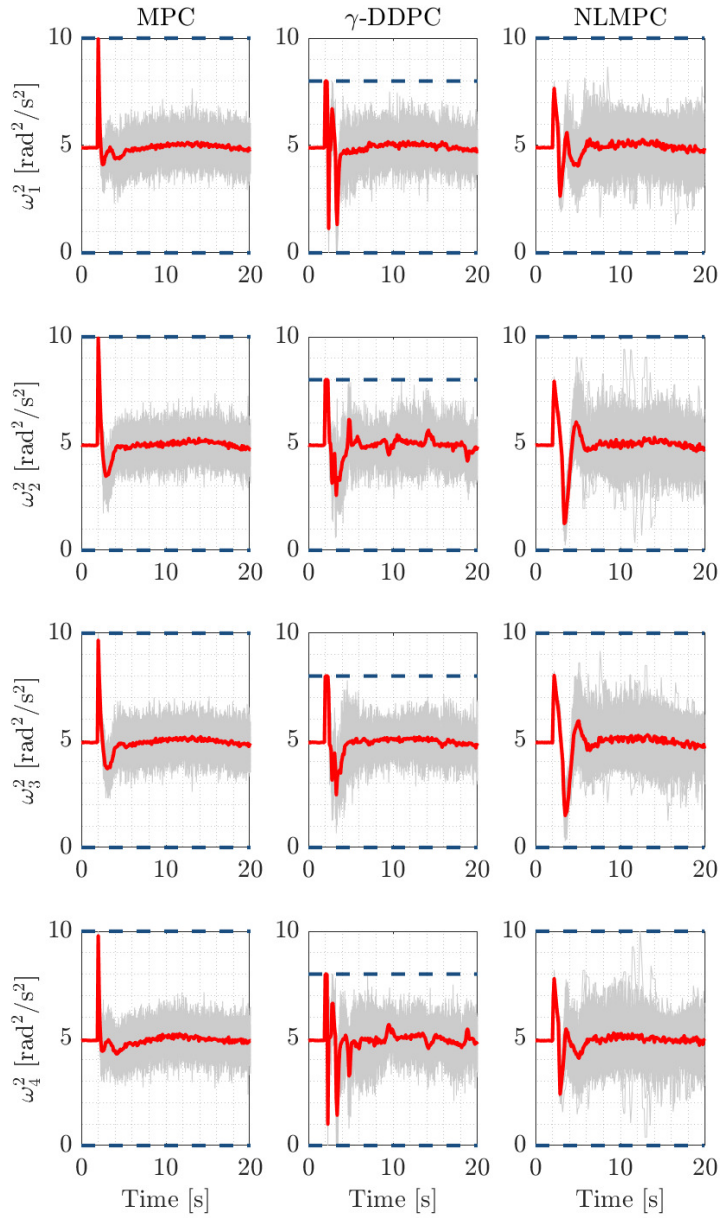


Figure 4.25: Control inputs comparison of data-driven and model-based controller on the model mismatch case

First, Figure 4.26 displays the mean control inputs for each method. It is evident that both the γ -DDPC and the NLMPC exhibit oscillations at the beginning of the control process compared to the MPC. Additionally, the γ -DDPC demonstrates oscillations

after settling, in contrast to the model-based approaches.

