Data-Driven Control Strategies for Rotary Wing Aerial Vehicles

Simão Fernandes Caeiro

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors:	Prof. Rita Maria Mendes de Almeida Correia da Cunha
	Prof. Bruno João Nogueira Guerreiro

July 2023

To my Little Star that never leaves me.

Declaration
I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First of all, I would like to express my sincere gratitude to my dissertation supervisors, Prof. Bruno Guerreiro and Prof. Rita Cunha, for their invaluable guidance, support, and expertise throughout this thesis.

Also, I would like to extend this thanks to João Pinto, Pedro Trindade and Renato Loureiro, for their assistance and contributions to this work.

To all my friends and family, thank you for all the fantastic and unforgettable experiences that we have shared together. A special thanks to my father, for being my greatest role model, whose guidance and wisdom have helped me to achieve my goals, and to my sister, for all the patience and encouragement.

Finally, I would like to deeply thank João, for his friendship and all the memorable experiences shared over these last five years, and Mariana, for all her unconditional support, understanding and love, which were the pillars that sustained me throughout this journey. I am eternally grateful for their presence in my life.

This work was partially funded by FCT projects CAPTURE (PTDC/EEI-AUT/1732/2020) and LARSYS (UIDB/50009/2020).

Resumo

A popularidade dos veículos aéreos não tripulados (UAVs) tem aumentado de forma significativa nos últimos anos, devido à sua estabilidade, resistência e versatilidade, com impacto em vários domínios militares e civis. Tradicionalmente, o controlo de UAVs tem-se fundamentado em abordagens baseadas em modelos. Contudo, a crescente complexidade dos sistemas e a ampla gama de dados disponíveis conduziram ao estudo e desenvolvimento de estratégias baseadas em dados que não requerem uma representação exata do modelo a controlar. De modo a avaliar o desempenho destas estratégias aplicadas a UAVs, esta tese foca-se na implementação e análise de métodos de controlo baseado em dados, com particular destaque para o algoritmo Data-enabled Predictive Control (DeePC). Este método de controlo preditivo calcula controlos ótimos para sistemas desconhecidos, através da realimentação da saída em tempo real e da implementação de um horizonte deslizante. Para além disso, este trabalho estuda a influência de diferentes hiperparâmetros no desempenho do algoritmo DeePC e realiza uma comparação realista entre este método e duas estratégias de controlo baseadas em modelos: o Linear Quadratic Regulator (LQR) e o Model Predictive Control (MPC). Os resultados de simulação ilustram a aplicabilidade e desempenho que pode ser obtido com o algoritmo DeePC. Adicionalmente, quando comparado com as abordagens convencionais implementadas, este método revela-se mais robusto à degradação do desempenho do anel interno do sistema e adapta-se melhor a um erro de calibração do ângulo de guinada. Contudo, a implementação do método DeePC torna-se inapropriada para sistemas não lineares mais complexos, sujeitos a trajetórias agressivas.

Palavras-chave: Veículos Aéreos Não Tripulados, Controlo Baseado em Dados, Data-enabled Predictive Control, Avaliação de Performance, Robustez

Abstract

Unmanned Aerial Vehicles (UAVs) have significantly grown in popularity due to their enhanced stability, endurance, and versatility in various military and civilian domains. Traditionally, the control of UAVs has relied on model-based approaches, but the increasing complexity of the systems and the widespread availability of data have led to the research and development of data-driven strategies that do not require an accurate representation of the system to be controlled. To evaluate the performance of data-driven control methods applied to UAVs, this thesis addresses the implementation and analysis of these strategies, in particular, the Data-enabled Predictive Control (DeePC) algorithm. This predictive control strategy computes optimal controls for unknown systems through real-time output feedback using a receding horizon implementation. Moreover, this work investigates the influence of different hyperparameters on the performance of the DeePC algorithm and conducts a realistic comparison between this method and two model-based control approaches: the Linear Quadratic Regulator (LQR) and the Model Predictive Control (MPC). The simulation results confirm the applicability of the DeePC algorithm and illustrate the performance that can be obtained. In addition, this method proves to be more robust to the performance degradation of the system's inner loop and adapts better to a yaw calibration error, when compared to the implemented conventional approaches. However, for more complex nonlinear systems subject to aggressive trajectories, the implementation of the DeePC method becomes inappropriate.

Keywords: Unmanned Aerial Vehicles, Data-driven Control, Data-enabled Predictive Control, Performance Evaluation, Robustness

Contents

List of Tables

List of Figures

Acronyms

3DR	3D Robotics
ARMAX	Autoregressive Moving Average with Exogenous Inputs
BEM	Blade-Element-Momentum
DeePC	Data-Enabled Predictive Control
EKF	Extended Kalman Filter
GP	Gaussian Process
GS-PID	Gain-Scheduled PID
IMU	Inertial Measurement Unit
LQR	Linear Quadratic Regulator
LS	Least Squares
LTI	Linear Time-Invariant
MFC	Model-Free Control
ML	Maximum Likelihood
MPC	Model Predictive Control
MRAC	Model Reference Adaptive Control
NN	Neural Network
PD	Proportional Derivative
PID	Proportional Integrative Derivative
PRBS	Pseudorandom Binary Sequence
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RoKDeePC	Robust and Kernelized DeePC

SITL	Software in the Loop
SMC	Sliding Mode Control
SPC	Subspace Predictive Control
SVD	Singular Value Decomposition
UAV	Unmanned Aerial Vehicle
VRFT	Virtual Reference Feedback Tuning
ZOH	Zero-Order Hold

Chapter 1

Introduction

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) have grown significantly in popularity and notoriety in recent years, largely attributable to their enhanced stability and endurance in several operations. Initially developed on a large scale during World War I, these systems were primarily used for military applications such as reconnaissance, airfield security, surveillance and target acquisition. Moreover, numerous civil applications have emerged over time, encompassing domains such as disaster management, education, environmental and climate studies, construction and infrastructure inspection, tourism, mapping, precision agriculture, meteorology, real-time traffic monitoring, and many others [1].

Since 2020, this type of vehicle has played a vital role in the COVID-19 pandemic, being used to deliver vaccines, medical supplies, and viral tests to potentially infected patients [2, 3]. Figure 1.1 depicts an example of a quadrotor adapted to COVID-19 tests distribution. More recently, in 2022, quadrotors have been extensively deployed in the Russia-Ukraine War, thereby marking a new era in air warfare characterised by direct drone combat.

Figure 1.1: Retrofitted quadrotor for COVID-19 tests distribution [2].

A UAV is commonly described as a pilotless aircraft with the capability to fly and stay airborne without requiring any human onboard operator. These attributes enable UAVs to perform critical missions without risking human life and provide more cost-effective operations than equivalent manned systems. Although UAVs can be remotely piloted, numerous research groups have developed several control design methods that enable these vehicles to operate autonomously. Hence, the control of these systems is conventionally tackled through a model-based approach, whereby an accurate representation of the system is initially extracted from data, and subsequently, the control strategy is formulated based on the identified model. Notwithstanding, with the increasing complexity of systems and the widespread availability of data, there has been a recent trend in the literature where classical model-based techniques have been superseded by data-driven methodologies [4]. These data-based techniques are appropriate for scenarios where first-principle models are not feasible, where the models are excessively complex for control design, and where modelling and identification of parameters are too costly.

The Data-enabled Predictive Control (DeePC), as presented in [5], is an example of a data-driven control technique, whose main goal is to learn the system's behaviour rather than attempting to learn a parametric model of the system. This predictive control strategy computes optimal controls for unknown systems through real-time output feedback, using a receding horizon implementation. It is worth noting that this methodology is much simpler to implement than model-based approaches that entail state observer design and system identification.

1.2 Objectives

The main objective of this work is to evaluate the performance and key elements of data-driven control techniques applied to quadrotors. To achieve this goal, the study and development of one of these methods is required. Following a detailed analysis of the various steps of the method under study, it is further intended to validate its performance through realistic simulation experiments.

Additionally, a comparison between the data-driven control strategy and other conventional control techniques can also help to clarify the advantages and disadvantages of this type of strategy.

1.3 Proposed Solutions and Contributions

To achieve the aforementioned objectives, the DeePC algorithm presented in [5] is implemented, adapted to the problem at hand and analysed in detail. Subsequently, a set of realistic simulation experiments is conducted, which take into consideration uncertainties, constraints, and nonlinearities of the quadrotor system. The performance of the DeePC algorithm is also evaluated on different models of varying complexities, whereas a novel control architecture is developed and tested on a different vehicle to the one used by the authors of the DeePC method.

Furthermore, a comprehensive examination of the influence of the data collection step and the selection of each hyperparameter of the algorithm is presented and, finally, a comprehensive and realistic comparison is performed between DeePC and two conventional model-based control methods of different complexity, designated by Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC). This comparative analysis includes an assessment of the algorithms' response to a simple trajectory and their adaptability to performance degradation of the system's inner loop and yaw calibration errors in the system measurements.

1.4 Thesis Outline

The remainder of this thesis is structured as follows:

- **Chapter 2 (Related Work):** presents a comprehensive overview of the state-of-the-art in control methods design, with a particular emphasis on classic model-based control techniques, system identification strategies, data-driven control approaches, and control methods based on machine learning techniques.
- **Chapter 3 (Theoretical Background):** provides the necessary theoretical background on the key concepts to consider throughout this dissertation. A description of both the linear and nonlinear models of the quadrotor's dynamics is presented, followed by a review of the conventional model-based methods relevant to this work.
- **Chapter 4 (Data-enabled Predictive Control):** formulates the optimisation problem associated with the DeePC algorithm for both deterministic LTI and nonlinear systems. The implementation of the data collection step is also detailed in this chapter.
- **Chapter 5 (Implementation and Basic Simulation Results):** describes the implementation and simulation setup used to test the proposed algorithms. Subsequently, a set of simulated step responses using the DeePC controller implemented in different control architectures is presented and analysed.
- **Chapter 6 (DeePC Performance Results):** studies the influence of several parameters on the simulated performance of the DeePC algorithm. A comparison between LQR, MPC and DeePC control methods is also discussed.
- **Chapter 7 (Conclusions):** provides some concluding remarks and suggestions for future work.

Chapter 2

Related Work

In this chapter, an overview of the research conducted on UAV control methods is presented. Firstly, classical control methods are briefly summarised, followed by an examination of system identification techniques, which are essential in the implementation of model-based control approaches. Next, data-driven control techniques are discussed, with a particular focus on the methods utilised in this thesis. Lastly, data-driven control methods based on machine learning strategies are discussed.

2.1 Classic Model-based Methods

UAVs were developed primarily for aerial operations without the need for a human operator, offering more cost-effective operations than equivalent manned systems and, essentially, carrying out vital duties without endangering human life. Therefore, much of the early work on UAVs developments focused on reliable control of the system, which was considered to be an exciting challenge, mainly because these vehicles exhibit characteristics of under actuation, nonlinearity, static instability, and strong coupling between dynamic states [6]. This section provides an overview of classical control algorithms, including linear and non-linear techniques.

In [7], Åström and Hägglund introduced the basic concepts of Proportional Integrative Derivative (PID) control, describing the properties of the controller in a closed-loop system. The design and tuning of these controllers were then examined, further demonstrating that they can be applied in a variety of real-world applications. Regarding UAVs, Bouabdallah et al. designed a PID controller for a fully autonomous micro quadrotor in [8], [9], and [10]. They used the Euler-Lagrange formulation to build the dynamic system model of the quadrotor, which included gyroscopic effects. The PID controller was then applied to a quadrotor test bench and the results were compared with a modern control strategy. The experimental results proved that the PID controller successfully stabilised the attitude dynamics, achieving success in hovering with the presence of small disturbances. In [11], Hoffmann et al. investigated the issues that arise when a quadrotor operates outside of a hover position. They focused on the analysis of three different aerodynamic factors: vehicular velocity, angle of attack, and airframe design. Through theoretical analysis and experiments using a test stand and actual flight tests, they also verified the effectiveness of a Proportional Derivative (PD) controller in regulating the vehicle's pitch during manoeuvres at low speeds. However, at higher speeds, blade flapping created additional control issues that required alternative approaches. On the other hand, position control was successfully implemented using a PID controller that could stabilise roll and pitch control inputs. Furthermore, according to their conclusions, the existing models and control methods were inadequate for tracking at high speeds and in unpredictable environments with wind or other disturbances. T. Zhang et al. proposed a control framework for achieving autonomous hovering, based on pose estimation and control, marker design, image processing, and Inertial Measurement Unit (IMU) [12]. The framework employed a closed-loop system with four PID controllers that utilised feedback from pose estimation to achieve stable hovering at a fixed altitude. The authors also experimentally tested the efficacy of their control approach in real-time by hovering over markers with some small oscillations. Overall, this work demonstrated the practical viability of the proposed control framework for autonomous hovering in quadrotors.

In [13], Sadeghzadeh et al. further expanded the basic PID control by incorporating fault-tolerant control strategies. They compared two different methods, designated as Model Reference Adaptive Control (MRAC) and Gain-Scheduled PID (GS-PID), for handling damage in a quadrotor, specifically partial damage to one of the propellers during flight. Through experiments, they demonstrated that both the MRAC and GS-PID methods were effective in compensating for the damage during hovering and flight conditions. Goodarzi et al. presented a nonlinear PID controller, in [14], that was designed to track position and attitude commands while taking into account uncertainties that exist in the quadrotor's translation and rotation dynamics. This controller was developed using a special Euclidean group and included a new integral term that allowed for asymptotic convergence of tracking errors in the presence of uncertainties in the quadrotor's dynamics. The simulation and preliminary experimental results demonstrated the utility of these methods. Yang et al. presented a PD controller for a quadrotor based on a dual closed-loop control framework [15]. In this work, active disturbance rejection control and PD control strategies are applied to the inner and outer loops, respectively. The gust wind perturbations were estimated in the inner loop, and both convergence and stabilisation were achieved for the closed-loop system. A simple PD control strategy was used for the control of attitude angles. Then, the stabilisation of the inner and outer closed-loop system was proved using the Lyapunov theory. The proposed controller was validated through experimental results, which demonstrated its effectiveness in handling wind disturbances.

To address some limitations of linear PID, a more flexible control strategy, designated as LQR, was proposed. In [16], Valenti et al. suggested an LQR-based control for position and attitude. This work focused on issues related to single and multi-vehicle health management. The control system was experimentally tested for hovering and waypoint tracking. Cowling et al. simulated a simple path-following LQR controller [17]. The simulation results demonstrated that the quadrotor was capable of achieving an accurate tracking of the desired reference trajectory, despite the presence of modelled wind and other disturbances. In [18], Yu et al. compared the performance of LQR and MPC methods applied to a quadrotor. Both control strategies were evaluated in scenarios with and without an actuator fault. The static error was taken into consideration during the design of the LQR-based control algorithm. Hence, the feedback control was implemented to ensure that the quadrotor followed a reference input without a static error. The simulations demonstrated that these control algorithms were capable of delivering satisfactory performance in both fault-free and actuator fault scenarios. Martins et al. proposed a control structure for trajectory tracking of UAVs employing an inner-outer loop design that included an LQR controller with integrative action [19]. This control structure produced effective trajectory tracking results and demonstrated robustness to perturbances in both the simulation and the real system.

Furthermore, the H_∞ control technique was also employed to improve the quadrotor system's robustness to external disturbances. In [20], Chen and Huzmezan created an H_∞ linearised controller to manage the quadrotor's velocities, throttle, and yaw. They integrated the H_∞ controller with an MPC controller to increase the system's constraint-handling ability during high-speed manoeuvres. To address the issue of trajectory tracking, the suggested H_∞ controller was optimised using a loop shaping technique to stabilise the velocities, throttle, and yaw control. The simulation demonstrated that the combination of these two techniques was effective for a wide range of trajectory scenarios. A generalised H_∞ controller with feedback linearisation to address actuator saturation was introduced by Mokhtari et al. [21]. This method was effective in tracking reference inputs for a nonlinear quadrotor system. Through simulation, the performance of the proposed controller was evaluated by examining the tracked error trajectories, which showed that the quadrotor system can handle disturbances and uncertainties in mass and inertia, despite limited actuator saturation. Lastly, in [22, 23], Raffo et al. presented an H_∞ control technique for solving the path-tracking problem. The control structure in both works relied on a nonlinear H_∞ controller, combined with other strategies, to enable path following even in the presence of external disturbances and modelling errors. The robustness of this controller was demonstrated in simulation tests.

Although various works demonstrated the effectiveness of tackling the control problem with linear techniques, it was found that applying nonlinear control methods that considered a more comprehensive model of vehicle dynamics led to better performance. To overcome some shortcomings of these linear controllers, a variety of nonlinear approaches applied to quadrotors can be found in the literature.

One of these nonlinear control techniques is the Backstepping method. The Backstepping control approach is a recursive technique that integrates the selection of a Lyapunov function with the design of feedback control. It decomposes the design problem for the complete system into a sequence of problems for lower-order systems. This approach can often resolve stabilisation, tracking, and robust control issues with less restrictive conditions than other methods [24]. Bouabdallah and Siegwart presented implementations of control strategies for the quadrotor using the Backstepping technique in [25, 26]. In [25], they proposed a Backstepping control approach for the quadrotor based on tracking position errors and utilising the Lyapunov theorem. Subsequently, they designed an improved controller through an integral Backstepping method in [26]. Both articles presented simulation and experimental outcomes. The proposed controller in [25] demonstrated effective tracking of the target position and heading angle. On the other hand, the control strategy proposed in [26] achieved favourable results for not only position control but also attitude and altitude controls. In [27], Madani and Benallegue employed the Backstepping method in a simulation to stabilise the system and effectively track a desired trajectory and yaw angle. They partitioned the quadrotor model into three interconnected subsystems. The simulation results demonstrated the robust performance of the proposed control strategy. Lippiello et al. published a research work based on the emergency landing of a quadrotor with a failed propeller [28]. The method involved turning off the opposing rotor to the broken one, effectively making the quadrotor a birotor, and using the Backstepping approach for control. While the birotor control allowed the quadrotor to successfully follow the planned emergency path, the performance of the control approach showed that yaw and roll angles were uncontrollable but bounded. The simulation results revealed promising path-following performance in various cases, including a simulation with an obstacle.

Subsequently, Sliding Mode Control (SMC) is another nonlinear methodology that is extensively employed. This control technique is characterised by the application of a discontinuous control signal to induce the system state to slide along a predefined manifold. The sliding mode is reached when the system state trajectory intersects with the sliding manifold, resulting in robustness to disturbances and uncertainties [24]. In [25], Bouabdallah and Siegwart implemented the SMC on a quadrotor by designing it for rotation subsystem control. The sliding surface was defined and confirmed using the Lyapunov theory. The proposed SMC's efficacy was compared to the Backstepping controller and the latter was determined to be superior to the SMC. Xu and Ozguner presented an SMC approach to stabilise a quadrotor under model error, parametric uncertainties, and other disturbances [29]. The objective of the proposed controller was to enable the quadrotor to reach a desired position with a specific heading angle. Furthermore, the proposed controller utilised a continuous approximation of the sign function to avoid the chattering effect. The simulation outcomes demonstrated that the proposed controller effectively achieved the desired objective with satisfactory outputs, even in the presence of parametric uncertainties. In [30], López-Gutiérrez et al. presented an adaptive SMC that was combined with robust attitude control. The proposed controller incorporated an adaptation rule in the control law, which reduced the gain while maintaining minimal control input and ensuring finite-time convergence. The simulation and experimental results indicated the effectiveness of the proposed controller, in the presence of external disturbances. Moreover, the study concluded that the proposed controller succeeded in reducing the chattering amplitude by minimising the gain.

On the other hand, the Feedback Linearisation method is also a nonlinear control technique, whose main objective is to algebraically transform the dynamics of the nonlinear system into a partially or fully linearised system so that linear control techniques can be applied [31]. Altug et al. introduced techniques for controlling quadrotors by primarily relying on visual feedback [32]. Their study involved implementing Feedback Linearisation and Backstepping controllers using various simulations of the model to validate their findings. Hence, these simulations revealed that the Backstepping controller outperformed the Feedback Linearisation controller. In [33], Lee and Sastry presented two types of nonlinear controllers for an autonomous quadrotor. These comprised a Feedback Linearisation controller involving high-order derivative terms and an adaptive SMC. According to their findings, the first method was sensitive to sensor noise and modelling uncertainty, in contrast to the adaptive SMC. In [34], Voos introduced a control system for a quadrotor, which combined various control strategies. These included Feedback Linearisation to handle the quadrotor's nonlinear dynamic behaviour. The simulation results demonstrated satisfactory performance in controlling the quadrotor's attitude using the suggested Feedback Linearisation technique.

Lastly, MPC is a control approach that employs a system's dynamic model to predict its future behaviour and optimise a control input over a finite time horizon, while considering the constraints. The control input is then applied to the system for a given period and the process is repeated at each time step [35]. In [36], Alexis et al. proposed using the MPC technique to design a position and attitude controller for a quadrotor in an indoor setting where there was no absolute localisation data available. Their approach aimed to achieve precise trajectory control. In a series of experiments that included tracking positions, hovering, aggressive attitude regulation manoeuvres, and mitigating the effects of strong wind gusts, the proposed control strategy demonstrated significant overall effectiveness. In [37], Abdolhosseini et al. developed an efficient MPC algorithm, which utilises a reduced number of prediction points and demands less computational resources. The experimental outcomes demonstrated that the suggested approach achieved a satisfactory performance when tested with the quadrotor. In [38], Pinto suggested planning and control strategies, with a focus on MPC techniques, to enable safe and efficient exchange of a parcel between two drones during flight. This work presented a model predictive controller that plans trajectories in real-time and its effectiveness was verified through simulations. Fang et al. introduced a fault-tolerant controller based on a nonlinear MPC algorithm to stabilise and control a quadrotor that experiences a total failure of one rotor [39]. The proposed method was validated through extensive simulations and real-world experiments. The results revealed that the approach was successful in restoring the quadrotor even if the failure occurred during aggressive manoeuvres.

Table 2.1 details the summary of the model-based control methods presented in this section.

2.2 System Identification

In Section 2.1, it was discussed that a quadrotor dynamic system can be controlled using model-based approaches that rely on explicit mathematical models of the system. Therefore, this section is dedicated to the presentation of system identification techniques. The fundamental concept of these techniques is to observe the behaviour of the dynamic system over a specific period and collect data on the measurements of the system's input and output to develop a mathematical model that describes its behaviour [40].

Primarily, one of the most widely employed time-domain approaches for system identification is the Least Squares (LS) method, which relies on Gauss method of minimising the summation of a sequence of squared terms [40]. In [41], Gremillion and Humbert reported on the development of a linear mathematical model to estimate the dynamics of a quadrotor micro air vehicle by implementing the LS method on vehicle flight test data. This study employed time domain system identification software developed at NASA Langley Research Center. Another study of the estimation of dynamic parameters for a quadrotor using the LS method was presented by Lopez-Sanchez et al. [42]. They aimed to improve upon the nominal parameters provided by the manufacturer and used an optimised trajectory to excite the quadrotor's dynamics and gather experimental data. The research highlighted the effectiveness of the proposed method in accurately identifying the quadrotor's parameters. In [43], Six et al. proposed a novel approach to determine the propeller coefficients and dynamic parameters of quadrotor drones in a single procedure. This method employs an LS identification technique and only necessitates the measurement of the quadrotor's mass, alongside a manual recording of flight data by an operator. The efficacy of this approach was tested through experimental validation, where it enabled an estimation of all pertinent dynamic parameters of the drone in proximity to hovering conditions.

Based on an additional statistical method, designated as Maximum Likelihood (ML), Burri et al. proposed a systematic approach to accurately estimate physical parameters [44]. This approach was designed to determine the parameters that provided the optimal explanation of sensor readings and also furnished an estimate of their corresponding level of uncertainty. The efficacy of the approach was demonstrated through an extensive evaluation of both simulated and real-world experimental data, exhibiting a substantial convergence region and producing accurate estimates. To implement a robust tracking controller, Rigter et al. described in [45] a system identification approach that identifies specific quadrotor parameters via ML estimation from flight data. This method was subsequently validated through experimental results.

Furthermore, Autoregressive Moving Average with Exogenous Inputs (ARMAX) models have found extensive application in system identification due to their ability to describe a broad range of real-world processes with reasonable accuracy and relatively low complexity. In [46], Schreurs et al. were able to obtain an ARMAX model that effectively described the behaviour of a quadrotor system. In [47], Angarita et al. implemented an ARMAX model to accurately design a generic quadrotor system model, with the primary aim of achieving accurate control over complex quadrotor manoeuvres.

Finally, Abas et al. implemented an Extended Kalman Filter (EKF), a commonly used estimation method, to model and identify system parameters for quadrotor self-stabilisation [48]. The flight test results demonstrated the EKF's efficacy in terms of performance.

The summary of the system identification techniques discussed in this section is provided in Table 2.2.

2.3 Data-driven Control

The system identification task, described in the preceding section, is frequently the most time-consuming and challenging component of model-based control methodologies. An inattentive system identification has the potential to engender substandard performance of the implemented model-based control method. Therefore, this stage usually needs expert knowledge and partial models of the system. In the literature, a propensity towards approaches that design control inputs directly from data has been observed, bypassing conventional model-based control techniques. These approaches, designated as data-driven control methods, are particularly advantageous in complex systems where system identification is excessively time-consuming and unwieldy, hence rendering them simpler to execute.

In [49], Willems et al. introduced the *Fundamental Lemma* which explained how to replace a system model with data in the implementation of data-driven control methods for Linear Time-Invariant (LTI) systems. This research proposed that a single input-output trajectory with a finite length could serve as a parametrisation for all feasible paths that an LTI system can generate, given that the input sequence excited the system dynamics sufficiently, which is technically known as persistency of excitation.

In [50], Markovsky and Rapisarda introduced a technique that utilises the *Fundamental Lemma* to compute the system response to a given input and initial conditions solely from a system trajectory, without requiring explicit identification of the system from the data. They proposed a method for computing a linear quadratic tracking control signal that bypasses the identification stage. Nevertheless, like the conventional approach, this technique entails deriving a model representation from given plant data and synthesising a control law based on the model and control specifications. The results were obtained assuming exact data and the simulated response or control input was constructed offline.

A new controller tuning method called Virtual Reference Feedback Tuning (VRFT) was described by Campi et al. in [51]. This approach involves the computation of a virtual reference signal based on the system's measured output. The controller is then optimised by minimising a performance criterion that compares the real system output to the virtual reference model output. Simulation results demonstrated the effectiveness of this method. In [52], Panizza et al. employed the VRFT method to address the challenge of tuning a cascade attitude control system of a variable-pitch quadrotor. In this work, the inner and outer loops were adjusted using only one set of experimental data. The resultant data-driven controller demonstrated effective tracking and disturbance rejection capabilities.

Younes et al. applied the Model-Free Control (MFC) technique to a quadrotor vehicle in [53, 54]. The main objective of MFC is to provide compensation for time-varying disturbances and unmodeled system dynamics that are beyond the capabilities of a basic controller to effectively handle. In [53], they compared the performance of an LQR feedback controller with and without MFC on a real quadrotor. Flight test results demonstrated that the use of MFC was critical for controlling the quadrotor system, particularly when the feedback controller was not functioning optimally. In [54], the authors proposed an integrated structure combining the nonlinear Integral Backstepping technique and the MFC. The proposed combination was validated through flight tests in real-time, which revealed its robust performance compared to other algorithms in both fault-free and actuator fault conditions.

Furthermore, the Subspace Predictive Control (SPC) method was introduced by Favoreel et al. in [55]. The primary outcome of this method was the replacement of the system identification and controller parameter computation steps with a process that involves the QR-decomposition and Singular Value Decomposition (SVD) of a matrix created solely from input and output measurements of the unknown system.

Moreover, the primary control technique explored in this thesis, the DeePC method, was introduced by Coulson et al. in [5]. The algorithm, which is based on the *Fundamental Lemma*, eliminates the need for function learning or system identification. Instead, the algorithm employs prior input/output data to anticipate future trajectories directly. These previously recorded input/output data points, which serve as motion primitives, establish the foundation for the range of feasible system trajectories. To summarise, the DeePC algorithm utilises a finite dataset to learn the unknown system's behaviour and then applies real-time output feedback to compute optimal controls that guide the system towards a desired path, while respecting system constraints. In [5], it was formally showed that the DeePC technique is equivalent to the classical and widely used MPC algorithm for deterministic LTI systems. For nonlinear stochastic systems, the authors suggested regularisations to this method. The simulation results indicated that the DeePC approach outperformed system identification followed by MPC. In [56], Elokda et al. exhibited the effectiveness of the regularised DeePC algorithm in the position control of a real-world quadrotor, thereby establishing a connection between theoretical developments and practical applications. As part of this work, a sensitivity analysis of the DeePC algorithm's hyperparameters was conducted in simulation, resulting in valuable insights into their impact. The real-world deployment of the DeePC method demonstrated its computational efficiency and ability to provide real-time solutions, with solution times well within the required timeframe. In [57], Coulson et al. proposed a suitable regularisation to the DeePC algorithm, which resulted in end-to-end distributional robustness against uncertainties in the data. In [58], Huang et al. introduced a novel data-driven approach for nonlinear systems, known as Robust and Kernelised DeePC (RoKDeePC), which combines kernel methods with a robust DeePC framework. The authors employed regularised kernel methods to learn the future behaviour of nonlinear systems implicitly, followed by the formulation of a data-to-control min-max optimisation problem to obtain robust and optimal control sequences. They proved that the cost of the system is bounded by the optimisation cost of this min-max problem, ensuring deterministic performance guarantees. Furthermore, it was shown that the min-max problem can be reformulated as a nonconvex yet structured minimisation problem when uncertainty sets are considered. To enable real-time implementation, a projected gradient descent algorithm exploiting the problem structure was devised to solve the RoKDeePC problem efficiently. Unlike learning-based control methods using Neural Networks (NN), this approach does not necessitate time-consuming offline learning, and it can handle nonlinear systems in real time with a plug-and-play ability.

Finally, Waarde presented a new experiment design method for data-driven modelling and control in [59]. The main concept involved the online selection of inputs based on past input/output data, which resulted in desirable rank properties of Hankel data matrices. When compared to the traditional persistency of excitation condition, this online method required fewer data samples and was shown to be entirely sample efficient.

Table 2.3 outlines a comprehensive summary of the data-driven control methods presented in this section.

2.4 Machine Learning Methods

The burgeoning interest in machine learning techniques has given rise to various approaches in data-driven control for quadrotors. These approaches, as discussed in the preceding section, are primarily directed towards resolving the issues encountered with traditional model-based control techniques.

Dierks and Jagannathan used neural networks in quadrotors, as reported in [60, 61]. In these works, the authors presented an output feedback controller for a quadrotor based on NN. The primary objective of this control strategy was to enable the quadrotor to follow a desired trajectory in the presence of model uncertainties and other disturbances. Lyapunov theory was employed to demonstrate that the proposed control strategy successfully constrained errors in position, orientation, velocity tracking, observer estimation, and NN weight estimation to be semi-globally bounded. Simulation results indicated that the proposed controller was more effective than a traditional linear controller. In [62], Bansal et al. used deep neural networks to generalise the dynamics of the system beyond the trajectories used for training, in contrast to the traditional learning methods. The authors conducted an experiment in which they acquired a quadrotor dynamics model through translational and rotational training trajectories, which were controlled independently. They then employed this model to simultaneously control the yaw and position of a quadrotor, which was a non-trivial task due to nonlinear couplings between the two motions. The results of the experiment showed that even simple NN can have good generalisation capabilities and can learn quadrotor dynamics with high accuracy. In summary, the study demonstrated that the acquired dynamics can be employed effectively to control the system. In [63], Mohajerin and Waslander implemented Recurrent Neural Networks (RNN) to discern the dynamics of two aerial vehicles. The use of RNNs in multistep prediction needed appropriate state initialisation, which involves assigning accurate initial values to the neuron outputs during the initial prediction step. The authors demonstrated that the network initialised using an NN-based method outperformed the identical network initialised via alternative methods.

In [64], Torrente et al. presented an approach to model aerodynamic impacts using Gaussian Processes (GP), a widely used technique in supervised machine learning. This approach was integrated into an MPC system to achieve effective and accurate real-time feedback control in a quadrotor. The integration of these methods resulted in a notable enhancement of positional tracking accuracy, demonstrated through simulation as well as in the real-world application of the quadrotor. Therefore, Bauersfeld et al. also developed a quadrotor dynamics model that can precisely capture intricate aerodynamic effects [65]. To accomplish this, they combined a rotor model based on Blade-Element-Momentum (BEM) theory with learned residual force and torque terms represented by a deep neural network. Experimental results showed that this approach could accurately model quadrotors, even throughout aggressive trajectories. Building upon the previous research, Salzmann et al. introduced a highly efficient framework known as Real-time Neural MPC [66]. This framework enabled the integration of large-capacity neural network architectures as dynamics constraints into the MPC formulation. In contrast to a naive integration of a deep network into an MPC framework, this approach allowed unconstrained model architecture selection, embedded real-time capability for larger models, and GPU acceleration, all without sacrificing performance. Through experiments conducted in simulation and on a real-world agile quadrotor platform, the feasibility of this framework was demonstrated by significantly reducing positional tracking error in comparison to MPC approaches without neural network dynamics.

Conversely, Reinforcement Learning (RL) is a distinct category of machine learning methods that endows an agent with the ability to learn in an interactive setting through trial-and-error learning, using feedback from its actions and experiences. Nonetheless, it is a challenge to utilise this technique in unstable systems that may collapse irreparably during the learning process prior to obtaining an efficient policy. To tackle these challenges, multiple strategies have been introduced in the literature. In [67], Zhang et al. presented a technique that integrated MPC with RL within the framework of guided policy search. The authors utilised MPC for data generation during training, while having access to complete state observations offered by an instrumented training environment. The generated data was then used to train a deep neural network policy, which was only permitted to access the raw observations from the vehicle's onboard sensors. Following training, the neural network policy was able to effectively control the vehicle, despite not knowing the complete state, and at a substantially reduced computational cost compared to MPC. The authors assessed the efficacy of this approach by learning obstacle avoidance policies for a simulated quadrotor, relying on simulated onboard sensors and no explicit state estimation during testing. Furthermore, Berkenkamp et al. introduced a novel learning algorithm that took into account safety considerations, as defined by stability guarantees [68]. The authors extended control-theoretic results on Lyapunov stability verification and explained how statistical models of the dynamics could be used to achieve high-performance control policies with verifiable stability certificates. More precisely, they demonstrated that it was feasible to exploit the system's regularity properties to safely learn about the dynamics and thereby improve the policy while expanding the estimated safe region of attraction without exiting it. Therefore, Kaufmann et al. conducted a benchmark comparison of existing learned control policies for agile quadrotor flight [69]. Their results indicated that training a control policy that regulated body rates and thrust was more resilient in transferring from simulations to real-world scenarios, as opposed to a policy that directly specified individual rotor thrusts. Additionally, the authors demonstrated that such a control policy, which had been trained through RL, could effectively operate a quadrotor in real-world experiments.

Finally, Table 2.4 provides an overview of the data-driven control methods based on machine learning strategies addressed in this section.

Table 2.1: Related work on classic model-based control methods.

References	Method	Brief Description
[7–15]	PID	Basic feedback control method that computes an error value at each time step and subsequently uses the proportional, integral, and derivative terms to make corrections.
[16–19]	LQR	Modern control strategy that aims to determine a linear feedback control law to meet the system's physical constraints and minimise a quadratic cost function.
[20–23]	H_∞	Control technique used to design robust controllers that minimise the effect of disturbances on the system.
[24–28]	Backstepping	Recursive control technique that integrates the selection of a Lyapunov function with the design of a feedback controller, in order to stabilise and track the output of a nonlinear system.
[24, 25, 29, 30]	SMC	Nonlinear control method that involves the application of a discontinuous control signal to induce the system state to slide along a predefined manifold.
[31–34]	Feedback Linearisation	Nonlinear control technique that algebraically transforms the nonlinear dynamics of a system into a partially or fully linearised system to allow the application of linear control techniques.
[35–39]	MPC	Nonlinear control approach that uses a dynamics model of the system to predict its future behaviour and optimise a control input over a finite time horizon.

Table 2.2: Related work on system identification techniques.

References	Method	Brief Description
[40–43]	Least Squares	Identification method that involves minimising the summation of a sequence of squared terms, based on Gauss method.
[44, 45]	Maximum Likelihood	Technique used to identify the parameters of a statistical model by finding the values that maximise the likelihood function.
[46, 47]	ARMAX	Model used for system identification that incorporates autoregressive and moving average models along with exogenous variables to represent the correlation between a variable of interest and other explanatory variables.
[48]	EKF	Recursive algorithm that estimates the state of a nonlinear system, based on first-order linearisation of the process and measurement functions.

Table 2.3: Related work on data-driven control approaches.

References	Method	Brief Description
[51, 52]	VFRT	Control approach that computes a virtual reference signal based on the system's measured output and then optimises the controller by minimising a performance criterion that compares the actual system output with the virtual reference output.
[53, 54]	MFC	Control method which compensates for time-varying disturbances and unmodeled system dynamics that are beyond the capabilities of a basic controller to handle effectively.
[55]	SPC	Control technique characterised by the combination of predictive control law and a subspace predictor, which also employs a multistep forward prediction model to compute the finite future trajectory in a single step.
[5, 56–58]	DeePC	Control method that computes optimal controls for unknown systems using real-time output feedback, through a receding horizon implementation.

Table 2.4: Related work on data-driven control methods based on machine learning strategies.

References	Method	Brief Description
[60–62, 65, 66]	NN	System with the interconnection between artificial neurons in different layers. Control methods that use NN aim to obtain a control law that addresses system uncertainties by training the network through the adjustment of its weights.
[63]	RNN	Type of NN that can process sequential data by maintaining a memory of the past inputs. It can also be employed as a control strategy, whereby an optimal control action is computed using predicted future states that are based on the current system state and control inputs.
[64]	GP	Generic supervised learning method designed to solve regression and probabilistic classification problems. It can also be incorporated into a control approach to accomplish effective and accurate control by learning the unmodelled dynamics of the system.
[67–69]	RL	Type of machine learning that allows an agent to learn in an interactive setting through trial-and-error learning, using feedback from its actions and experiences. This method can be used to solve control problems by learning optimal control policies.

Chapter 3

Theoretical Background

In this chapter, certain theoretical aspects needed through this dissertation are revised. First, an overview of the nonlinear and linear models of the quadrotor dynamics is presented. Afterwards, the model-based control strategies used in this thesis are discussed, particularly the LQR method, which will play a fundamental role in the data collection stage, and the MPC method, which will serve as the standard model-based control approach to be compared with DeePC.

3.1 Quadrotor Model

3.1.1 Nonlinear Model

This section presents the nonlinear model of the UAV, based on Newton-Euler formalism. The nonlinear dynamics are described in both the body-fixed reference frame $\{B\}$ and the inertial reference frame $\{I\}$, as illustrated in Figure 3.1. The unit vectors along the axis of the body-fixed frame $\{B\}$ are labelled as $\{\boldsymbol{x}_B, \boldsymbol{y}_B, \boldsymbol{z}_B\}$, while the unit vectors along the inertial frame $\{I\}$ axis are denoted by $\{\boldsymbol{x}_I, \boldsymbol{y}_I, \boldsymbol{z}_I\}$. It is assumed that the origin of the body-fixed frame $\{B\}$ coincides with the centre of mass of the quadrotor. The x_B and y_B axes are situated in the plane defined by the four rotors, while the z_B axis is perpendicular to this plane and points downward, indicating the direction of total thrust. Additionally, Figure 3.1 also depicts the propeller numbering convention.

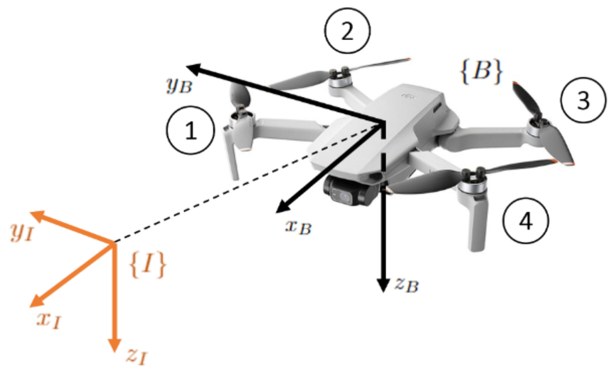


Figure 3.1: Reference frames and rotor numbering.