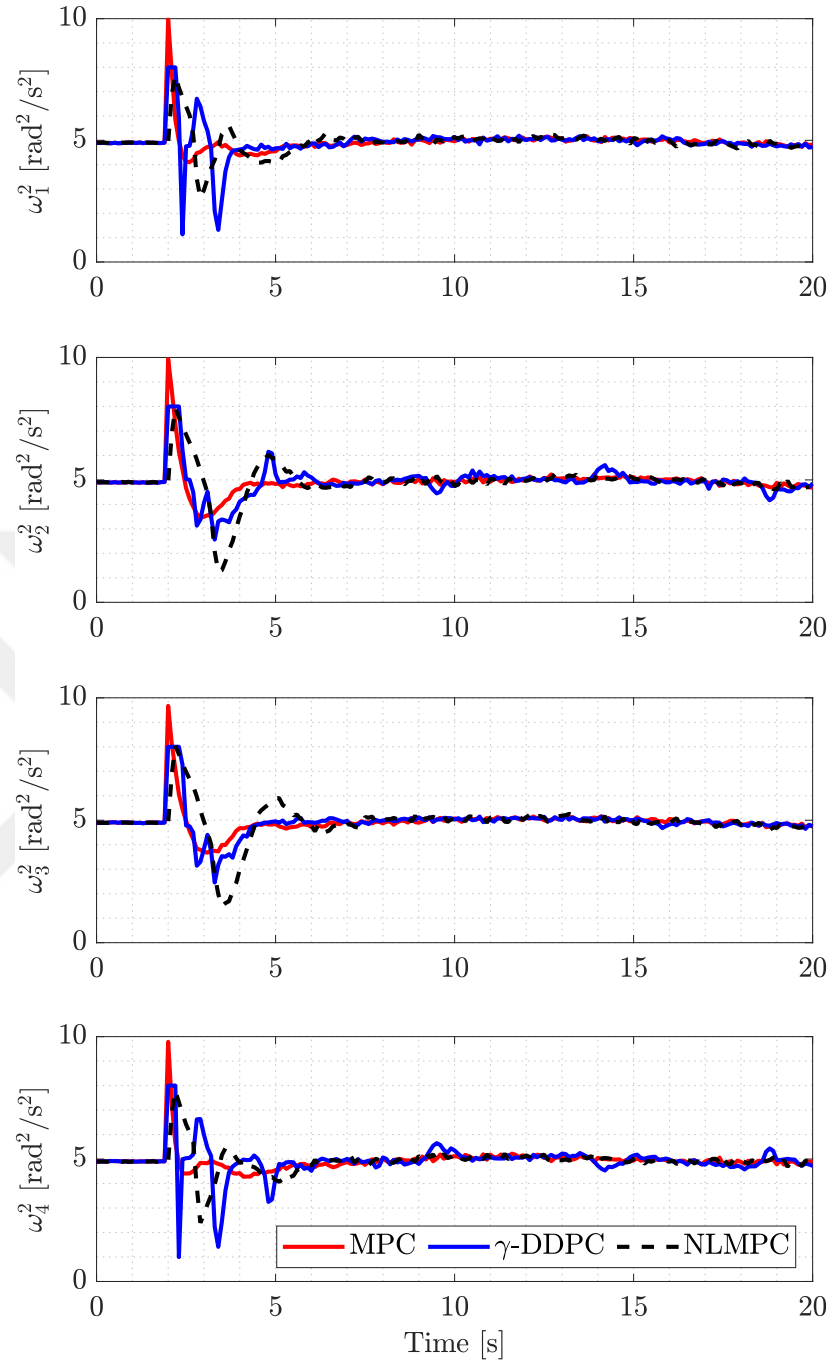


Figure 4.26: Mean control inputs comparison of data-driven and model-based controller on the model mismatch case

Furthermore, Figure 4.27 displays the standard deviations of control inputs for each method. It can be said that NLMPC have larger variance with respect to the MPC and

the γ -DDPC methods. Therefore, it can be said that control inputs generated by the NLMPC are much affected by the uncertainty with respect to the linear MPC.