Let $\boldsymbol{p} = [x, y, z]^T$ denote the position vector of the centre of mass of the UAV in $\{I\}$. Let $\boldsymbol{R}$ represent the rotation matrix from the body-fixed frame to the inertial frame, which can be parameterised by $\boldsymbol{R}(\boldsymbol{\lambda})$ with $\boldsymbol{\lambda} = [\phi, \theta, \psi]^T$, where the Euler angles ϕ, θ, and ψ correspond to the roll, pitch and yaw angles, respectively. Consider $\boldsymbol{\omega} = [p, q, r]^T$ as the angular velocity of frame $\{B\}$ relative to $\{I\}$ expressed in $\{B\}$. Let the control input to the system $\boldsymbol{u}$ comprise the total thrust $T \in \mathbb{R}^+$ and the body torques $\boldsymbol{\tau} = [\tau_x, \tau_y, \tau_z]^T \in \mathbb{R}^{\nVDash}$, both defined in $\{B\}$. Based on [70], the rigid body equations of motion of the quadrotor are given by

$$\begin{aligned} \dot{\boldsymbol{p}} &= \boldsymbol{v}, \\ m_b\dot{\boldsymbol{v}} &= m_b g \boldsymbol{z}_I - \boldsymbol{R}T\boldsymbol{z}_I, \\ \dot{\boldsymbol{R}} &= \boldsymbol{R}S(\boldsymbol{\omega}), \\ \boldsymbol{J}\dot{\boldsymbol{\omega}} &= -\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{\tau}, \end{aligned} \tag{3.1}$$

where $m_b \in \mathbb{R}^+$ represents the mass of the quadrotor, $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ denotes the inertia tensor described in $\{B\}$, the constant g corresponds to the Earth's gravity, and $S(\cdot)$ denotes the skew-symmetric operator such that $S(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$. Considering that the Euler angles follow the sequence of rotation Z-Y-X, it is possible to conclude that the resultant rotation matrix $\boldsymbol{R}(\boldsymbol{\lambda})$ is given by

$$\boldsymbol{R}(\boldsymbol{\lambda}) = \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) - \cos(\phi)\cos(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}.$$

Subsequently, assuming also the effect of drag on the translational motion of the quadrotor, the equation describing the translational dynamics presented in (3.1) is modified as follows:

$$m_b\dot{\boldsymbol{v}} = m_b g \boldsymbol{z}_I - \boldsymbol{R}T\boldsymbol{z}_I - \boldsymbol{R}^T\boldsymbol{C}_D\boldsymbol{R}\boldsymbol{v}, \tag{3.2}$$

where $\boldsymbol{C}_D$ denotes the drag coefficients matrix associated with the translational dynamics, which is represented in $\{B\}$, and $\boldsymbol{R}^T$ represents the rotation matrix from the inertial frame to the body-fixed frame.

Furthermore, the angle rates $\dot{\boldsymbol{\lambda}} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ are obtained from the body rotational rates $\boldsymbol{\omega}$ by applying:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \boldsymbol{Q}(\boldsymbol{\lambda})\boldsymbol{\omega} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}.$$

On the other hand, according to [70], the force F_i and the torque τ_i generated by the rotor i, hovering in free space, can be modelled using the following equations:

$$\begin{aligned} F_i &= c_F\omega_i^2, \\ \tau_i &= (-1)^{i+1}c_\tau\omega_i^2, \end{aligned}$$

where $c_F, c_\tau \in \mathbb{R}^+$. These constants, which can be obtained experimentally, are dependent on several factors such as the area of the disk, the rotor radius, the air density, the geometry and the profile of the rotor, and the effect of drag produced by the rotor flow. Consequently, the relation between the torque τ_i produced by a rotor and its generated force F_i is described by the following expression:

$$\tau_i = \frac{c_\tau}{c_F} F_i = cF_i.$$

Furthermore, it is also defined that rotors 1 and 3 rotate anticlockwise, whereas rotors 2 and 4 rotate clockwise. This configuration is chosen in such a way that the total moment generated by the pair of rotors about the z_B axis is null during the hovering flight. Thus, the control signal $\boldsymbol{u}$ for the system is composed of the total thrust $T \in \mathbb{R}^+$ and the body torques $\boldsymbol{\tau} = [\tau_x, \tau_y, \tau_z]^T$ generated by the four rotors. For the vehicle illustrated in Figure 3.1, the following expressions are verified:

$$\begin{aligned} T &= \sum_{i=1}^{4} F_i, \\ \tau_x &= -(F_1 + F_2)b_x + (F_3 + F_4)b_x, \\ \tau_y &= (F_1 + F_4)b_y - (F_2 + F_3)b_y, \\ \tau_z &= \sum_{i=1}^{4} \tau_i = c(F_1 + F_2 + F_3 + F_4), \end{aligned} \tag{3.3}$$

where b_x and b_y denote the perpendicular distance of the rotor to the x_B and y_B axes, respectively.

Therefore, by rearranging the terms of (3.3), it is possible to conclude that the control signal $\boldsymbol{u}$ can be mathematically expressed in terms of the generated forces F_i as

$$\boldsymbol{u} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -b_x & -b_x & b_x & b_x \\ b_y & -b_y & -b_y & b_y \\ c & -c & c & -c \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}.$$

Finally, by considering the state vector $\boldsymbol{x} = [\boldsymbol{p}^T, \boldsymbol{v}^T, \boldsymbol{\lambda}^T, \boldsymbol{\omega}^T]^T$, the input vector $\boldsymbol{u} = [T, \boldsymbol{\tau}^T]^T$, and the equations described above, it is possible to express the nonlinear dynamics of the quadrotor in the compact form $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, where

$$f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{v} \\ g\boldsymbol{z}_I - \frac{1}{m_b}\boldsymbol{R}T\boldsymbol{z}_I \\ \boldsymbol{Q}(\boldsymbol{\lambda})\boldsymbol{\omega} \\ -\boldsymbol{J}^{-1}S(\boldsymbol{\omega})\boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{J}^{-1}\boldsymbol{\tau} \end{bmatrix}. \tag{3.4}$$

3.1.2 Linear Model

Given the detailed derivation of the nonlinear model of quadrotor dynamics presented above, in order to design linear control laws for the system, it is necessary to linearise the obtained model around the equilibrium point.

To simplify the process, it is assumed that the equilibrium point for which the linearisation is performed corresponds to the quadrotor's hover condition, at a particular position $\boldsymbol{p}_0$ and a specific yaw angle ψ_0. Assuming $\psi_0 = 0$ and taking into account the hovering flight conditions, the equilibrium state vector $\boldsymbol{x}_0$ and the equilibrium input vector $\boldsymbol{u}_0$ can be, respectively, described as

$$\begin{gathered}\boldsymbol{x}_0 = [\boldsymbol{p}_0^T \;\; \boldsymbol{v}_0^T \;\; \boldsymbol{\lambda}_0^T \;\; \boldsymbol{\omega}_0^T]^T = [\boldsymbol{p}_0^T \;\; \mathbf{0}_{3\times1}^T \;\; \mathbf{0}_{3\times1}^T \;\; \mathbf{0}_{3\times1}^T]^T, \\ \boldsymbol{u}_0 = [T_0 \;\; \boldsymbol{\tau}_0^T]^T = [m_b g \;\; \mathbf{0}_{3\times1}^T]^T.\end{gathered} \tag{3.5}$$

Then, continuing the linearisation process, the incremental variables of this model are defined as

$$\begin{aligned}&\delta\boldsymbol{x} = [\delta\boldsymbol{p}^T \;\; \delta\boldsymbol{v}^T \;\; \delta\boldsymbol{\lambda}^T \;\; \delta\boldsymbol{\omega}^T]^T \in \mathbb{R}^{12}, \\ &\delta\boldsymbol{u} = [\delta T \;\; \delta\boldsymbol{\tau}^T]^T \in \mathbb{R}^4,\end{aligned}$$

where

$$\begin{aligned}\delta\boldsymbol{x} &= \boldsymbol{x} - \boldsymbol{x}_0, \\ \delta\boldsymbol{u} &= \boldsymbol{u} - \boldsymbol{u}_0.\end{aligned}$$

Hence, using the Taylor series expansion to linearise (3.4) around the equilibrium point results in the following expression:

$$\dot{\boldsymbol{x}} \approx \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} \boldsymbol{x} + \left.\frac{\partial f}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} \boldsymbol{u}, \tag{3.6}$$

where the higher-order terms of the Taylor series are neglected.

Firstly, from (3.4) and (3.6), it can be inferred that

$$\left.\frac{\partial \dot{\boldsymbol{p}}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = [\mathbf{0}_{3\times3} \;\; \boldsymbol{I}_{3\times3} \;\; \mathbf{0}_{3\times3} \;\; \mathbf{0}_{3\times3}]. \tag{3.7}$$

Regarding $\dot{\boldsymbol{v}}$ and taking into account (3.4), (3.5), and (3.6), the following auxiliary matrix is defined:

$$\begin{aligned}\boldsymbol{G} := \left.\frac{\partial \dot{\boldsymbol{v}}}{\partial \boldsymbol{\lambda}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} &= -\frac{1}{m}T_0 \frac{\partial}{\partial \boldsymbol{\lambda}}(\boldsymbol{R}(\boldsymbol{\lambda})\boldsymbol{z}_I)\Big|_{\boldsymbol{x}_0,\boldsymbol{u}_0} \\ &= -g\frac{\partial}{\partial \boldsymbol{\lambda}}\left.\begin{bmatrix}\cos(\phi)\sin(\theta)\cos(\psi)+\sin(\phi)\sin(\psi) \\ \cos(\phi)\sin(\theta)\sin(\psi)-\sin(\phi)\cos(\psi) \\ \cos(\phi)\cos(\theta)\end{bmatrix}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} \\ &= -g\begin{bmatrix}0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & -g & 0 \\ g & 0 & 0 \\ 0 & 0 & 0\end{bmatrix}.\end{aligned} \tag{3.8}$$

Thus, it is possible to state that

$$\left.\frac{\partial \dot{\boldsymbol{v}}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = [\mathbf{0}_{3\times3} \;\; \mathbf{0}_{3\times3} \;\; \boldsymbol{G} \;\; \mathbf{0}_{3\times3}]. \tag{3.9}$$

Concerning $\dot{\boldsymbol{\lambda}}$, from (3.4), (3.5), and (3.6), it yields

$$\left.\frac{\partial\,\dot{\boldsymbol{\lambda}}}{\partial\,\boldsymbol{x}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \dfrac{\partial\,\boldsymbol{Q}(\boldsymbol{\lambda})}{\partial\,\boldsymbol{\lambda}}\boldsymbol{\omega}_0 & \boldsymbol{Q}(\boldsymbol{\lambda}_0)\dfrac{\partial\,\boldsymbol{\omega}}{\partial\,\boldsymbol{\lambda}}\end{bmatrix} = [\mathbf{0}_{3\times3}\;\;\mathbf{0}_{3\times3}\;\;\mathbf{0}_{3\times3}\;\;\boldsymbol{I}_{3\times3}]. \tag{3.10}$$

In relation to $\dot{\omega}$ and given that

$$\begin{aligned}\left.\frac{\partial\,\dot{\boldsymbol{\omega}}}{\partial\,\boldsymbol{\omega}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} &= -\boldsymbol{J}^{-1}\frac{\partial}{\partial\,\boldsymbol{\omega}}(S(\boldsymbol{\omega})\boldsymbol{J}\boldsymbol{\omega})\Big|_{\boldsymbol{x}_0,\boldsymbol{u}_0}\\ &= -\boldsymbol{J}^{-1}\frac{\partial\,S(\boldsymbol{\omega})}{\partial\,\boldsymbol{\omega}}\boldsymbol{J}\boldsymbol{\omega}_0 - \boldsymbol{J}^{-1}S(\boldsymbol{\omega}_0)\boldsymbol{J}\\ &= \mathbf{0}_{3\times3},\end{aligned}$$

it is possible to conclude that

$$\left.\frac{\partial\,\dot{\boldsymbol{\omega}}}{\partial\,\boldsymbol{x}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = [\mathbf{0}_{3\times3}\;\;\mathbf{0}_{3\times3}\;\;\mathbf{0}_{3\times3}\;\;\mathbf{0}_{3\times3}]. \tag{3.11}$$

In summary, from (3.7), (3.9), (3.10), and (3.11), one can establish the linear system state matrix as

$$\boldsymbol{A}_c := \left.\frac{\partial\,f}{\partial\,\boldsymbol{x}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = \begin{bmatrix}\mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{G} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix}. \tag{3.12}$$

A similar approach is employed for the input vector, resulting in

$$\left.\frac{\partial\,\dot{\boldsymbol{p}}}{\partial\,\boldsymbol{u}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{0}_{3\times3}\end{bmatrix}, \tag{3.13}$$

$$\left.\frac{\partial\,\dot{\boldsymbol{v}}}{\partial\,\boldsymbol{u}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = \begin{bmatrix}-\frac{1}{m_b}\boldsymbol{R}(\boldsymbol{\lambda}_0)\boldsymbol{z}_I & \mathbf{0}_{3\times3}\end{bmatrix} = \begin{bmatrix}-\frac{1}{m_b}\boldsymbol{z}_I & \mathbf{0}_{3\times3}\end{bmatrix}, \tag{3.14}$$

$$\left.\frac{\partial\,\dot{\boldsymbol{\lambda}}}{\partial\,\boldsymbol{u}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{0}_{3\times3}\end{bmatrix}, \tag{3.15}$$

$$\left.\frac{\partial\,\dot{\boldsymbol{\omega}}}{\partial\,\boldsymbol{u}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = \begin{bmatrix}\mathbf{0}_{3\times1} & \boldsymbol{J}^{-1}\end{bmatrix}. \tag{3.16}$$

Therefore, from (3.13), (3.14), (3.15), and (3.16), it is possible to define

$$\boldsymbol{B}_c := \left.\frac{\partial\,f}{\partial\,\boldsymbol{u}}\right|_{\boldsymbol{x}_0,\boldsymbol{u}_0} = \begin{bmatrix}\mathbf{0}_{3\times1} & \mathbf{0}_{3\times3}\\ -\frac{1}{m_b}\boldsymbol{z}_I & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3}\\ \mathbf{0}_{3\times1} & \boldsymbol{J}^{-1}\end{bmatrix}. \tag{3.17}$$

From (3.12) and (3.17), one rewrites (3.6) as

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_c\boldsymbol{x} + \boldsymbol{B}_c\boldsymbol{u}.$$

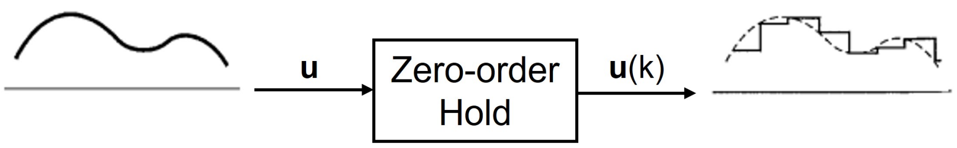


Figure 3.2: Discretisation of a signal using the ZOH method.

In addition to the linearised model deduced above, this work also uses another quadrotor model, which assumes the existence of an inner loop of body rates. Therefore, the actuation signal transmitted to the system comprises input commands of thurst and body rates. Subsequently, the system state vector is constituted only by the position, velocity, and orientation vectors. Hence, in this case, the state and input vectors are, respectively, given by:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^T & \boldsymbol{v}^T & \boldsymbol{\lambda}^T \end{bmatrix},$$
$$\boldsymbol{u} = \begin{bmatrix} T & \boldsymbol{\omega}^T \end{bmatrix}.$$

By employing the identical linearisation process discussed above and considering the relevant formulations in (3.4), the ensuing matrices can be obtained:

$$\boldsymbol{A}_c = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{G} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \end{bmatrix}, \quad \boldsymbol{B}_c = \begin{bmatrix} \boldsymbol{0}_{3\times1} & \boldsymbol{0}_{3\times3} \\ -\frac{1}{m_b}\boldsymbol{z}_I & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times1} & \boldsymbol{I}_{3\times3} \end{bmatrix}. \tag{3.18}$$

Finally, the discretisation of this linear system is necessary to allow the implementation of the subsequent proposed methods. According to [71], the discretisation method referred to as Zero-Order Hold (ZOH) assumes that the control input $\boldsymbol{u}$ remains constant over a sampling time T_s. Figure 3.2 shows the piecewise-constant signal resulting from the application of this method. Thus, using the ZOH method, the linear system can be described in the following equivalent discrete state space representation:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k),$$
$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k),$$

where $\boldsymbol{A} = e^{\boldsymbol{A}_c T_s}$, $\boldsymbol{B} = \left(\int_0^{T_s} e^{\boldsymbol{A}_c \tau}\, d\tau\right)\boldsymbol{B}_c$, $\boldsymbol{C} = \boldsymbol{C}_c$, and $\boldsymbol{D} = \boldsymbol{D}_c$.

3.2 Model-based Methods

3.2.1 Linear Quadratic Regulator

In this subsection, the LQR is presented, based on [72]. The LQR is a modern control strategy that aims to determine a feedback control law so that the system to be controlled can meet physical constraints while also minimising a quadratic cost function.

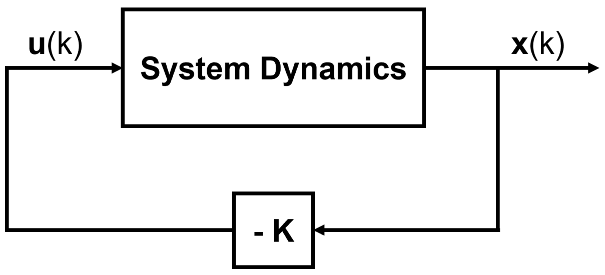


Figure 3.3: Structure of the Linear Quadratic Regulator.

Consider a discrete system described by the linear state-space representation

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \tag{3.19}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of state-space variables and $\boldsymbol{u} \in \mathbb{R}^m$ is the system's input. The optimal regulator problem computes the gain matrix $\boldsymbol{K}$ of the optimal control vector

$$\boldsymbol{u}(k) = -\boldsymbol{K}\boldsymbol{x}(k) \tag{3.20}$$

in order to minimise the cost function

$$J = \sum_{k=0}^{\infty} \boldsymbol{x}(k)^T \boldsymbol{Q}\boldsymbol{x}(k) + \boldsymbol{u}(k)^T \boldsymbol{R}\boldsymbol{u}(k), \tag{3.21}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$, denominated as state weighting matrix, is a positive semi-definitive matrix and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$, designated as control weighting matrix, is a positive definitive matrix. Hence, the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ determine, respectively, the relative importance of the deviation of the state x from the desired state and the expenditure of the energy of the control signals. It is also important to note that since this is an infinite time control problem, the control solution turns into a steady-state solution, leading to a constant optimal gain matrix $\boldsymbol{K}$. Figure 3.3 depicts a block diagram representation of an LQR controller.

Through algebraic manipulation, it is possible to conclude that the gain matrix $\boldsymbol{K}$ that minimises the cost function J is given by

$$\boldsymbol{K} = (\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{B} + \boldsymbol{R})^{-1}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{A}. \tag{3.22}$$

The matrix $\boldsymbol{P}$ in (3.22) can be calculated by the discrete-time *Riccati* equation:

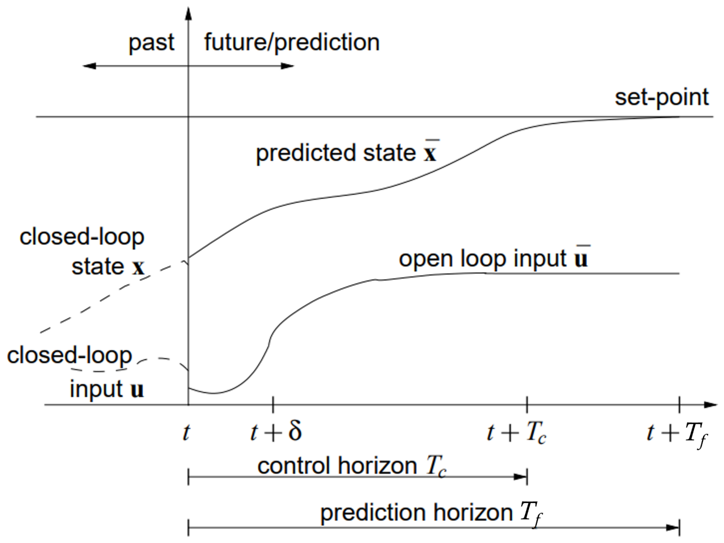
$$\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A} - \boldsymbol{P} - \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{B} + \boldsymbol{R})^{-1}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} = 0.$$

It should also be noted that this thesis employs the LQR method to design a linear control law based on the model resulting from the discretisation of (3.18).

3.2.2 Model Predictive Control

Following the approach of classical methods, one of the most popular control strategies, known as MPC, is introduced. The subsequent descriptions of this method are based on [5, 35, 73].

The fundamental principle of MPC is to predict the future behaviour of the controlled system over a specified time horizon and compute an optimal control input that minimises a chosen cost function while ensuring the satisfaction of the system constraints. Thus, Figure 3.4 depicts this fundamental rule of MPC.



past
future/prediction
set-point
predicted state $\bar{\mathbf{x}}$
closed-loop state $\mathbf{x}$
open loop input $\bar{\mathbf{u}}$
closed-loop input $\mathbf{u}$
t
$t+\delta$
$t+T_c$
$t+T_f$
control horizon T_c
prediction horizon T_f

Figure 3.4: Basic principle of MPC strategy (adapted from [73]).

As in the previous section, let $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^p$, and $\boldsymbol{u} \in \mathbb{R}^m$ denote the vector of state-space variables, the system's output, and the system's input, respectively. Furthermore, the classical MPC algorithm uses a receding horizon approach to solve the following reference tracking optimal control problem:

$$
\begin{aligned}
\underset{\boldsymbol{u},\boldsymbol{x},\boldsymbol{y}}{\text{minimise}} \quad & \sum_{k=0}^{T_f-1} [\boldsymbol{y}(k) - \overline{\boldsymbol{y}}(k)]^T \boldsymbol{Q} [\boldsymbol{y}(k) - \overline{\boldsymbol{y}}(k)] + \\
& \qquad + [\boldsymbol{u}(k) - \overline{\boldsymbol{u}}(k)]^T \boldsymbol{R} [\boldsymbol{u}(k) - \overline{\boldsymbol{u}}(k)] \\
\text{subject to} \quad & \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \forall k \in \{0,\dots,T_f-1\}, \\
& \boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k), \forall k \in \{0,\dots,T_f-1\}, \\
& \boldsymbol{x}(0) = \hat{\boldsymbol{x}}(t), \\
& \boldsymbol{u}(k) \in \mathcal{U}, \forall k \in \{0,\dots,T_f-1\}, \\
& \boldsymbol{y}(k) \in \mathcal{Y}, \forall k \in \{0,\dots,T_f-1\},
\end{aligned}
\tag{3.23}
$$

where t is the current time, k is the temporal instance of the predictive horizon window, $T_f \in \mathbb{N}$ is the time horizon, $\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{y}$ are the decision variables, $\mathcal{U} \subseteq \mathbb{R}^m$ is an input constraint set, $\mathcal{Y} \subseteq \mathbb{R}^p$ is an output constraint set, $\overline{\boldsymbol{u}}(k) \in \mathbb{R}^m$ and $\overline{\boldsymbol{y}}(k) \in \mathbb{R}^p$ are, respectively, the desired input and output reference calculated for each iteration of the algorithm, $\boldsymbol{Q} \in \mathbb{R}^{p\times p}$ is the output cost matrix, $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ is the control cost matrix, and $\hat{\boldsymbol{x}}(t)$ is the estimated state at time t.

Finally, Algorithm 1 summarises the implementation of the MPC control strategy.

Algorithm 1 MPC [5]

```
Input: matrices $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$, input and output references $(\overline{\boldsymbol{u}}, \overline{\boldsymbol{y}})$, current state $\hat{\boldsymbol{x}}_t$, constraint sets $\mathcal{U}$ and $\mathcal{Y}$, and performance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$.
    1. Solve (3.23) for $\boldsymbol{u}^\star = (\boldsymbol{u}^\star(0)^T, \dots, \boldsymbol{u}^{\star T}(T_f - 1))^T$.
    2. Apply inputs $\boldsymbol{u}(t)^T = \boldsymbol{u}^\star(0)^T$.
    3. Update $\hat{\boldsymbol{x}}(t)$.
    4. Return to 1.
```

Chapter 4

Data-enabled Predictive Control

In this chapter, the DeePC is presented based on [5, 56]. First, the necessary preliminaries are provided. Then, the optimisation problem related to the DeePC algorithm is defined, although it will only be effective when applied to deterministic LTI systems. Hence, robustness regularisations are introduced, which will enable the algorithm's adaptation for the nonlinear quadrotor system with noisy measurements. Finally, a detailed description of the implementation of the data collection stage is provided, highlighting its influence on the performance of this data-driven control technique.

4.1 DeePC Algorithm

Consider an unknown deterministic discrete-time LTI system represented by

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k), \end{aligned} \tag{4.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^p$, $\boldsymbol{u} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$, and $\boldsymbol{D} \in \mathbb{R}^{p\times m}$. Assuming that (4.1) is given in its minimal realisation, it is possible to ensure controllability and observability properties of the represented system. The lag of the system (4.1) is defined by the smallest integer $\ell \in \mathbb{Z}_{\geq 0}$ for which the observability matrix

$$\mathcal{O}_\ell(\boldsymbol{A}, \boldsymbol{C}) := [\boldsymbol{C}, \boldsymbol{C}\boldsymbol{A}, ..., \boldsymbol{C}\boldsymbol{A}^{\ell-1}]^T$$

has rank n.

From the unknown system (4.1) and during an offline procedure, T_d sequences of input/output data $(\boldsymbol{u}_d, \boldsymbol{y}_d) = \{\boldsymbol{u}_d(k), \boldsymbol{y}_d(k)\}_{k=0}^{T_d}$ are collected. After that, the collected data is reorganised into two Hankel matrices, as shown below:

$$
\begin{aligned}
\boldsymbol{\mathcal{H}}_L(\boldsymbol{u}_d) &= \begin{bmatrix} \boldsymbol{u}_d(1) & \boldsymbol{u}_d(2) & \dots & \boldsymbol{u}_d(T_d - L + 1) \\ \boldsymbol{u}_d(2) & \boldsymbol{u}_d(3) & \dots & \boldsymbol{u}_d(T_d - L + 2) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{u}_d(L) & \boldsymbol{u}_d(L+1) & \dots & \boldsymbol{u}_d(T_d) \end{bmatrix}, \\
\boldsymbol{\mathcal{H}}_L(\boldsymbol{y}_d) &= \begin{bmatrix} \boldsymbol{y}_d(1) & \boldsymbol{y}_d(2) & \dots & \boldsymbol{y}_d(T_d - L + 1) \\ \boldsymbol{y}_d(2) & \boldsymbol{y}_d(3) & \dots & \boldsymbol{y}_d(T_d - L + 2) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{y}_d(L) & \boldsymbol{y}_d(L+1) & \dots & \boldsymbol{y}_d(T_d) \end{bmatrix},
\end{aligned} \tag{4.2}
$$

where L is the sum of the length of the initialisation sequence, $T_{ini} \in \mathbb{Z}_{\geq 0}$, and the prediction horizon, $T_f \in \mathbb{Z}_{\geq 0}$.

Definition 4.1 (Persistency of excitation [49])**.** Let $L, T \in \mathbb{Z}_{\geq 0}$ such that $T \geq L$. The sequence of signals $\boldsymbol{u} = \{\boldsymbol{u}(k)\}_{k=0}^{T} \in \mathbb{R}^{mT}$ is persistently exciting of order L if the Hankel matrix $\boldsymbol{\mathcal{H}}_L(\boldsymbol{u})$ has full row rank.

The term persistently exciting refers to an input signal that is sufficiently rich and long to excite the system and produce an output sequence that is representative of its behaviour. Note that the DeePC algorithm is based on the following fundamental result.

Theorem 4.1. [49] Consider (4.1) and let $T_d, L \in \mathbb{Z}_{\geq 0}$. Let $(\boldsymbol{u}_d, \boldsymbol{y}_d) = \{\boldsymbol{u}_d(k), \boldsymbol{y}_d(k)\}_{k=0}^{T_d}$ be a trajectory of (4.1) of length T_d, assuming that the sequence of signals $\boldsymbol{u}_d$ is persistently exciting of order L. Hence, $(\boldsymbol{u}, \boldsymbol{y}) = \{\boldsymbol{u}(k), \boldsymbol{y}(k)\}_{k=0}^{L}$ is a trajectory of (4.1) if and only if there exists $\boldsymbol{g} \in \mathbb{R}^{T_d - L + 1}$ such that

$$
\begin{bmatrix} \boldsymbol{\mathcal{H}}_L(\boldsymbol{u}_d) \\ \boldsymbol{\mathcal{H}}_L(\boldsymbol{y}_d) \end{bmatrix} \boldsymbol{g} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}.
$$

Note that, in this particular case, considering that $\boldsymbol{u}_d$ is persistently exciting of order L, it implies that the condition $T_d \geq (m+1)L - 1$ must be satisfied.

According to the previous statement, the subspace spanned by the columns of the Hankel matrix, $[\boldsymbol{\mathcal{H}}_L(\boldsymbol{u}_d), \boldsymbol{\mathcal{H}}_L(\boldsymbol{y}_d)]^T$, exactly matches the subspace of potential trajectory of (4.1). Therefore, the Hankel matrix may be used as a nonparametric model for (4.1), which may be generated directly from raw time-series data without the need for any learning.

Returning to (4.2), these Hankel matrices, composed by input/output collected data, are divided into two parts,

$$
\begin{bmatrix} \boldsymbol{U}_p \\ \boldsymbol{U}_f \end{bmatrix} := \boldsymbol{\mathcal{H}}_{T_{ini}+T_f}(\boldsymbol{u}_d), \quad \begin{bmatrix} \boldsymbol{Y}_p \\ \boldsymbol{Y}_f \end{bmatrix} := \boldsymbol{\mathcal{H}}_{T_{ini}+T_f}(\boldsymbol{y}_d), \tag{4.3}
$$

where $\boldsymbol{U}_p$ consists of the first T_{ini} block rows of $\boldsymbol{\mathcal{H}}_{T_{ini}+T_f}(\boldsymbol{u}_d)$ and $\boldsymbol{U}_f$ consists of the last T_f block rows of $\boldsymbol{\mathcal{H}}_{T_{ini}+T_f}(\boldsymbol{u}_d)$ (similarly for $\boldsymbol{Y}_p$ and $\boldsymbol{Y}_f$). Hence, the data in $\boldsymbol{U}_p$ and $\boldsymbol{Y}_p$ are used to estimate the initial conditions, whereas the data in $\boldsymbol{U}_f$ and $\boldsymbol{Y}_f$ are used to predict future trajectories. Consider further that the input signal $\boldsymbol{u}_d$, collected during an offline process, is persistently exciting of order $T_{ini} + T_f + n$.

From Theorem 4.1, $(\boldsymbol{u}, \boldsymbol{y}) = \{\boldsymbol{u}(k), \boldsymbol{y}(k)\}_{k=0}^{T_f-1}$ is a possible future trajectory of (4.1) if and only if there exists $\boldsymbol{g} \in \mathbb{R}^{T_d - T_{ini} - T_f + 1}$ such that

$$\begin{bmatrix} \boldsymbol{U}_p \\ \boldsymbol{Y}_p \\ \boldsymbol{U}_f \\ \boldsymbol{Y}_f \end{bmatrix} \boldsymbol{g} = \begin{bmatrix} \boldsymbol{u}_{ini} \\ \boldsymbol{y}_{ini} \\ \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix}. \tag{4.4}$$

In order to uniquely fix the initial condition from which the future trajectory departs, it is necessary to ensure that $T_{ini} \geq \ell$. Therefore, this condition also implies that the predicted trajectory calculated by $\boldsymbol{y} = \boldsymbol{Y}_f \boldsymbol{g}$ is unique.

Given a reference input $\boldsymbol{u}_r \in \mathbb{R}^m$, a reference trajectory $\boldsymbol{y}_r \in \mathbb{R}^p$, past input/output data $[\boldsymbol{u}_{ini}, \boldsymbol{y}_{ini}]^T$, an input constraint set $\mathcal{U} \subseteq \mathbb{R}^m$, an output constraint set $\mathcal{Y} \subseteq \mathbb{R}^p$, an output cost matrix $\boldsymbol{Q} \in \mathbb{R}^{p \times p}$, and a control cost matrix $\boldsymbol{R} \in \mathbb{R}^{m \times m}$, it is possible to formulate the following data-driven optimisation problem:

$$\begin{aligned} \underset{\boldsymbol{g}, \boldsymbol{u}, \boldsymbol{y}}{\text{minimise}} \quad & \sum_{k=0}^{T_f-1} [\boldsymbol{y}(k) - \boldsymbol{y}_r]^T \boldsymbol{Q} [\boldsymbol{y}(k) - \boldsymbol{y}_r] + [\boldsymbol{u}(k) - \boldsymbol{u}_r]^T \boldsymbol{R} [\boldsymbol{u}(k) - \boldsymbol{u}_r] \\ \text{subject to} \quad & (4.4) \\ & \boldsymbol{u}(k) \in \mathcal{U}, \forall k \in \{0, \dots, T_f - 1\}, \\ & \boldsymbol{y}(k) \in \mathcal{Y}, \forall k \in \{0, \dots, T_f - 1\}. \end{aligned} \tag{4.5}$$

Hence, when the unknown system has the form (4.1), it has been demonstrated that the optimisation problem (4.5) is equivalent to the MPC problem presented in (3.23).

4.2 Regularised DeePC Algorithm

One of the motivations of this work is to implement the DeePC algorithm in a real-world quadrotor described in Section 3.1, i.e., in a nonlinear system corrupted by process noise. Thus, it is necessary to include some regularisations in the optimal control problem (4.5), resulting in the following regularised optimisation problem:

$$\begin{aligned} \underset{\boldsymbol{g}, \boldsymbol{u}, \boldsymbol{y}, \boldsymbol{\sigma}_y}{\text{minimise}} \quad & \sum_{k=0}^{T_f-1} c(\boldsymbol{u}(k), \boldsymbol{y}(k)) + \lambda_g \|\boldsymbol{g}\|_2 + \lambda_y \|\boldsymbol{\sigma}_y\|_2 \\ \text{subject to} \quad & \begin{bmatrix} \boldsymbol{U}_p \\ \boldsymbol{Y}_p \\ \boldsymbol{U}_f \\ \boldsymbol{Y}_f \end{bmatrix} \boldsymbol{g} = \begin{bmatrix} \boldsymbol{u}_{ini} \\ \boldsymbol{y}_{ini} \\ \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\sigma}_y \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \\ & \boldsymbol{u}(k) \in \mathcal{U}, \forall k \in \{0, \dots, T_f - 1\}, \\ & \boldsymbol{y}(k) \in \mathcal{Y}, \forall k \in \{0, \dots, T_f - 1\}, \end{aligned} \tag{4.6}$$

where $c(\boldsymbol{u}(k), \boldsymbol{y}(k)) = [\boldsymbol{y}(k) - \boldsymbol{y}_r]^T \boldsymbol{Q} [\boldsymbol{y}(k) - \boldsymbol{y}_r] + [\boldsymbol{u}(k) - \boldsymbol{u}_r]^T \boldsymbol{R} [\boldsymbol{u}(k) - \boldsymbol{u}_r]$, $\boldsymbol{\sigma}_y \in \mathbb{R}^{pT_{ini}}$ is an auxiliary slack variable, and $\lambda_g, \lambda_y \in \mathbb{R}_{>0}$ are regularisation parameters.

The inclusion of the slack variable $\boldsymbol{\sigma}_y$ is primarily intended to always ensure the feasibility of the constraint equation, since this constraint may become inconsistent when the output measurements are corrupted by noise. The slack variable was also penalised with a two-norm penalty function. Moreover, it is intended to choose λ_y sufficiently large, in such a way that the condition $\boldsymbol{\sigma}_y \neq 0$ is only verified in cases where the constraint is infeasible. Furthermore, the inclusion of the two-norm regularisation on $\boldsymbol{g}$ is a common technique in a distributionally robust problem formulation.

Finally, the DeePC method, in which (4.6) is implemented in a receding horizon approach, is summarised in Algorithm 2.

Algorithm 2 Regularised DeePC [56]

```
Input: $T_d, T_f, T_{ini}$, $\boldsymbol{\mathcal{H}} = [\boldsymbol{U}_p^T, \boldsymbol{Y}_p^T, \boldsymbol{U}_f^T, \boldsymbol{Y}_f^T]^T$, input and output references $(\boldsymbol{u}_r, \boldsymbol{y}_r)$, constraint sets $\mathcal{U}$ and $\mathcal{Y}$, performance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, regularisation parameters ($\lambda_g$, $\lambda_y$), and past input/output data $(\boldsymbol{u}_{ini}, \boldsymbol{y}_{ini})$.
    1. Solve (4.6) for $\boldsymbol{g}^\star$.
    2. Compute the optimal input sequence $\boldsymbol{u}^\star = \boldsymbol{U}_f \boldsymbol{g}^\star$.
    3. Apply inputs $(\boldsymbol{u}(t)^T, \ldots, \boldsymbol{u}(t+s)^T)^T = (\boldsymbol{u}^\star(0)^T, \ldots, \boldsymbol{u}^\star(s)^T)^T$ for some $s \leq T_f - 1$.
    4. Set $t$ to $t+s$ and update $u_{ini}$ and $y_{ini}$ to the $T_{ini}$ most recent past input/output measurements.
    5. Return to 1.
```

4.3 Data Collection

As mentioned in Section 4.1, the input signal used to fill the Hankel matrices represented in (4.3) must be persistently exciting of sufficient order. Therefore, it is imperative to carefully collect the data for the successful implementation of this data-driven control algorithm.

This data can be collected either by applying a random input sequence or by conducting a manual flight experiment, where a human operator performs as an outer controller. The former approach was adopted in order to ensure the repeatability of results. Consequently, a Pseudorandom Binary Sequence (PRBS) was used, as it typically provides satisfactory performance for conventional system identification techniques. It should also be noted that the input signals employed for data collection comprise the PRBS excitation signal added to an existing simple controller that maintains the quadrotor around the hover state. This controller is primarily used to stabilize the quadrotor, ensuring an improved algorithm performance and minimum safe conditions throughout the data collection process.

According to [74], a PRBS is inherently periodic with a maximum period of $2^n - 1$, where $n \in \mathbb{N}$ is the order of the PRBS. Given that it is intended to collect T_d sequences of input/output data, it can be inferred that the order of the PRBS to be generated is $n_{Td} = \texttt{floor}(\log_2(T_d + 1))$ and consequently, the desired period of the PRBS is determined by $\text{PRBS}_{\text{period}} = 2^{n_{Td}} - 1$. Moreover, the number of periods of this PRBS can be obtained by $\texttt{ceil}(T_d/\text{PRBS}_{\text{period}})$. The MATLAB functions $\texttt{floor}(X)$ and $\texttt{ceil}(X)$ round the elements of X to the nearest integers towards $-\infty$ and $+\infty$, respectively. In this dissertation, the PRBS excitation signal was generated using the MATLAB function designated as `idinput()`. Hence, this design process can be summarized in the following steps:

1. Initially, a PRBS is generated, which will later serve as the basis for creating the excitation signals used for each system input. Therefore, it is necessary to determine the amplitude, the period length, the number of periods, and the desired clock period, i.e., the minimum number of sampling intervals for which the value of the PRBS does not change. Note that performing future calculations on entire periods is essential since generating only a partial period of a PRBS will not produce a signal with the desired properties.

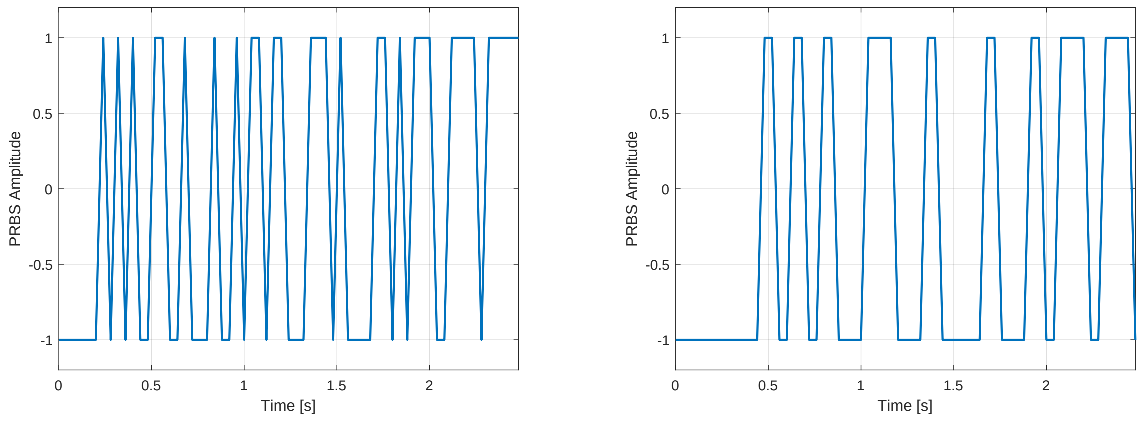
2. Then, select only the first T_d elements of the generated PRBS.