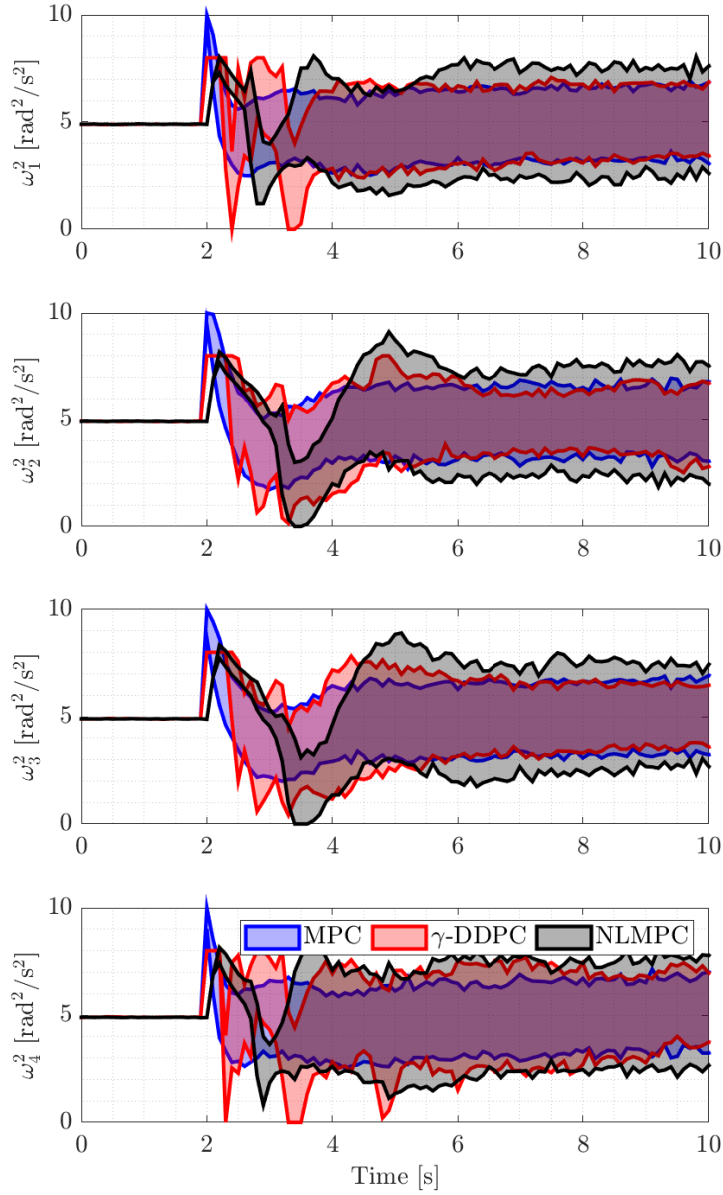


Figure 4.27: Standard deviations of control inputs comparison of data-driven and model-based controller on the model mismatch case

Table 4.17 indicates that γ -DDPC is the most energy-efficient strategy, maintaining

Table 4.17: Input Energy (IE) Comparison for the Model Mismatch Case

MPC		γ -DDPC		NLMPC	
$\bar{\text{IE}}$	σ_{IE}	$\bar{\text{IE}}$	σ_{IE}	$\bar{\text{IE}}$	σ_{IE}
18215	409.79	18175	38.27	18407	191.31

the lowest IE (18175) with minimal variability ($\sigma_{\text{IE}} = 38.27$), making it a stable choice. In contrast, MPC, despite having a comparable IE (18215), exhibits the highest standard deviation ($\sigma_{\text{IE}} = 409.79$), indicating significant fluctuations in control effort. NLMPC, although requiring the most energy (18407), demonstrates moderate variability ($\sigma_{\text{IE}} = 191.31$), suggesting a balance between robustness and control effort. These findings highlight the trade-offs between energy efficiency, stability, and robustness in handling model mismatch uncertainties.

4.8.3 Computational Cost

Figure 4.28 provides a comparative analysis of the computing times in an ideal case scenario. DeePC exhibits the highest computational burden, with its computing time spiking initially to over 1 second before stabilizing around 0.4 seconds. This can be attributed to its iterative optimization requirements, which are computationally intensive. Conversely, the other controllers demonstrate significantly lower and more consistent computing times. SPC and MPC emerge as the most computationally efficient, maintaining values well below 0.1 seconds throughout the iterations. γ -DDPC and NLMPC show intermediate performance, with γ -DDPC slightly outperforming NLMPC in terms of lower computational effort. These results highlight the trade-off between computational efficiency and control performance, with DeePC being the most computationally expensive despite its potential benefits. Overall, the figure underscores the computational efficiency of SPC, MPC, and γ -DDPC, making them more suitable for real-time applications where computational constraints are critical.

Figure 4.29 illustrates the computing time comparison under the model mismatch case. Similar to the ideal scenario, DeePC again exhibits the largest computational burden, with a notable spike, reaching values above 1 second before stabilizing around