3. Furthermore, determine the desired amplitude for the excitation signal of the first input of the system. To obtain this excitation signal, it is necessary to multiply the chosen amplitude by the PRBS generated previously.

4. To generate the excitation signal linked with the subsequent system input, it is once again required to multiply the PRBS by a predetermined amplitude. Nonetheless, it is crucial to execute a circular shift of the positions of the original PRBS elements before performing the multiplication in this situation. The amount of positions by which the elements are shifted is determined by $\texttt{floor}(\text{PRBS}_{\text{period}}(i+1)/m)$, where $i \in \mathbb{Z}_{\geq 0}$ denotes the number of circular shifts that have already been conducted on the original PRBS.

5. Finally, repeat the previous step to produce the excitation signals of the remaining system inputs.

Finally, Figure 4.1 illustrates part of two generic PRBS, generated for different clock period values, assuming $T_s =$ 0.04s. When the clock period is set to 2 samples, signal value variations at successive sample times are no longer observed. Thus, it is possible to corroborate that the clock period influences the minimum number of sampling intervals during which the PRBS value does not change.



(a) Clock Period = 1 sample

(b) Clock Period = 2 samples

Figure 4.1: Part of a generic PRBS for different clock periods, with sample time of $T_s =$0.04s.

Chapter 5

Implementation and Basic Simulation Results

This chapter presents a detailed description of the implementation of the proposed algorithms, along with a concise analysis of the simulation results obtained by subjecting the system to a unit step. Firstly, the vehicle model used, the DeePC controller's implementation, and schematics of its control architecture are provided. Then, the hyperparameters and constraints associated with the DeePC algorithm are described. Finally, the step responses for the different types of control architectures using the DeePC controller are discussed and compared.

5.1 Simulation Architecture

In order to evaluate the performance of the proposed methods, the simulated results are obtained from a very realistic model implemented in MATLAB/Simulink [75, 76]. Based on the one provided by the authors of [56], this simulation model includes the accurate modelling of rotor dynamics, actuation constraints, drag, measurement noise, and inner-loop controller dynamics. It is also important to note that the necessary adjustments were made to align the orientation axes employed in the simulation with those detailed in Section 3.1. The subsequent outcomes were obtained from a computer with the operating system Windows 11, equipped with an Intel Core i7-1165G7 CPU 2.80GHz, and 16GB of RAM. The optimisation problems of the DeePC algorithm were solved using the OSQP solver [77].

5.1.1 3DR® Iris+ Quadrotor

The 3DR® Iris+ quadrotor, represented in Figure 5.1, was employed to run the simulations. This commercial vehicle, developed by 3D Robotics (3DR), was chosen mainly because it is a vehicle widely studied in the literature, whose parameters are well identified experimentally. Moreover, another advantage is the existence of several realistic simulators of this quadrotor, which can be used in the future to compare results. Table 5.1 details the physical properties used to simulate the quadrotor model.

Figure 5.1: 3DR® Iris+ quadrotor [78].

Table 5.1: Experimental physical properties of 3DR® Iris+ quadrotor (obtained from [78]).

Mass (kg)	$\boldsymbol{J_{xx}}$ (kg m^2)	$\boldsymbol{J_{yy}}$ (kg m^2)	$\boldsymbol{J_{zz}}$ (kg m^2)	$\boldsymbol{\bar{b}_x}$ (m)	$\boldsymbol{\bar{b}_y}$ (m)	$\boldsymbol{c_F/c_\tau}$
1.37	0.0219	0.0109	0.0306	0.111	0.240	0.065

Hence, based on the experimental results presented in [78], it was possible to obtain:

- the vehicle mass, including the battery pack used;
- the inertia tensor, which was considered as a diagonal matrix;
- the average perpendicular distances of the rotor to the x_B and y_B axes, designated respectively by $\bar{b}_x$ and $\bar{b}_y$;
- the torque-to-thrust coefficient of the rotor, denoted by c_F/c_τ.

5.1.2 DeePC Controller Implementation

As previously stated, before the implementation of the DeePC controller, it is essential to perform the crucial task of data collection. Figure 5.2 illustrates the block diagram that succinctly schematises the implementation of the data collection step in Simulink. In accordance with Section 4.3, the input signals used for data collection also result from the sum of the PRBS excitation signal and the signal generated by the LQR controller, which was appropriately tuned to maintain the quadrotor in the hover state.

Moreover, Figure 5.3 depicts the block diagram of the DeePC controller implementation in Simulink. Initially, while the simulation time is lower than a predetermined DeePC trial start time, the quadrotor is stabilised in a hover state through the LQR controller. Once the opposite is verified, the DeePC controller is enabled, assuming exclusive control over the quadrotor. This transition between controllers is secured through the utilisation of the logic and comparison blocks, presented in Figure 5.3. Additionally, the merge block ensures that only the input signal generated by the active controller is effectively applied to the quadrotor model.

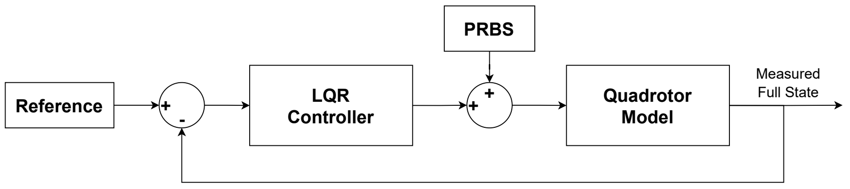


Figure 5.2: Simplified structure of the data collection step implemented in Simulink.

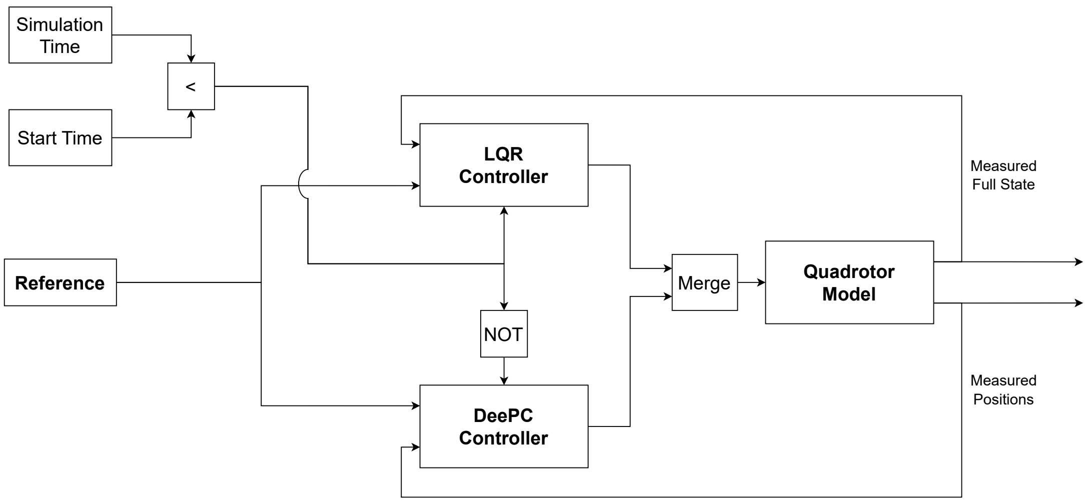


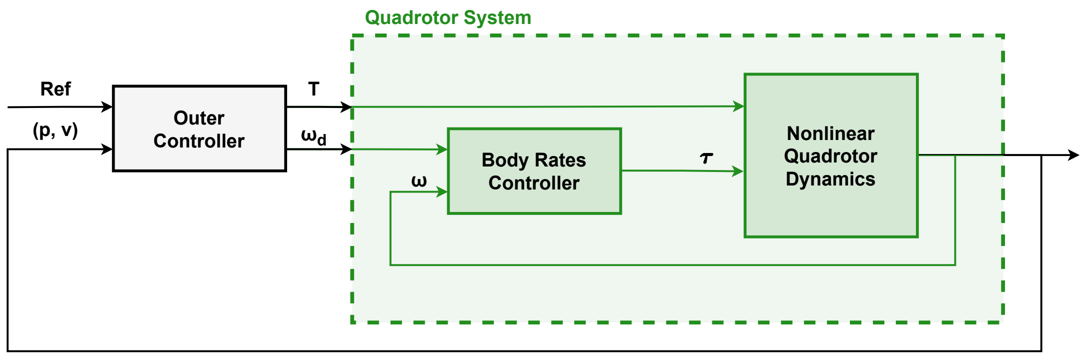
Figure 5.3: Generic structure of the DeePC controller tested in Simulink.

It should also be noted that in contrast to LQR, the DeePC controller does not require access to the measured full state. Instead, the latter only needs position measurements to function properly and is thus referred to as a real-time output feedback controller.

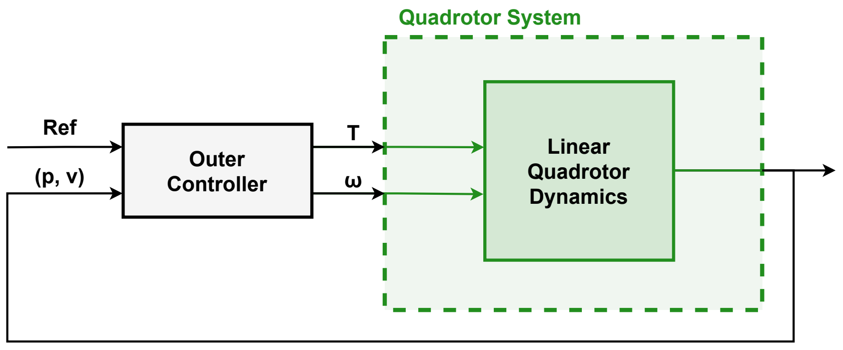
5.1.3 Control Architecture

A more detailed representation of the implemented cascade control architecture is presented. In the simulation model provided by the authors of [56], the outer controller, whether it is the LQR or DeePC controller, applies thrust and body rate commands to the quadrotor system. Subsequently, in the nonlinear case, an inner-loop PID controller is incorporated. This inner-loop controller generates a torque command through the difference between the desired and estimated body rates. The combination of thrust and torque commands is finally applied to the quadrotor dynamics model described in (3.4). On the other hand, for the linear case, the resulting set of thrust and body rate commands is employed in the linear quadrotor model defined by the matrices presented in (3.18). In this dissertation, the architecture described above is designated as **control architecture A**. Figure 5.4 presents a schematic representation summarising this control architecture, for both linear and nonlinear scenarios.

Nevertheless, the use of body-rate commands for actuation has its drawbacks. In the future, the testing of the proposed algorithms in realistic Software-in-the-Loop (SITL) simulations and experimental trails will use quadrotors equipped with the PX4 autopilot [79], which already provides a very fine-tuned attitude controller. Hence, the implementation of control architecture A will lead to modifications in the



(a) Schematic representation for the nonlinear quadrotor dynamics, as defined in (3.4)



(b) Schematic representation for the linear quadrotor dynamics, as defined in (3.18)

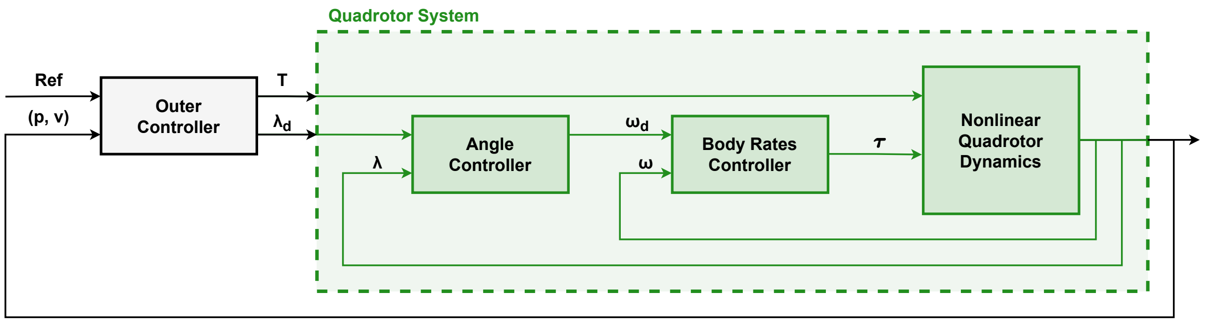
Figure 5.4: Block diagram of the control architecture A.

PX4 code, making the process more difficult. Besides that, it should also be noted that the operating rate of the DeePC method is not ideal for actuating on body-rate commands. Consequently, for these reasons and in order to make a new contribution to the existing work developed by the authors of [56], a novel control architecture, referred to as **control architecture B**, was designed and implemented in this thesis.

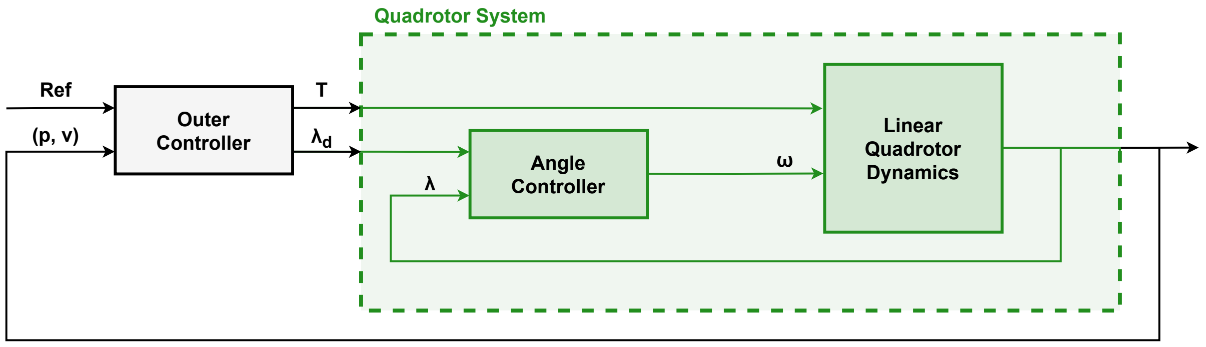
After a block reconfiguration, the quadrotor system now also includes an angle controller, which generates a body rate command by multiplying a gain with the difference between the desired and estimated Euler angles. Thus, the outer controller has transitioned into a position controller, producing thrust and orientation commands for the quadrotor system. The gains for both the outer and the angle controllers are determined using the LQR method, as in control architecture A.

Furthermore, in the nonlinear case, the inner-loop controller responsible for generating torque commands remains unchanged. Hence, the combination of thrust and torque commands is once again applied to the quadrotor dynamics model defined in (3.4). For the linear case, the body rate commands generated by the angle controller, alongside the thrust commands, is applied to the quadrotor linear model characterised by the matrices specified in (3.18). Figure 5.5 depicts the block diagram of the cascade control architecture B, for both linear and nonlinear cases.

In this control architecture, several modifications have been made to the outer controller. Firstly, the position and velocity errors are inputted into a simple position controller that produces a virtual signal, denoted as $\boldsymbol{u}_v \in \mathbb{R}^3$. Subsequently, this signal is applied to a new block which, after rotating



(a) Schematic representation for the nonlinear quadrotor dynamics, as defined in (3.4)



(b) Schematic representation for the linear quadrotor dynamics, as defined in (3.18)

Figure 5.5: Block diagram of the cascade control architecture for the controller model B.

the quadrotor about the z axis, generates the thrust and orientation commands based on the following expressions:

$$\begin{aligned}
\boldsymbol{f}_d &= m_b \boldsymbol{R}_z^T(\psi_d)(g\boldsymbol{z}_I - \boldsymbol{u}_v),\\
\phi_d &= \text{arctg}\left(\frac{-\boldsymbol{f}_{d_2}}{\sqrt{\boldsymbol{f}_{d_1}^2 + \boldsymbol{f}_{d_3}^2}}\right),\\
\theta_d &= \text{arctg}\left(\frac{\boldsymbol{f}_{d_1}}{\boldsymbol{f}_{d_3}}\right),\\
\psi_d &= \psi_{\text{ref}},\\
T &= -\boldsymbol{u}_{v_3} + m_b g,
\end{aligned} \tag{5.1}$$

where T denotes the thrust commands, the vector $(\phi_d, \theta_d, \psi_d)$ represents the orientation commands, $\boldsymbol{f}_d \in \mathbb{R}^3$ is the desired force used to compute the orientation commands, ψ_{ref} is the yaw angle reference, and $\boldsymbol{R}_z^T(\psi_d)$ denotes the rotation matrix about the z axis from $\{I\}$ to $\{B\}$ with a dependence on ψ_d.

Furthermore, for simplicity, it is intended to maintain the yaw angle at zero through the inner-loop controller. Hence, considering that the DeePC controller only requires position measurements, it can be concluded that the number of outputs is $p = 3$. In addition, taking into account the control architectures presented above and the yaw angle statement, it can be inferred that the number of inputs used is $m = 3$.

5.1.4 DeePC Algorithm Hyperparameters and Constraints

The regularised DeePC optimisation problem, defined in (4.6), incorporates several hyperparameters that influence the performance of this algorithm. Therefore, in order to successfully implement the DeePC method, it is crucial to tune the following hyperparameters:

- T_d, the total number of data points used to build the Hankel matrices described in (4.2);
- T_{ini}, the time horizon used for initial condition estimation;
- T_f, the prediction time horizon;
- λ_y, the weight on the regularisation of the initial condition constraint;
- λ_g, the weight on the regularisation of $\boldsymbol{g}$;
- $\boldsymbol{Q}$, the tracking error cost matrix;
- $\boldsymbol{R}$, the control effort cost matrix.

In addition to these hyperparameters, the excitation amplitudes employed in the PRBS generation are fundamental parameters in the data collection phase, thus having a significant influence on the performance of the DeePC algorithm. Therefore, the importance of a correct selection of these parameters is also highlighted.

Regarding constraint sets, for control architecture A, the control input constraint set $\mathcal{U}$ is given by

$$\begin{aligned} T &\in [7.906, 30.094] \text{ N}, \\ \omega_{d,x}, \omega_{d,y} &\in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \text{ rad/s}. \end{aligned}$$

On the other hand, for the control architecture B case, the same constraint set $\mathcal{U}$ is defined by

$$\begin{aligned} T &\in [7.906, 30.094] \text{ N}, \\ \phi_d, \theta_d &\in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right] \text{ rad}. \end{aligned}$$

Finally, for both architectures, the output constraint set $\mathcal{Y}$ is denoted by

$$x, y, z \in [-5, 5] \text{ m}.$$

5.2 Step Responses

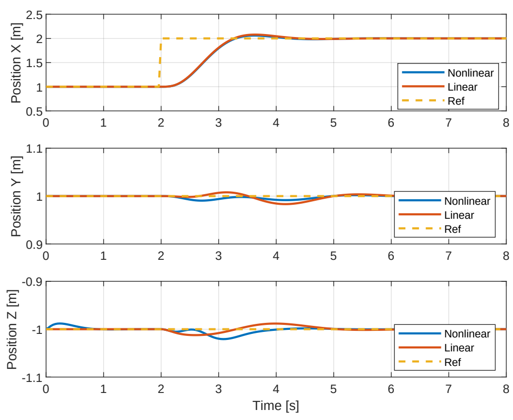
5.2.1 Control Architecture A

This section aims to analyse and discuss the system response to a unit step input in different position variables, when using the DeePC controller implemented in the control architecture A. Table 5.2 presents the tuned hyperparameters used in the implementation of this control architecture. In addition to the values presented in the table, it was also assumed that $\boldsymbol{Q} = \text{diag}(100, 100, 1000, 0, 0, 0, 0, 0, 40)$ and $\boldsymbol{R} = \text{diag}(1, 10, 10, 2)$. These performance matrices are used for both LQR and DeePC controllers. Hence, all the subsequent outcomes presented in this section were obtained using these specified hyperparameters.

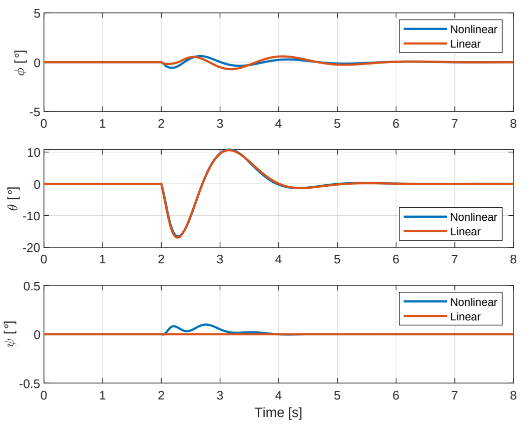
Figures 5.6, 5.7, and 5.8 illustrate the position and orientation responses to a unit step input in x, in y, and in z, respectively. The nonlinear and linear results were obtained using the two variants of the control architecture A as depicted in Figure 5.4.