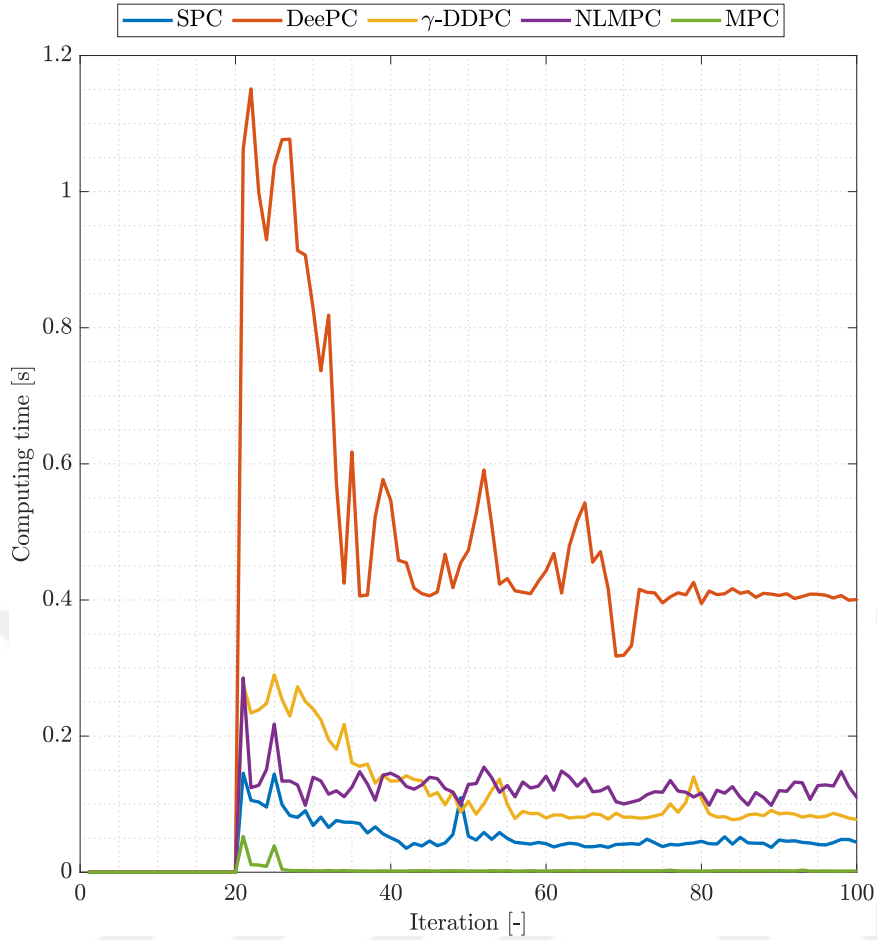


Figure 4.28: Computing time comparison of data-driven and model-based controller on an ideal case

0.4 seconds. However, NLMPC shows greater variability in computation time under uncertainty, with occasional peaks exceeding 0.3 seconds. This suggests that NLMPC’s nonlinear optimization struggles to maintain consistency under the model mismatch conditions.

In contrast, SPC and MPC maintain relatively stable and low computing times, both remaining under 0.1 seconds across all iterations, demonstrating their robustness and computational efficiency. γ -DDPC shows a slight increase compared to the ideal case but still outperforms NLMPC and DeePC regarding computational stability and efficiency. These results emphasize that while DeePC and NLMPC may offer advantages in certain scenarios, they face significant computational challenges under uncertainty, limiting their real-time applicability. In comparison, SPC, MPC, and γ -DDPC

emerge as more reliable candidates for real-world applications where computational resources are constrained and uncertainties are present.

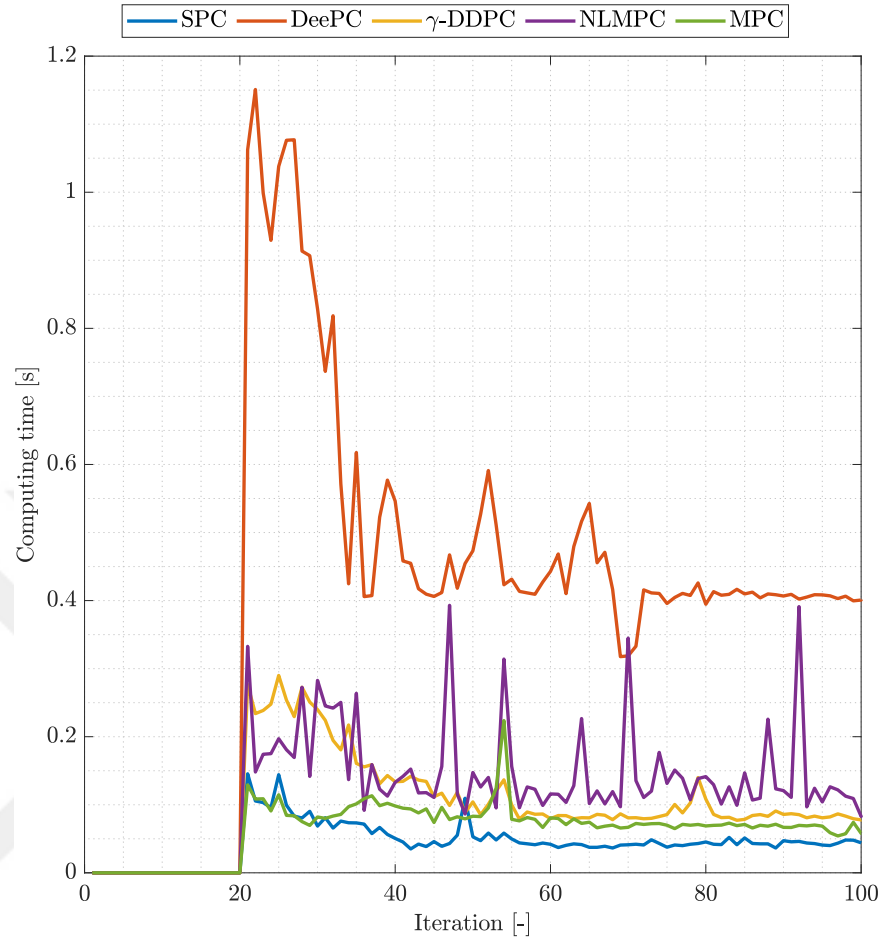


Figure 4.29: Computing time comparison of data-driven and model-based controller on an the model mismatch case



CHAPTER 5

CONCLUSIONS

This thesis explores the comparative analysis of data-driven predictive methods in the context of quadrotor control systems. Simulations of these methods are conducted in MATLAB using a realistic quadrotor model, with the introduction of parameter uncertainties. Several significant insights are yielded from these findings, which contribute to the advancement of understanding data-driven approaches in modern control systems.

1. **Performance of DDPC methods:** The results suggest that γ -DDPC provides the most consistent and balanced performance across all spatial axes. It achieves the smallest deviations in settling time for both the x and y axes while maintaining performance comparable to SPC along the z axis. Additionally, γ -DDPC exhibits lower standard deviations in rise time compared to SPC, indicating more reliable behavior. In terms of overshoot, γ -DDPC demonstrates superior regulation on the y and z axes, whereas DeePC excels on the x axis and minimizes overshoot variation in the z direction. This highlights DeePC's ability to effectively regulate specific directional dynamics. The findings indicate that both γ -DDPC and DeePC are better suited for managing coupled dynamics in the xy plane compared to SPC. While γ -DDPC offers the most robust overall control across axes, DeePC shows significant strengths in managing overshoot, emphasizing their complementary advantages within data-driven control frameworks.
2. **Robustness under uncertainty:** One of the notable strengths of γ -DDPC lies in its remarkable ability to manage uncertainties effectively. In practical applications, it frequently encounters variations in critical parameters, such as rotor

dynamics, mass distribution, and moment of inertia. Despite these fluctuations, γ -DDPC's performance in control tasks remains largely unaffected, showcasing its robustness and reliability. In stark contrast, traditional model-based control methods tend to exhibit significant sensitivity to these uncertainties. This vulnerability often necessitates frequent recalibration or the implementation of compensation techniques to maintain consistent and dependable operation, emphasizing the superiority of γ -DDPC in complex, real-world scenarios.

3. **Drawbacks of DDPC methods:** DDPC methods rely heavily on the quality of the collected dataset. This data must be carefully designed and sufficiently informative to capture the system dynamics. Poorly collected data may introduce excessive noise or lead to overfitting, reducing the controller's effectiveness. Additionally, DDPC methods may exhibit larger overshoots in transient responses, which can be critical in applications like obstacle avoidance—where precise positioning is essential. Lastly, DDPC methods often involve higher computational costs than traditional model-based controllers, which may limit their use in real-time or resource-constrained systems.
4. **Implications for Future Work:** The research emphasizes the increasing significance of data-driven methodologies within the realm of control engineering. Despite this growing focus, it is essential to recognize that the effectiveness of DDPC methods relies heavily on the quality and volume of the data utilized. This dependence underlines the urgent requirement for efficient strategies to collect and preprocess data in real-world applications. Moreover, in order to reduce computing time and complexity, modern machine learning methods can be used instead of traditional quadratic programming.

In conclusion, this study illustrates that DDPC methods are viable alternatives to traditional control strategies for managing complex systems such as quadrotors. Their impressive robustness and adaptability render them particularly suitable for situations characterized by model uncertainty or considerable variability in system behavior. Future research could enhance these methods further by investigating ways to minimize computing time and complexity. Additionally, it could explore alternative approaches that stabilize unstable systems without relying on open-loop strategies. This

could ultimately enhance operational reliability and performance in various dynamic environments.





REFERENCES

- [1] J. Berberich and F. Allgöwer. A trajectory-based framework for data-driven system analysis and control. In *2020 European Control Conference (ECC)*, pages 1365–1370. IEEE, 2020.
- [2] M. Bergamasco and M. Lovera. Identification of linear models for the dynamics of a hovering quadrotor. *IEEE transactions on control systems technology*, 22(5):1696–1707, 2014.
- [3] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.
- [4] G. J. C. Betancourt Vera. *Autonomous robust control techniques for dynamic target tracking*. Theses, Université de Technologie de Compiègne, June 2021.
- [5] S. Bouabdallah, P. Murrieri, and R. Siegwart. Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 5, pages 4393–4398. IEEE, 2004.
- [6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Number 1. böl. in Berichte über verteilte messsysteme. Cambridge University Press, 2004.
- [7] V. Breschi, A. Chiuso, and S. Formentin. Data-driven predictive control in a stochastic setting: A unified framework. *Automatica*, 152:110961, 2023.
- [8] V. Breschi, M. Fabris, S. Formentin, and A. Chiuso. Uncertainty-aware data-driven predictive control in a stochastic setting. *IFAC-PapersOnLine*, 56(2):10083–10088, 2023.
- [9] P. Castillo, R. Lozano, and A. E. Dzul. *Modelling and control of mini-flying machines*. Springer Science & Business Media, 2005.