Table 5.2: Tuned hyperparameters for implementation of DeePC algorithm, in the case of control architecture A.

$\boldsymbol{T_{ini}}$	$\boldsymbol{T_d}$	$\boldsymbol{T_f}$	$\boldsymbol{\lambda_g}$	$\boldsymbol{\lambda_y}$
5	600	50	500	7.5×10^8

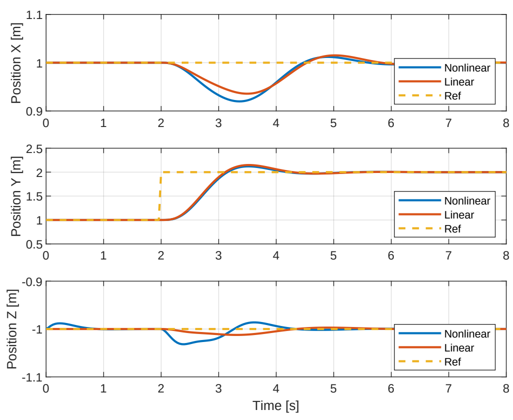


(a) Position response

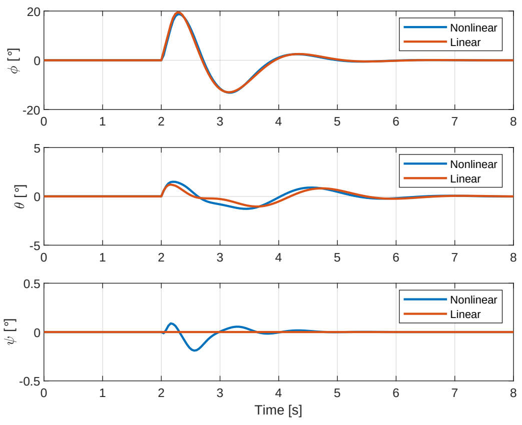


(b) Orientation response

Figure 5.6: Nonlinear and linear responses to a unit step in x for control architecture A.

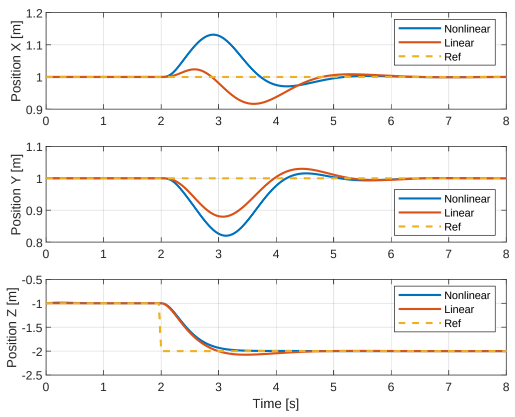


(a) Position response

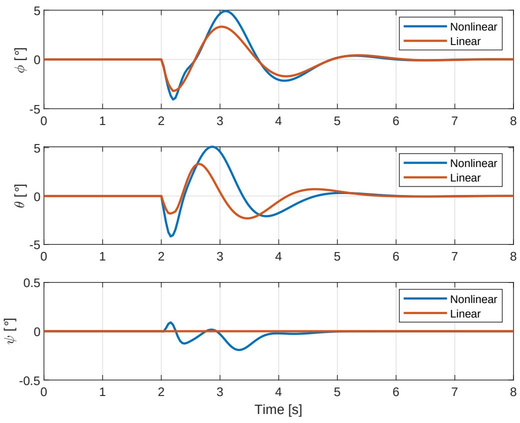


(b) Orientation response

Figure 5.7: Nonlinear and linear responses to a unit step in y for control architecture A.



(a) Position response



(b) Orientation response

Figure 5.8: Nonlinear and linear responses to a unit step in z for control architecture A.

From the analysis of the figures, it is evident that all three scenarios exhibit a satisfactory response, characterised by an acceptable settling time. Concerning overshoot, although not significant, it is noteworthy that the one observed in the position y stands out when the step is taken in that same variable. Moreover, the coupling between the variables represented is noted, essentially in the position response of Figures 5.7 and 5.8.

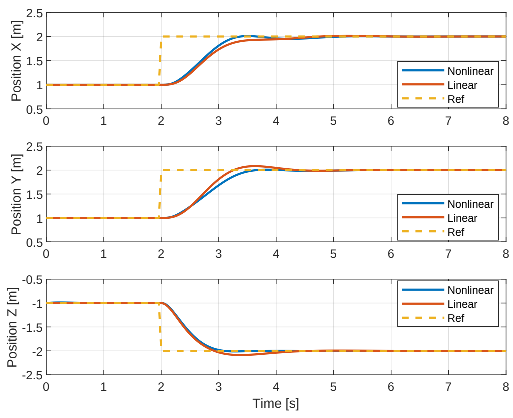
Regarding the comparison of nonlinear and linear responses, these are similar overall, especially in the variables directly influenced by the applied step. For the other variables, the nonlinear response tends to present larger amplitude oscillations. This can be attributed to the fact that the nonlinear quadrotor dynamics model incorporates significant interdependencies among the state variables, in contrast to the linear model.

Proceeding with this analysis, Figure 5.9 presents the nonlinear and linear system responses to a unit step applied to all three position variables. As previously observed, the quadrotor system demonstrates an adequate position and orientation responses, characterised by an acceptable settling time and a minimal overshoot. It is also denoted that the nonlinear response presents slightly better behaviour than the linear one. This can be explained since the used hyperparameters were adjusted to the nonlinear quadrotor model.

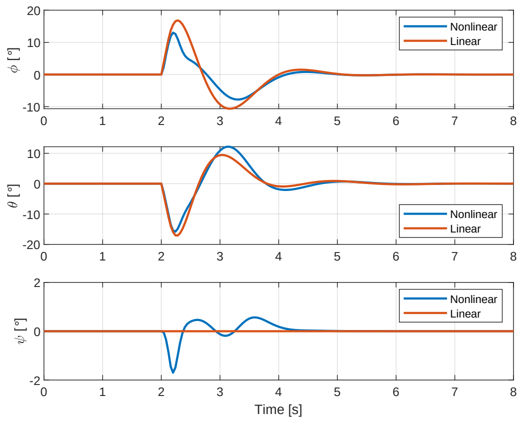
5.2.2 Control Architecture B

In this section, it is intended to repeat the same study for the two variants of the control architecture B represented in Figure 5.5, analysing once again the simulated nonlinear and linear responses to a unit step using the DeePC controller. During the implementation of this control architecture, it was necessary to perform a new adjustment of the employed hyperparameters, resulting in the values presented in Table 5.3 and in the following LQR and DeePC performance matrices:

$$\begin{aligned} \boldsymbol{Q} &= \text{diag}(40, 40, 500, 0, 0, 0, 0, 0, 40), \\ \boldsymbol{R} &= \text{diag}(0.5, 20, 20, 2). \end{aligned} \tag{5.2}$$

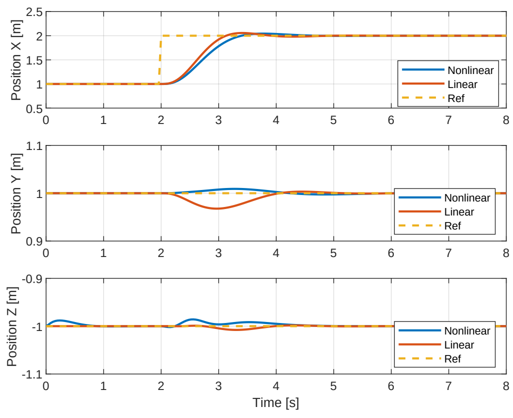


(a) Position response

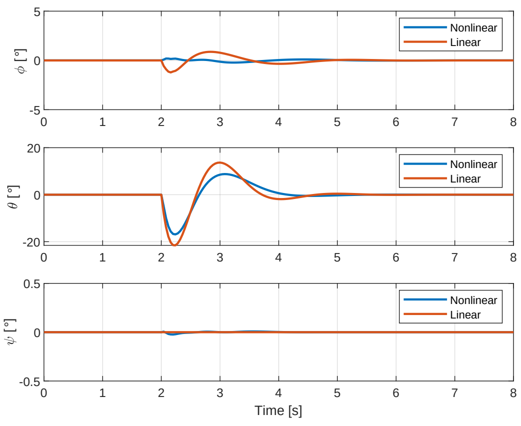


(b) Orientation response

Figure 5.9: Nonlinear and linear responses to a unit step in x, y, and z for control architecture A.



(a) Position response



(b) Orientation response

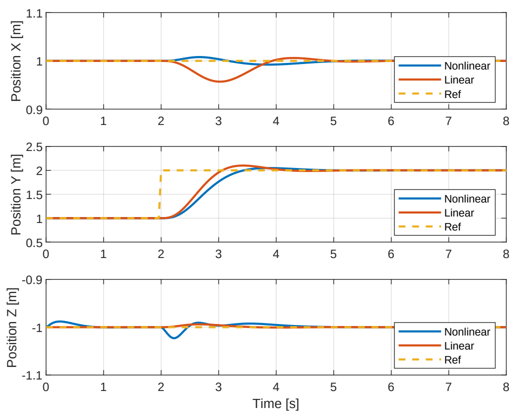
Figure 5.10: Nonlinear and linear responses to a unit step in x for control architecture B.

Table 5.3: Tuned hyperparameters for implementation of DeePC algorithm, in the case of control architecture B.

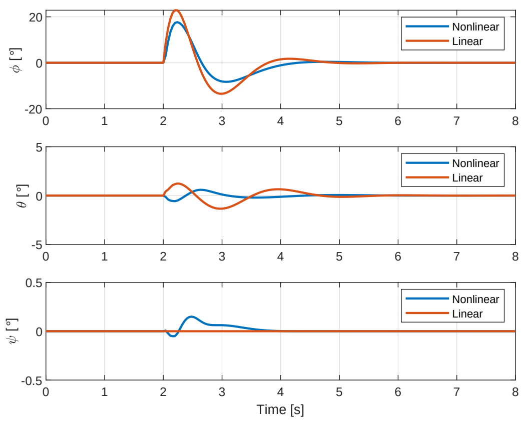
$\boldsymbol{T_{ini}}$	$\boldsymbol{T_d}$	$\boldsymbol{T_f}$	$\boldsymbol{\lambda_g}$	$\boldsymbol{\lambda_y}$
5	900	50	500	$7.5{\times}10^8$

The main changes were applied to the most relevant hyperparameters in the data collection phase. This observation was expected since the transition from control architecture A to B alters the inputs provided to the quadrotor system, thereby influencing the Hankel matrices construction that occurs during the data collection step. The following reported results were obtained using these hyperparameters.

The system responses to a unit step input in x, in y, and in z are depicted in Figures 5.10, 5.11, and 5.12, respectively. From the presented plots, it is possible to verify an acceptable position and orientation responses, especially those generated from the nonlinear quadrotor dynamics model. Upon closer examination, it is concluded that the linear response exhibits not only a shorter settling time but also a larger overshoot in comparison to the nonlinear one. Thus, given that the linear model does not

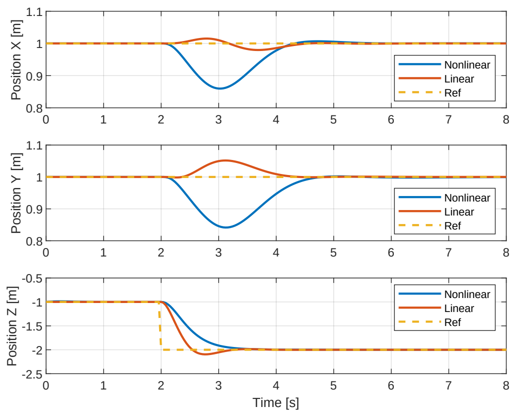


(a) Position response

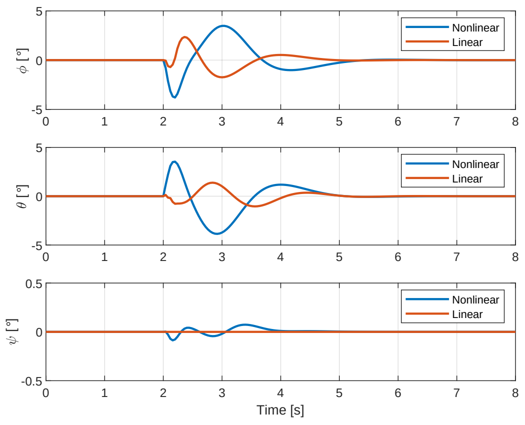


(b) Orientation response

Figure 5.11: Nonlinear and linear responses to a unit step in y for control architecture B.



(a) Position response



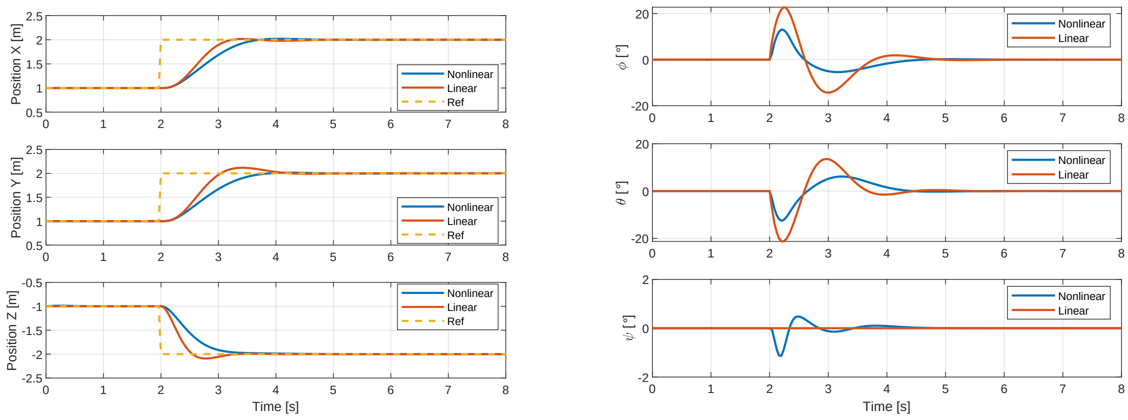
(b) Orientation response

Figure 5.12: Nonlinear and linear responses to a unit step in z for control architecture B.

account for the physical limitations of the quadrotor, it can be inferred that the used controller gains should be slightly higher in this case, resulting in an overly aggressive response. This observation is further corroborated by the fact that the utilised hyperparameters were adjusted for the nonlinear model.

From the presented plots, it is possible to verify an acceptable position and orientation responses, especially those generated from the nonlinear quadrotor dynamics model. Upon closer examination, it is concluded that the linear response exhibits not only a shorter settling time but also a larger overshoot in comparison to the nonlinear one. Thus, given that the linear model does not account for the physical limitations of the quadrotor, it can be inferred that the used controller gains should be slightly higher in this case, resulting in an overly aggressive response. This observation is further corroborated by the fact that the utilised hyperparameters were adjusted for the nonlinear model.

As in the previous section, there are also couplings between the variables, which are accentuated in the response to a step in z represented in Figure 5.12. Nevertheless, these plots indicate that the nonlinear and linear responses exhibit different oscillation frequencies. Additionally, concerning the variables that are not directly linked to the applied step, the corresponding nonlinear and linear responses



(a) Position response

(b) Orientation response

Figure 5.13: Nonlinear and linear responses to a unit step in x, y, and z for control architecture B.

show opposite behaviours.

Finally, Figure 5.13 shows the system responses to a step applied in x, y, and z, denoting once again satisfactory results. The discrepancies between the nonlinear and linear responses are due to the reasons already stated above.

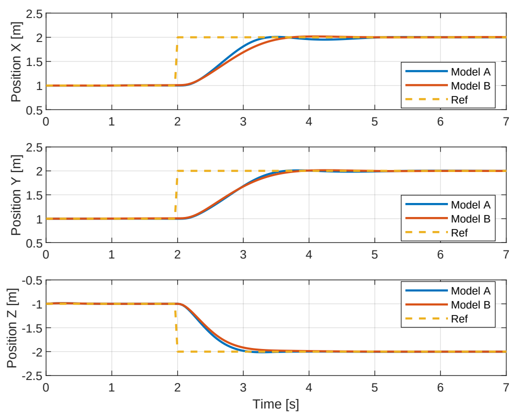
5.2.3 Comparison of Control Architectures

This section aims to compare the results achieved for the two control architectures A and B, essentially focusing on the position and orientation responses for the nonlinear quadrotor model defined in (3.4). Figure 5.14 presents the system responses to a step applied in x, y, and z, implementing the nonlinear variants of the control architectures A and B.

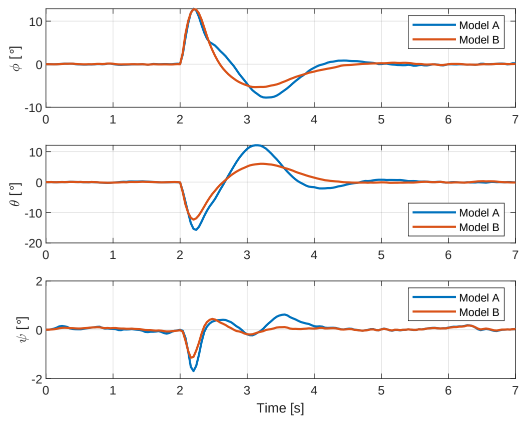
It is observed that the responses are similar, which would be expected since the control architecture B arises from a reconfiguration of the control architecture A. Therefore, the differences, mainly observed in the orientation response, can be explained by the inclusion of the nonlinearities expressed in (5.1). This new element contributes to a less aggressive system response, corroborating the fact that the outcomes of control architecture B present longer settling times and lower oscillation amplitudes in relation to the Euler angles.

Adding a new comparative point to this discussion, Table 5.4 details the average computation time of the DeePC algorithm, t_c, for control architectures A and B. It is possible to infer that the t_c of the control architecture A is 70% lower than the t_c of the control architecture B. This discrepancy was already expected since the value of the hyperparameter T_d is higher in case B. As will be further confirmed in the next chapter, an increase in the value of T_d leads to an increase in t_c. Nevertheless, it should be noted that, despite this difference, both values remain within an acceptable range, considering that the sampling time of the outer controller is 40ms.

In sum, it can be concluded that control architecture A exhibits a superior performance. However, considering that control architecture B emerges as a novel contribution of this work, the subsequent chapter will analyse in detail the performance of the DeePC controller implemented in the nonlinear control architecture B, as illustrated in Figure 5.5(a).



(a) Position response



(b) Orientation response

Figure 5.14: Nonlinear response to a unit step input for the control architectures A and B.

Table 5.4: Average algorithm computation time for the control architectures A and B.

Control Architecture	t_c (ms)
A	8.6
B	12.4

Chapter 6

DeePC Performance Results

In the previous chapter, the details of the implementation of the proposed algorithms and the system's simulated response to an applied step were presented. Following this discussion, this chapter presents the simulation results of the DeePC algorithm performance obtained for the nonlinear variant of control architecture B. Initially, important details about the data collection phase are reviewed. Then, the influence of each hyperparameter on the DeePC performance is discussed. Next, the influence of drag and noise on the measurements is also analysed. Finally, a complete comparison of the performance of MPC, LQR, and DeePC control methods is conducted.

6.1 Data Collection Stage

This section aims to present the data acquired during the data collection stage, which were subsequently used by the DeePC controller, yielding the outcomes depicted in Section 5.2. Furthermore, it is also intended to analyse the influence of various parameters characteristic of the data collection phase on the performance of the DeePC algorithm.

According to Section 4.3, to acquire the collected data that fills the Hankel matrices, it is necessary to establish the following parameters:

- the inverse of the required clock period, $B = 1\text{samples}^{-1}$;
- the desired amplitude for the thrust excitation signal, $T_{\text{exc}_{\text{amp}}} = g \times 10^{-1}\text{N}$;
- the desired amplitude for the roll and pitch excitation signals, $\phi_{\text{exc}_{\text{amp}}} = \theta_{\text{exc}_{\text{amp}}} = 0.1\text{rad}$.

Using the hyperparameters defined in (5.2) and in Table 5.3, in conjunction with the aforementioned parameters, it is possible to obtain the data collected at the input and output of the system after the excitation process. These data are respectively illustrated in Figures 6.1 and 6.2.

Firstly, from Figure 6.1, it is possible to conclude that the x and y positions show a larger variation than z, during the data collection stage. Thus, it can be inferred that a large variation in the z position is not crucial for the acquired data to verify the persistency of excitation condition.