- [10] C.-C. Chen and Y.-T. Chen. Feedback linearized optimal control design for quadrotor with multi-performances. *IEEE Access*, 9:26674–26695, 2021.
- [11] N. C. Coops, T. R. Goodbody, and L. Cao. Four steps to extend drone use in research. *Nature*, 572(7770):433–435, 2019.
- [12] J. Coulson. *Data-enabled predictive control: Theory and practice*. PhD thesis, ETH Zurich, 2022.
- [13] J. Coulson, J. Lygeros, and F. Dörfler. Data-enabled predictive control: In the shallows of the deepc. In *2019 18th European Control Conference (ECC)*, pages 307–312. IEEE, 2019.
- [14] D.-M. Dobrea and M.-C. Dobrea. An autonomous uav system for video monitoring of the quarantine zones. *Roman J Inf Sci Technol*, 23:S53, 2020.
- [15] W. Favoreel, B. De Moor, and M. Gevers. Spc: Subspace predictive control. *IFAC Proceedings Volumes*, 32(2):4004–4009, 1999.
- [16] J. Garcia and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.
- [17] S. Gros and M. Zanon. Learning for mpc with stability & safety guarantees. *Automatica*, 146:110598, 2022.
- [18] Z.-S. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.
- [19] J. P. How, B. Behnhke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, 28(2):51–64, 2008.
- [20] T. M. Inc. Derive quadrotor dynamics for nonlinear model predictive control, 2024. <https://www.mathworks.com/help/symbolic/derive-quadrotor-dynamics-for-nonlinearMPC.html>.
- [21] F. A. Internationale. What links a dragonfly and the centenary of the first fai rotorcraft record?, 2024. <https://www.fai.org/news/centenary-first-fai-rotorcraft-record-oehmichen>.

- [22] M. Islam, M. Okasha, and E. Sulaeman. A model predictive control (mpc) approach on unit quaternion orientation based quadrotor for trajectory tracking. *International Journal of Control, Automation and Systems*, 17(11):2819–2832, 2019.
- [23] W. Johnson. *Helicopter Theory*. Dover Books on Aeronautical Engineering Series. Dover Publications, 1994.
- [24] J. Kim, S. Kim, C. Ju, and H. I. Son. Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications. *Ieee Access*, 7:105100–105115, 2019.
- [25] M. Lazar and P. C. N. Verheijen. Offset-free data-driven predictive control. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 1099–1104, 2022.
- [26] I. Markovsky and F. Dörfler. Behavioral systems theory in data-driven analysis, signal processing, and control. *Annual Reviews in Control*, 52:42–64, 2021.
- [27] I. Markovsky and P. Rapisarda. Data-driven simulation and control. *International Journal of Control*, 81(12):1946–1959, 2008.
- [28] N. Mohajerin and S. L. Waslander. Multistep prediction of dynamic systems with recurrent neural networks. *IEEE transactions on neural networks and learning systems*, 30(11):3370–3383, 2019.
- [29] M. M. Morato and M. S. Felix. Data science and model predictive control:: A survey of recent advances on data-driven mpc algorithms. *Journal of Process Control*, 144:103327, 2024.
- [30] K. Munson. *Helicopters and Other Rotorcraft Since 1907*. The Pocket encyclopedia of world aircraft in color. Macmillan, 1969.
- [31] K. R. Muske and J. B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993.
- [32] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.
- [33] K. Ogata. *System Dynamics: Pearson New International Edition*. Always learning. Pearson Education Limited, 2013.

- [34] B. Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.
- [35] R. J. A. Schreurs, S. Weiland, H. Tao, Q. Zhang, J. Zhu, Y. Zhu, and C. Xu. Open loop system identification for a quadrotor helicopter system. In *2013 10th IEEE International Conference on Control and Automation (ICCA)*, pages 1702–1707, 2013.
- [36] B. Skorup and C. Haaland. How drones can help fight the coronavirus. *Merca-tus Center Research Paper Series, Special Edition Policy Brief (2020)*, 2020.
- [37] T. V. F. Society. Vertipedia, 2024. <https://vertipedia.vtol.org/aircraft/getAircraft/aircraftID/298>.
- [38] T. V. F. Society. Vertipedia, 2024. <https://vertipedia.vtol.org/aircraft/getAircraft/aircraftID/292>.
- [39] T. V. F. Society. Vertipedia, 2024. <https://vertipedia.vtol.org/aircraft/getAircraft/aircraftID/535>.
- [40] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza. Data-driven mpc for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776, 2021.
- [41] K. P. Valavanis. *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*. Springer Science & Business Media, 2008.
- [42] P. van Overschee and B. de Moor. *Subspace Identification for Linear Systems: Theory — Implementation — Applications*. Springer US, 1996.
- [43] H. J. Van Waarde, C. De Persis, M. K. Camlibel, and P. Tesi. Willems’ fundamental lemma for state-space systems and its extension to multiple datasets. *IEEE Control Systems Letters*, 4(3):602–607, 2020.
- [44] P. Verheijen, V. Breschi, and M. Lazar. Handbook of linear data-driven predictive control: Theory, implementation and design. *Annual Reviews in Control*, 56:100914, 2023.
- [45] P. Verheijen, G. G. da Silva, and M. Lazar. Data-driven rate-based integral predictive control with estimated prediction matrices. In *2021 25th International*

- Conference on System Theory, Control and Computing (ICSTCC)*, pages 630–636. IEEE, 2021.
- [46] D. K. Villa, A. S. Brandao, and M. Sarcinelli-Filho. A survey on load transportation using multirotor uavs. *Journal of Intelligent & Robotic Systems*, 98:267–296, 2020.
 - [47] J. Wang, K. A. Alattas, Y. Bouterraa, O. Mofid, and S. Mobayen. Adaptive finite-time backstepping control tracker for quadrotor uav with model uncertainty and external disturbance. *Aerospace Science and Technology*, 133:108088, 2023.
 - [48] K. Wang, C. Hua, J. Chen, and M. Cai. Dual-loop integral sliding mode control for robust trajectory tracking of a quadrotor. *International Journal of Systems Science*, 51(2):203–216, 2020.
 - [49] J. C. Willems. The behavioral approach to open and interconnected systems. *IEEE Control Systems Magazine*, 27(6):46–99, 2007.
 - [50] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor. A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329, 2005.
 - [51] S. Yang, Z. Zou, Y. Li, H. Shi, and Q. Fu. Adaptive fault-tolerant tracking control of quadrotor uavs against uncertainties of inertial matrices and state constraints. *Drones*, 7(2):107, 2023.
 - [52] C. Yuan, Z. Liu, and Y. Zhang. Aerial images-based forest fire detection for fire-fighting using optical remote sensing techniques and unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 88:635–654, 2017.
 - [53] M. Zanon and S. Gros. Safe reinforcement learning using robust mpc. *IEEE Transactions on Automatic Control*, 66(8):3638–3652, 2020.
 - [54] D. ZHURILO, M. Gutnyk, and A. ZHURILO. George de bothezat and his contribution into the world aviation and astronautics. *Kosmìčna nauka ì tehnologiâ*, 28:70–80, 02 2022.



APPENDIX A

EXPLICIT MATRIX EXPRESSIONS FOR THE MATHEMATICAL MODEL

A.1 The Matrix for the Rotational Kinetic Energy

The matrix J is the matrix for the rotational kinetic energy of the quadrotor expressed directly in terms of the generalized coordinates η . Then, the matrix J can be written as,

$$J = T^\top I T \quad (\text{A.1})$$

$$J = \begin{bmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \end{bmatrix} \quad (\text{A.2})$$

Since its elements are long to write in Chapter 2, the elements of the matrix J are given below.

$$j_{11} = I_{xx} \quad (\text{A.3})$$

$$j_{12} = 0 \quad (\text{A.4})$$

$$j_{13} = -I_{xx} \sin \theta \quad (\text{A.5})$$

$$j_{21} = 0 \quad (\text{A.6})$$

$$j_{22} = I_{yy} \cos^2 \phi + I_{zz} \sin^2 \phi \quad (\text{A.7})$$

$$j_{23} = (I_{yy} - I_{zz}) \cos \phi \sin \phi \cos \theta \quad (\text{A.8})$$

$$j_{31} = -I_{xx} \sin \theta \quad (\text{A.9})$$

$$j_{32} = (I_{yy} - I_{zz}) \cos \phi \sin \phi \cos \theta \quad (\text{A.10})$$

$$j_{33} = I_{xx} \sin^2 \theta + I_{yy} \sin^2 \phi \cos^2 \theta + I_{zz} \cos^2 \phi \cos^2 \theta \quad (\text{A.11})$$