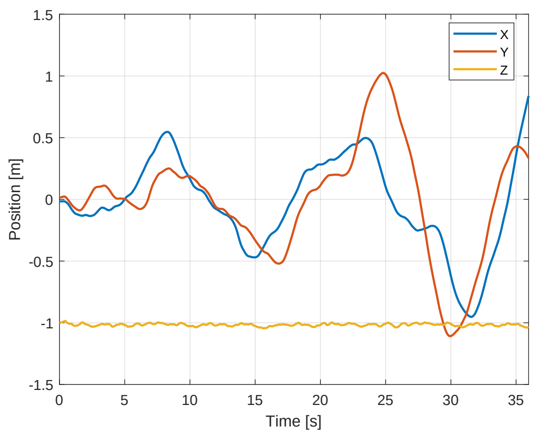
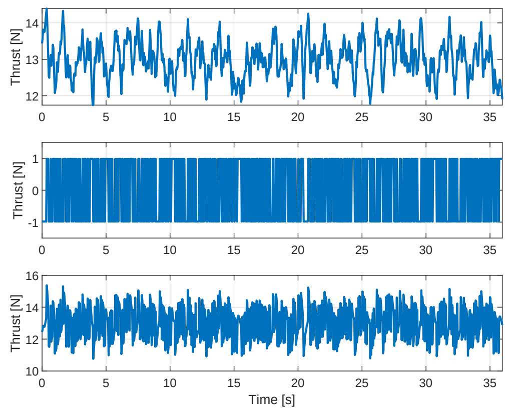
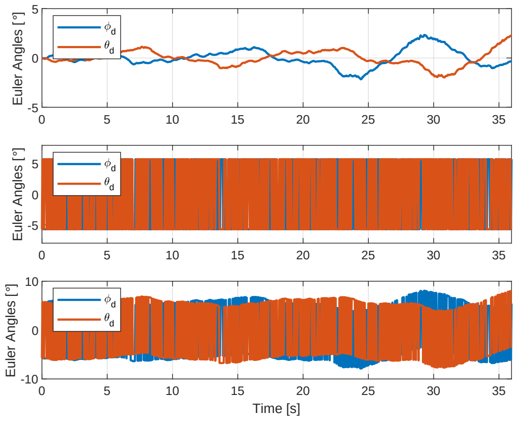


Figure 6.1: Position data collected after the system excitation process.



(a) Thrust data



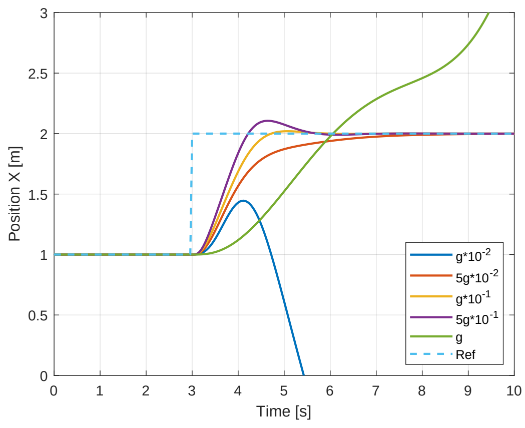
(b) ϕ_d and θ_d data

Figure 6.2: Input data collected after the system excitation process: (i) data generated by the LQR controller; (ii) generated PRBS; (iii) collected thrust and orientation data.

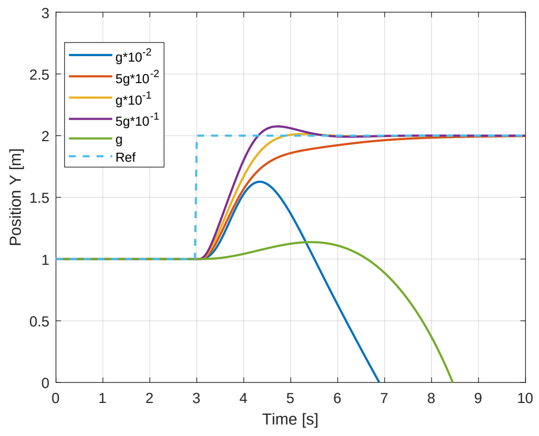
Figure 6.2 shows the data generated by the LQR controller, the derived PRBS, and the respective sum of these latter signals, resulting in the data collected at the system input after the excitation phase. For the desired Euler angles ϕ_d and θ_d, it can be observed that the signal resulting from the sum of the PRBS and the signal produced by the LQR controller is excessively oscillatory, being physically impossible for the system to respond directly to all these transmitted inputs. However, after several attempts to decrease the frequency and amplitude of the PRBS, it was concluded that the best performance of the DeePC algorithm occurred for the situation displayed.

Proceeding to the analysis of the influence of the parameters used in the generation of the PRBS, the illustration of the position response for different values of the thrust excitation signal amplitude is available in Figure 6.3. From the examination of this figure, it can be noted that the DeePC controller fails to stabilise the system when the amplitude is either excessively high or excessively low.

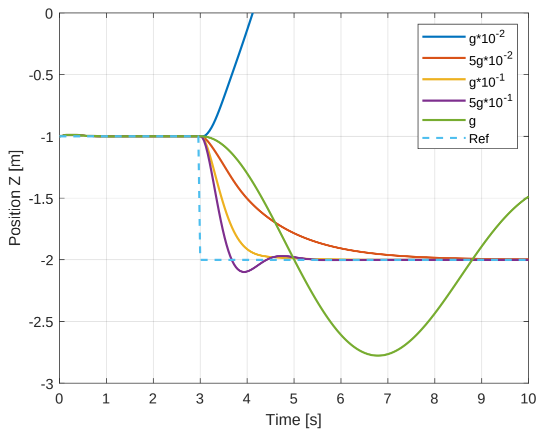
It is also observable that small variations in $T_{\text{exc}_{\text{amp}}}$ result in a significant disparity in the performance of the DeePC controller. For the scenario where $T_{\text{exc}_{\text{amp}}} = 5g \times 10^{-2}$N, a small decrease in amplitude leads to a response characterised by a large settling time. Conversely, in the case of $T_{\text{exc}_{\text{amp}}} = 5g \times 10^{-1}$N, a



(a) Position x response



(b) Position y response



(c) Position z response

Figure 6.3: Influence of the amplitude of the thrust excitation signal on the position response to a unit step.

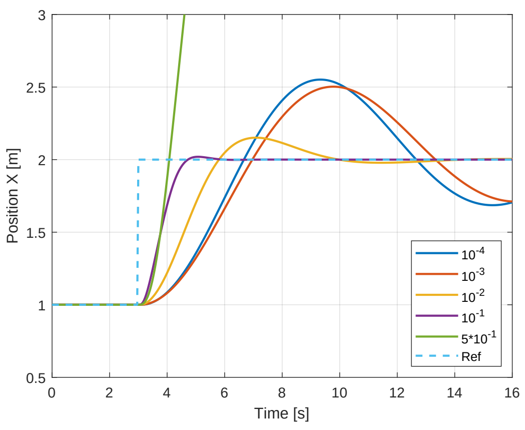
small increase in amplitude originates a response with high overshoot.

Therefore, Figure 6.4 depicts the influence of the amplitude of the roll and pitch excitation signals on the performance of the DeePC controller. As can be seen in the plots, an increase in the order of magnitude of $\phi_{\text{exc}_{\text{amp}}}$ and $\theta_{\text{exc}_{\text{amp}}}$ leads to an unstable response of the nonlinear system. However, in contrast to the previous case, a significant decrease in the amplitude of these excitation signals originates a very oscillatory but stable response.

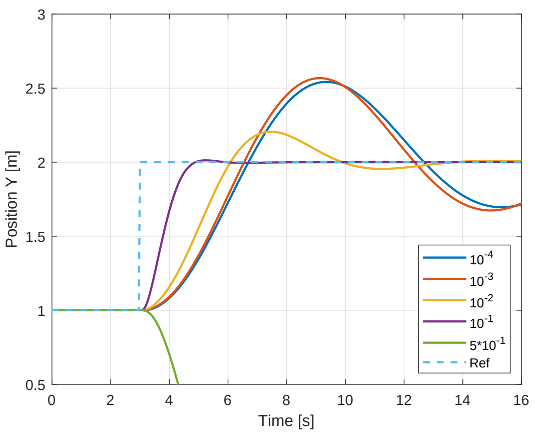
Additionally, it is also ratified that $\phi_{\text{exc}_{\text{amp}}}$ and $\theta_{\text{exc}_{\text{amp}}}$ have a greater influence on the response of positions x and y than on the position z response.

Subsequently, Figure 6.5 displays the influence of the band of the excitation signal on the performance of the DeePC controller. The parameter B, shown in the legend of the plots, represents the inverse of the desired clock period, as stated above. Thus, the generated signal has to remain constant for at least $\texttt{floor}(1/B)$ samples.

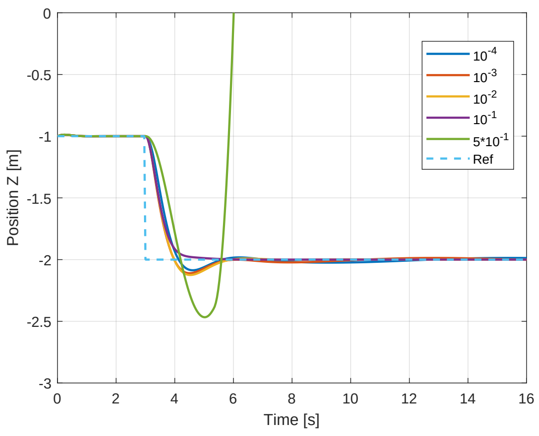
Overall, it can be concluded that for smaller values of B, the nonlinear position response shows aggressive and oscillatory behaviour. It is also noteworthy that the position z is, once again, the least influenced variable by changes in parameter B.



(a) Position x response

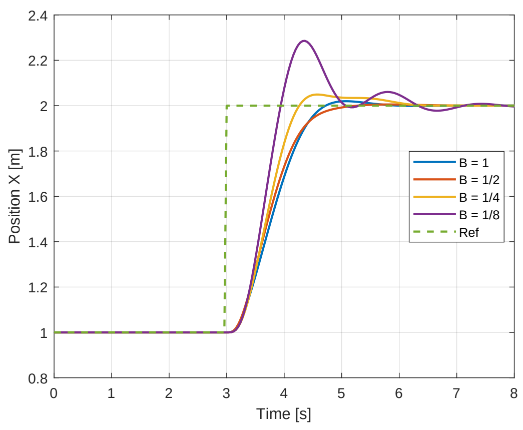


(b) Position y response

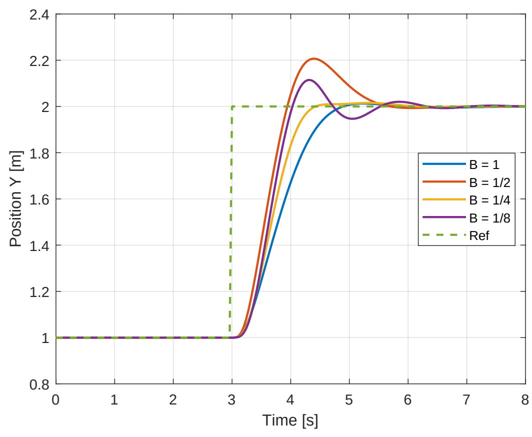


(c) Position z response

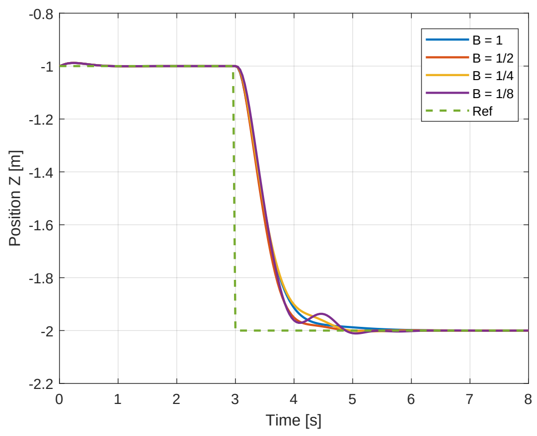
Figure 6.4: Influence of the amplitude of the roll and pitch excitation signals on the position response to a unit step.



(a) Position x response



(b) Position y response



(c) Position z response

Figure 6.5: Influence of the excitation signal band on the position response to a unit step.

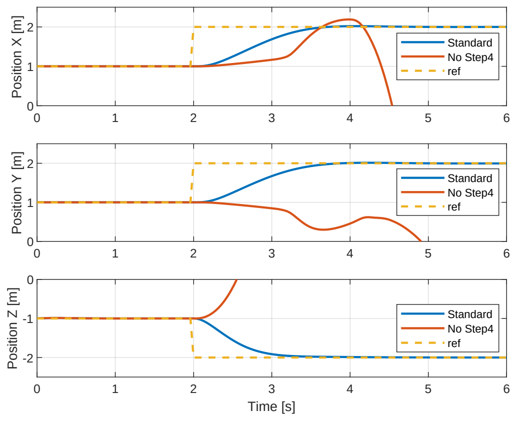


Figure 6.6: Influence of step 4, described in Section 4.3, on the position response to a unit step.

Finally, Figure 6.6 represents the influence of step 4, described in Section 4.3, on the position response to a unit step input in x, y, and z. From the visualisation of the plots, it becomes evident that the DeePC controller fails to stabilise the system response when the generation of the PRBS neglects this step. Upon further analysis, this step involves executing a circular shift of the standard PRBS positions, which yields the resulting excitation signals. Therefore, by not performing this circular shift, the resulting excitation signals will not be independent of each other and consequently, the constructed Hankel matrices do not contain sufficiently relevant information. Thus, it is concluded that the acquisition of data obtained from independent signals is crucial for the proper functioning of the DeePC controller.

6.2 Influence of Hyperparameters on Algorithm Performance

In this section, the influence of each hyperparameter on the performance of the DeePC controller is addressed. Therefore, to study this impact in detail, series of experiments were conducted, changing only the value of a single hyperparameter at each time. The remaining hyperparameters were fixed at the values specified in Table 5.3 and (5.2). Subsequently, the performance of the DeePC controller in each scenario is evaluated by analysing the average overshoot S, maximum settling time t_s, average algorithm computation time t_c, and average static error e.

Table 6.1 details the influence of the hyperparameter T_{ini} on the position response to a unit step applied in x, y, and z. In a first analysis, it is possible to infer that for all displayed values of T_{ini}, the system demonstrates a stable response without static error. Moreover, an increase of T_{ini} tends to directly increase both S and t_c parameters. However, it should be noted that the selected value was not the minimum possible option ($T_{ini} = 3$), taking into account that the settling time does not evolve as the other parameters. Corroborating this observation, it is important to emphasise that this particular hyperparameter determines the time horizon used for initial condition estimation and consequently, choosing an excessively small value would fail to expose certain nonlinearities inherent in the system's response to the controller. Hence, $T_{ini} = 5$ exhibited the best performance, characterised by a shorter settling time when compared to the remaining tested values.

Table 6.1: Influence of T_{ini} on the position response to a unit step input.

T_{ini}	S (%)	t_s (s)	t_c (ms)	e (m)
3	**0.71**	2.48	**13.05**	0
4	1.69	2.56	13.83	0
5	1.07	**1.80**	15.87	0
6	1.87	2.52	16.56	0
7	1.06	2.28	18.75	0
8	3.64	2.80	18.73	0
9	3.55	2.48	18.93	0
10	6.18	2.92	19.48	0

Table 6.2: Influence of T_d on the position response to a unit step input.

T_d	S (%)	t_s (s)	t_c (ms)	e (m)
300	-	-	**5.93**	0.40
600	8.86	3.84	9.56	0
800	2.39	2.36	12.88	0
900	**1.07**	**1.80**	15.87	0
1000	7.09	2.60	18.26	0
1200	6.02	2.36	22.19	0
1500	3.53	2.40	28.27	0
2000	6.14	2.24	39.26	0

The influence of the hyperparameter T_d on the performance of the DeePC controller is represented in Table 6.2. It is observed that, for an excessively low value of T_d, the system stabilises in a position with an average static error of 0.40m. In addition, an increase in T_d leads to an unequivocal increase in t_c. This finding was already expected since T_d is directly associated with the dimension of the Hankel matrices used by the algorithm in the optimisation problem, defined in (4.6). Thus, a higher value of T_d implies a larger size of the Hankel matrices, which consequently leads to a higher computation time of the DeePC algorithm. In this case, $T_d = 900$ was chosen since it is linked to the response with lower S and t_s.

Then, Table 6.3 presents the influence of the hyperparameter T_f on the system position response. It can be inferred that for low values of T_f, the results obtained are characterised by a high value of S and t_s. However, there is a threshold beyond which augmenting T_f does not improve the performance of the DeePC controller. Instead, only the significant increase of t_c is verified. Hence, the best performance was obtained for $T_f = 50$.

Table 6.3: Influence of T_f on the position response to a unit step input.

$\boldsymbol{T_f}$	$\boldsymbol{S}$ (%)	$\boldsymbol{t_s}$ (s)	$\boldsymbol{t_c}$ (ms)	$\boldsymbol{e}$ (m)
10	38.99	26.12	**3.41**	0
25	4.30	2.60	6.99	0
40	2.92	2.48	12.30	0
50	**1.07**	**1.80**	15.87	0
60	1.33	2.52	21.94	0
75	1.69	2.56	26.11	0
90	2.11	2.64	33.67	0
100	1.59	2.52	38.78	0

Table 6.4: Influence of λ_g on the position response to a unit step input.

$\boldsymbol{\lambda_g}$	$\boldsymbol{S}$ (%)	$\boldsymbol{t_s}$ (s)	$\boldsymbol{t_c}$ (ms)	$\boldsymbol{e}$ (m)
0	-	-	**14.26**	1.03
100	7.76	2.16	15.02	0
300	2.00	2.28	16.34	0
500	1.07	**1.80**	15.87	0
700	0.50	2.12	16.29	0
1000	0.23	2.64	17.55	0
2000	**0.01**	5.12	16.21	0
5000	0.27	21.60	16.46	0

The influence of the regularisation parameters λ_g and λ_y is respectively detailed in Tables 6.4 and 6.5. In both situations, the importance of including regularisations in the optimisation problem of the DeePC algorithm is ratified. In the case of $\lambda_g = 0$, although the controller manages to stabilise the response, the latter presents a static error of 1.03m. Regarding λ_y, its non-utilisation leads to an infeasible optimisation problem and consequently, the DeePC controller fails to stabilise the system.

After a more detailed analysis, it becomes evident that the value of these regularisations parameters do not have a significant influence on t_c. Concerning λ_y, there is once again a threshold from which the results obtained are similar. This observation reinforces the notion that the crucial factor is to select a value of λ_y sufficiently high to render the optimisation problem feasible, regardless of its precise tuning. With regards to λ_g, an increase in this value produces a softer but significantly slower response. On the other hand, low values of this parameter lead to an oscillatory response. Subsequently, $\lambda_g = 500$ and $\lambda_y = 7.5 \times 10^8$ were chosen, since these values exhibited the best performance results.

Lastly, Tables 6.6 and 6.7 show the impact of the performance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, respectively, on the position response of the system to a unit step. It should be noted that the displayed values $\boldsymbol{Q}_{\text{old}}$ and $\boldsymbol{R}_{\text{old}}$ are referred to the values defined in (5.2).

Table 6.5: Influence of λ_y on the position response to a unit step input.

$\boldsymbol{\lambda_y}$	$\boldsymbol{S}$ (%)	$\boldsymbol{t_s}$ (s)	$\boldsymbol{t_c}$ (ms)	$\boldsymbol{e}$ (m)
0	-	-	-	-
1 $\times 10^5$	**0.17**	3.52	**15.40**	0
1 $\times 10^8$	1.05	**1.80**	16.12	0
7 $\times 10^8$	1.07	**1.80**	16.91	0
7.5 $\times 10^8$	1.07	**1.80**	15.87	0
8 $\times 10^8$	1.07	**1.80**	16.20	0
1 $\times 10^9$	1.07	**1.80**	16.24	0
1 $\times 10^{10}$	1.07	**1.80**	16.39	0

Table 6.6: Influence of $\boldsymbol{Q}$ on the position response to a unit step input.

$\boldsymbol{Q}$	$\boldsymbol{S}$ (%)	$\boldsymbol{t_s}$ (s)	$\boldsymbol{t_c}$ (ms)	$\boldsymbol{e}$ (m)
0.01$\cdot\boldsymbol{Q}_{\text{old}}$	29.20	21.00	**13.65**	0
0.1$\cdot\boldsymbol{Q}_{\text{old}}$	7.44	4.56	14.43	0
$\boldsymbol{Q}_{\text{old}}$	**1.07**	**1.80**	15.87	0
10$\cdot\boldsymbol{Q}_{\text{old}}$	21.77	2.20	16.43	0
100$\cdot\boldsymbol{Q}_{\text{old}}$	-	-	-	-

Table 6.7: Influence of $\boldsymbol{R}$ on the position response to a unit step input.