A.2 Coriolis Matrix

The variable $C(\mu, \dot{\mu})$ referring to the Coriolis terms and gyroscopic and centrifugal terms associated with the η dependence of J is introduced before. Since its elements are long to write in Chapter 2, the elements of the matrix C are given below.

$$C(\mu, \dot{\mu}) = \dot{J} - \frac{1}{2} \frac{\partial}{\partial \eta} \left(\dot{\eta}^\top J \right) \quad (\text{A.12})$$

$$C(\mu, \dot{\mu}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \quad (\text{A.13})$$

where

$$c_{11} = 0 \quad (\text{A.14})$$

$$c_{12} = (I_{yy} - I_{zz}) \left(\dot{\theta} \cos \phi \sin \phi + \dot{\psi} \sin^2 \phi \cos \theta \right) + (I_{zz} - I_{yy}) \dot{\psi} \cos^2 \phi \cos \theta - I_{xx} \dot{\psi} \cos \theta \quad (\text{A.15})$$

$$c_{13} = (I_{zz} - I_{yy}) \dot{\psi} \cos \phi \sin \phi \cos \theta^2 \quad (\text{A.16})$$

$$c_{21} = (I_{zz} - I_{yy}) \left(\dot{\theta} \cos \phi \sin \phi + \dot{\psi} \sin^2 \phi \cos \theta \right) + (I_{yy} - I_{zz}) \dot{\psi} \cos^2 \phi \cos \theta + I_{xx} \dot{\psi} \cos \theta \quad (\text{A.17})$$

$$c_{22} = (I_{zz} - I_{yy}) \dot{\phi} \cos \phi \sin \phi \quad (\text{A.18})$$

$$c_{23} = -I_{xx} \dot{\psi} \sin \theta \cos \theta + I_{yy} \dot{\psi} \sin^2 \phi \cos \theta \sin \theta + I_{zz} \dot{\psi} \cos^2 \phi \cos \theta \sin \theta \quad (\text{A.19})$$

$$c_{31} = (I_{yy} - I_{zz}) \dot{\theta} \cos \phi \sin \phi \cos \theta - I_{xx} \dot{\theta} \cos \theta \quad (\text{A.20})$$

$$c_{32} = (I_{zz} - I_{yy}) \left(\dot{\phi} \cos \phi \sin \phi + \dot{\psi} \sin^2 \phi \cos \theta \right) + (I_{yy} - I_{zz}) \dot{\psi} \cos^2 \phi \cos \theta + I_{xx} \dot{\psi} \cos \theta \quad (\text{A.21})$$

$$c_{33} = (I_{yy} - I_{zz}) \dot{\phi} \cos \phi \sin \phi \cos \theta^2 - I_{yy} \dot{\theta}^2 \sin^2 \phi \cos \theta \sin \theta - I_{zz} \dot{\theta}^2 \cos^2 \phi \cos \theta \sin \theta + I_{xx} \dot{\theta} \cos \theta \sin \theta \quad (\text{A.22})$$

APPENDIX B

SETTINGS FOR LINEAR AND NONLINEAR MPC

Table B.1: Linear MPC settings for the trajectory tracking

Parameter	Value
Sampling rate (T_s) [s]	0.1
Prediction horizon (N) [-]	30
State weight matrix (Q)	$\begin{bmatrix} 5000 \otimes I_2 & 0 & \dots & \dots & \dots & 0 \\ 0 & 1000 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 50 \otimes I_3 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 5000 \otimes I_2 & \ddots & \vdots \\ \vdots & \dots & \dots & \dots & 1000 & \vdots \\ 0 & \dots & \dots & \dots & \dots & 50 \otimes I_3 \end{bmatrix}$
Input weight matrix (R)	$50 \times I_4$
Input lower BC (u_l)	$0 \times 1_4$
Input upper BC (u_u)	$10 \times 1_4$

Table B.2: Nonlinear MPC settings for the trajectory tracking

Parameter	Value
Sampling rate (T_s) [s]	0.1
Prediction horizon (N) [-]	18
Control horizon (N_c) [-]	2
State weight matrix (Q)	$\begin{bmatrix} 1000 \otimes I_2 & 0 & \dots & \dots & \dots & 0 \\ 0 & 700 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 \otimes I_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 100 & \ddots & \vdots \\ \vdots & \dots & \dots & \dots & 0 & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 \end{bmatrix}$
Input weight matrix (R)	$50 \times I_4$
Input lower BC (u_l)	$0 \times 1_4$
Input upper BC (u_u)	$10 \times 1_4$