$\boldsymbol{R}$	$\boldsymbol{S}$ (%)	$\boldsymbol{t_s}$ (s)	$\boldsymbol{t_c}$ (ms)	$\boldsymbol{e}$ (m)
0.01$\cdot\boldsymbol{R}_{\text{old}}$	-	-	14.41	0.31
0.1$\cdot\boldsymbol{R}_{\text{old}}$	**0.01**	3.20	**14.27**	0
$\boldsymbol{R}_{\text{old}}$	1.07	**1.80**	15.87	0
10$\cdot\boldsymbol{R}_{\text{old}}$	9.85	3.60	16.85	0
100$\cdot\boldsymbol{R}_{\text{old}}$	26.05	9.88	16.24	0

In an initial analysis, it is denoted that the performance matrices used in the DeePC and LQR controllers demonstrate a significant impact on the system response. Regarding the tracking error cost matrix $\boldsymbol{Q}$, an increase in this matrix leads to an aggressive response, characterised by a high overshoot and a low settling time. Moreover, an excessive increase of this hyperparameter results in system instability. On the other hand, a severe decrease in the value of $\boldsymbol{Q}$ leads to an overly slow and oscillatory position response. In relation to the control effort cost matrix $\boldsymbol{R}$, changes in this parameter originate opposite outcomes compared to when the same alteration is applied to the value of the matrix $\boldsymbol{Q}$. Thus, a significant increase in $\boldsymbol{R}$ generates a slow and oscillatory response. Nevertheless, a decrease in the matrix $\boldsymbol{R}$, although not resulting in an unstable response, leads to a static error of 0.31m. Based on the above statements, the values chosen for these performance matrices are the same as those presented in (5.2).

6.3 Influence of Noise and Drag

Next, the objective of this section is to demonstrate the influence of noise in the measurements and translational drag on the performance of the DeePC controller. The results presented below were obtained through the implementation of the DeePC controller in the nonlinear variant of control architecture B. To account for the presence of translational drag, the nonlinear quadrotor dynamics model defined in (3.1) and (3.2) was utilised, where $\boldsymbol{C}_D = \text{diag}(1, 1, 2)$. The simulation of the noisy measurements was executed following the implementation approach described in the simulation model provided by the authors of [56].

Figure 6.7 illustrates the system response when subjected to noise. It can be immediately concluded that the presence of noise in the system measurements does not change the good performance already observed.

Subsequently, the comparison between the response of the system subject to noise with the one subject to both noise and drag is depicted in Figure 6.8.

Overall, the inclusion of drag in the simulation model resulted in a slower response with fewer oscillations. Specifically, in the case of the orientation response, the observed reduction in oscillations is highlighted, confirming the fact that drag dissipates energy from the system.

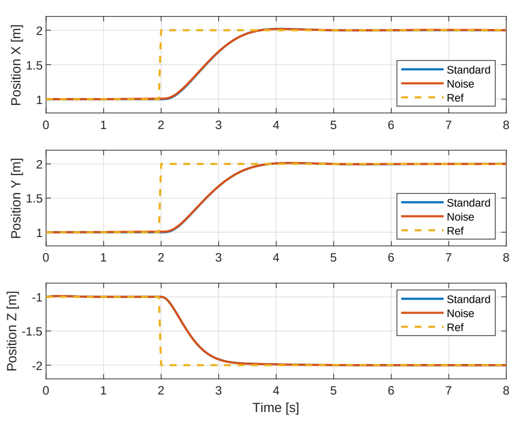
Finally, Figure 6.9 shows the influence of noise and drag on the system response to a square wave input. Once again, it is possible to observe satisfactory behaviour in both cases. Thus, it can be ratified that the performance of the DeePC controller is not adversely affected by the presence of translational drag and noise in the system measurements. Additionally, it is also noted that changing the type of input supplied to the system does not have a significant effect on the performance of this controller.

6.4 Comparison of the Performance of Control Methods

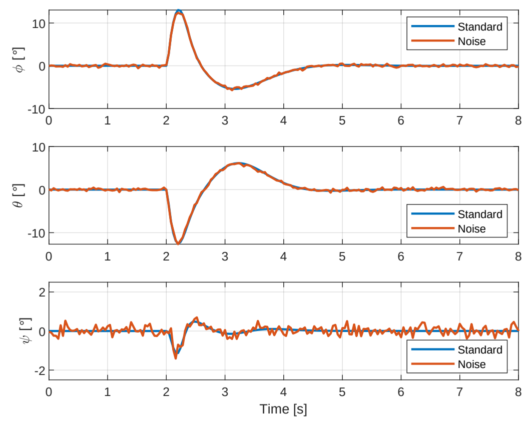
After a detailed study of the DeePC algorithm, this section aims to compare the performance of the DeePC controller with two conventional model-based controllers: the LQR and MPC controllers. In this thesis, these traditional controllers were implemented according to the information presented in Section 3.2.

Firstly, it should be noted that in this work we employ an LQR controller in conjunction with the DeePC controller as depicted in Figures 5.2 and 5.3. This choice is made since the implementation of the latter requires the existence of another simpler controller that can specifically maintain the quadrotor in the hovering state during the data collection phase.

Therefore, implementing both the DeePC and LQR controllers in the nonlinear control architecture B, using the same performance matrices defined in (5.2), and considering the presence of noise, the results illustrated in Figure 6.10 are obtained. It can be noted that the performance of the DeePC controller is superior in comparison to the LQR controller, especially in the responses of x and y positions. This observation leads to the conclusion that a successful implementation of the DeePC controller does not require the utilisation of a finely tuned conventional control method with good performance.

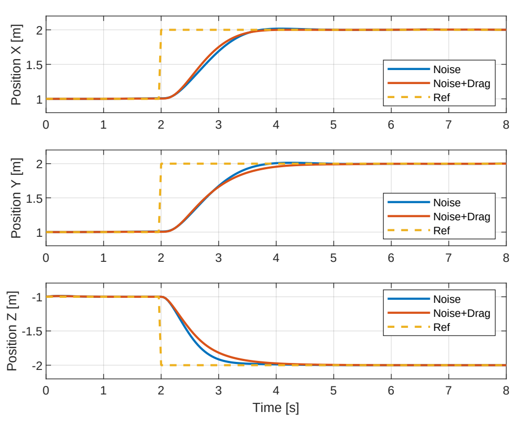


(a) Position response

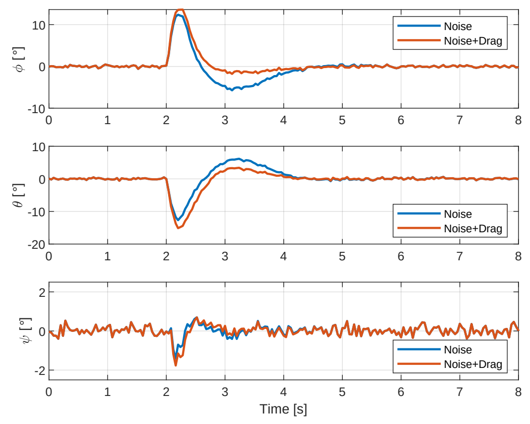


(b) Orientation response

Figure 6.7: Response to a unit step input subject to noise.

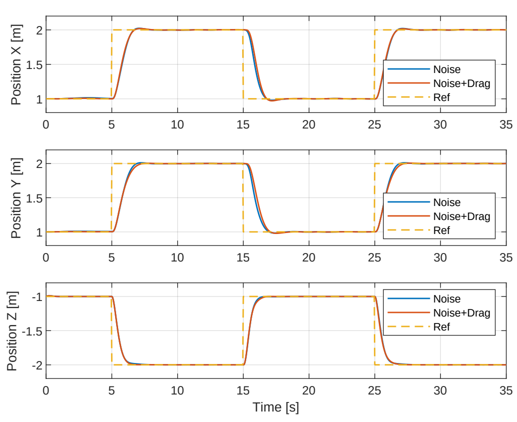


(a) Position response

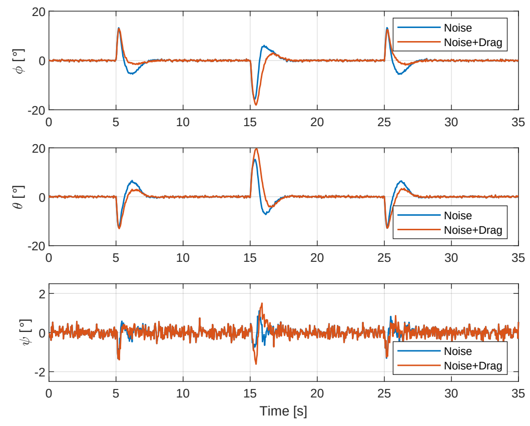


(b) Orientation response

Figure 6.8: Response to a unit step input subject to noise and drag.

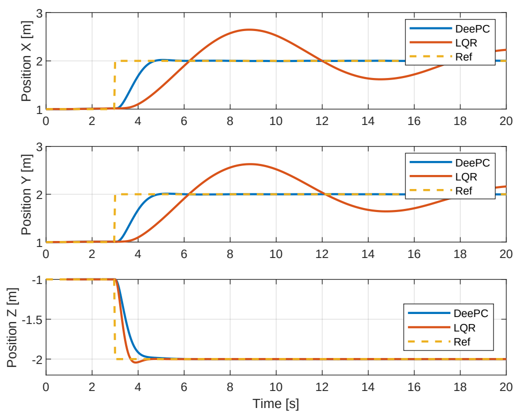


(a) Position response

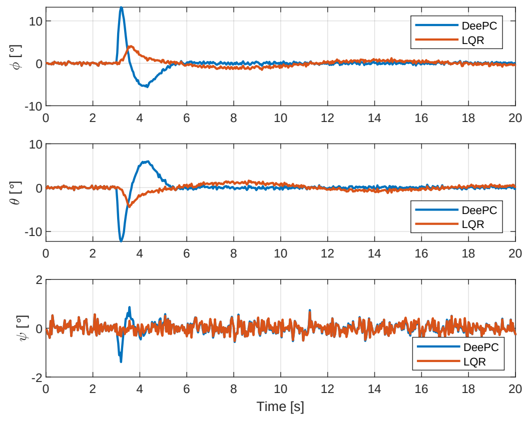


(b) Orientation response

Figure 6.9: Response to a square wave input subject to noise and drag.



(a) Position response subject to noise



(b) Orientation response subject to noise

Figure 6.10: Comparison of the performance of DeePC and LQR controllers.

Then, to ensure a fair comparison between the methods, the tuned LQR and MPC controllers were implemented. Regarding the LQR controller, the utilised state and control weighting matrices were respectively defined as follows:

$$
\begin{aligned}
\boldsymbol{Q}_{\text{LQR}} &= \text{diag}(50, 50, 500, 1, 1, 10, 20, 20, 40),\\
\boldsymbol{R}_{\text{LQR}} &= \text{diag}(1, 10, 10, 20).
\end{aligned}
$$

On the other hand, the linear and unconstrained version of the MPC controller was implemented using the following parameters:

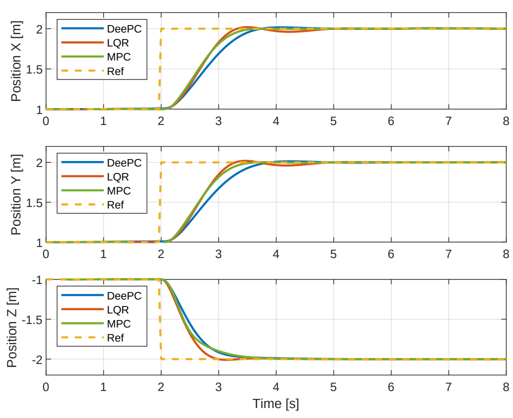
$$
\begin{aligned}
N &= 5,\\
\boldsymbol{Q}_{\text{MPC}} &= \text{diag}(220, 220, 350, 7, 7, 11, 10, 10, 5),\\
\boldsymbol{R}_{\text{MPC}} &= \text{diag}(0.5, 10, 10, 5),
\end{aligned}
$$

where N denotes the prediction time horizon.

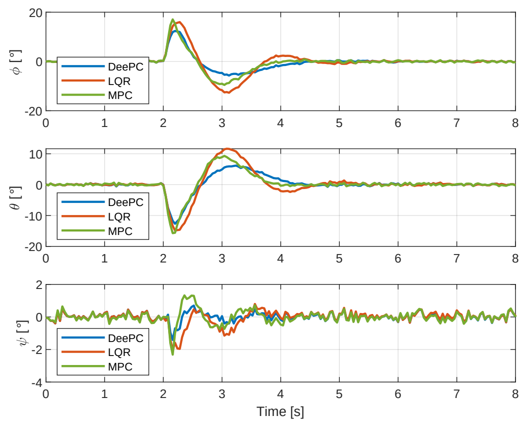
Figure 6.11 shows the system responses to a unit step employing the DeePC, LQR, and MPC controllers. It is important to highlight that the results presented were obtained using the same inner-loop controllers of the nonlinear control architecture B presented in Figure 5.5(a). The DeePC results were also achieved assuming the hyperparameters defined in (5.2) and Table 5.3.

The initial analysis shows a satisfactory performance of all controllers. It can also be inferred that the DeePC responses in general are characterised by a longer settling time when compared to the others. Nevertheless, it is important to emphasise that these results allow the conclusion that the DeePC algorithm exhibits similar performance to the other two conventional model-based control approaches, when the quadrotor system is subjected to simple trajectories. It should also be noted that the noisy responses observed, especially in the yaw angle case, are due to the presence of noise in the system measurements.

The next step is to study the robustness of the implemented methods to performance degradation

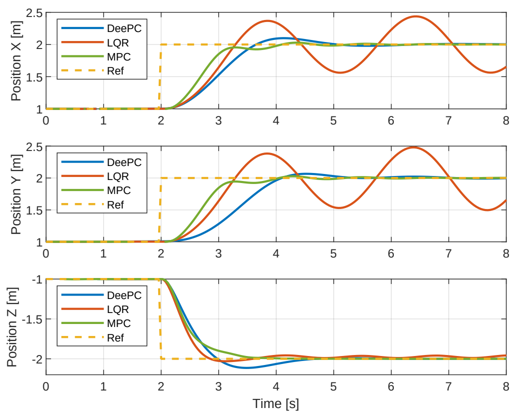


(a) Position response subject to noise

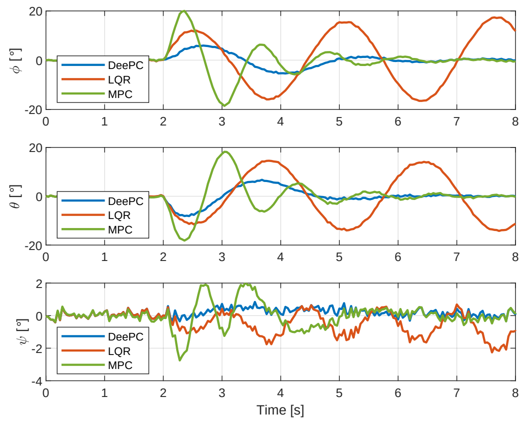


(b) Orientation response subject to noise

Figure 6.11: Comparison of the performance of DeePC, LQR, and MPC controllers.



(a) Position response subject to noise



(b) Orientation response subject to noise

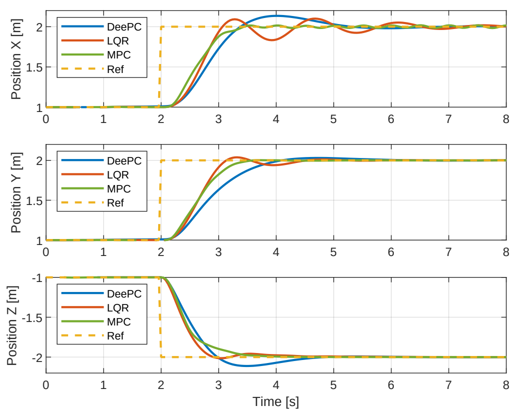
Figure 6.12: Comparison of the performance of DeePC, LQR, and MPC controllers when the inner angle controller gains are changed.

of the inner-loop controllers. Hence, the gains of these controllers were modified to slow down their response. Starting with the inner angle controller, its gains were obtained through the multiplication of the original values by $\frac{1}{3}$. Regarding the inner body rates PID controller, only the proportional gains were modified in the same proportion as for the inner angle controller.

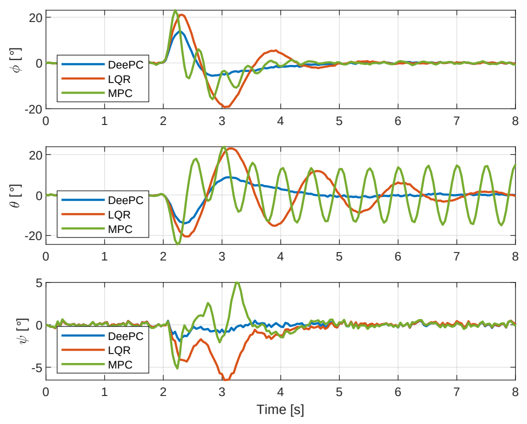
Figures 6.12 and 6.13 depict, respectively, the effect of the degradation of the inner angle controller and the inner body rates controller on the performance of the DeePC, LQR and MPC controllers.

Regarding Figure 6.12, the poor performance of the LQR controller is immediately observed, as it exhibits very oscillatory position and orientation responses. Additionally, this change in the inner angle controller leads to slightly oscillatory and slow responses from the DeePC and MPC controllers. From the orientation results, it can be inferred that the DeePC controller response stabilises faster than the MPC controller response. Thus, it is concluded that the DeePC controller is the most robust to performance degradation of the inner angle controller.

Furthermore, the analysis of Figure 6.13 concludes that the MPC controller shows an excessively



(a) Position response subject to noise



(b) Orientation response subject to noise

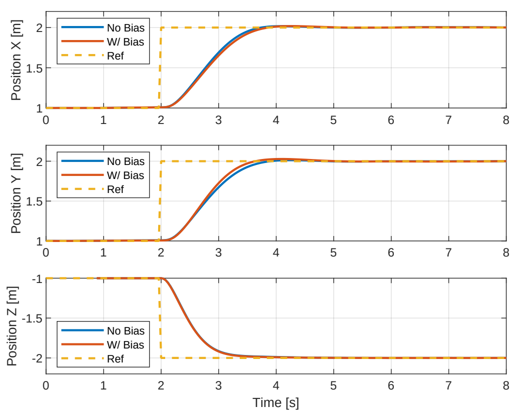
Figure 6.13: Comparison of the performance of DeePC, LQR, and MPC controllers when the gains of the inner body rates controller are changed.

oscillatory response, failing to converge the response of θ. Moreover, it is also observed that the modifications in the inner body rates controller lead to the oscillatory behaviour of the LQR controller, while in the DeePC controller, these changes result in a slow response with overshoot. In the orientation response, it is again inferred that the DeePC response is the fastest method to converge to zero.

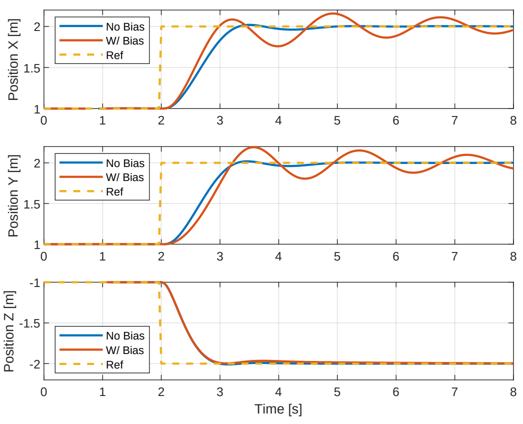
In summary, from the detailed analysis of Figures 6.12 and 6.13, it can be concluded that the DeePC controller is the least influenced method by the performance degradation of the inner loop controllers. This robustness demonstrates the adaptive properties of this data-driven control method, which is one of the advantages of DeePC over conventional model-based methods.

Finally, the effect of the presence of a bias in the system measurements was evaluated. Considering that it is not always possible to obtain accurate measurements of the yaw angle in a real-world experiment, it was decided to simulate a yaw miscalibration, by introducing a 25° offset between the true and measured yaw angles. It should be noted that this value is excessively high, but it was selected to highlight the differences between the results of the different controllers. In addition, it is also stated that the controllers implemented in the simulation are not aware of this calibration error. Figure 6.14 illustrates the simulation results obtained for each controller in the presence of a yaw calibration error of 25° .

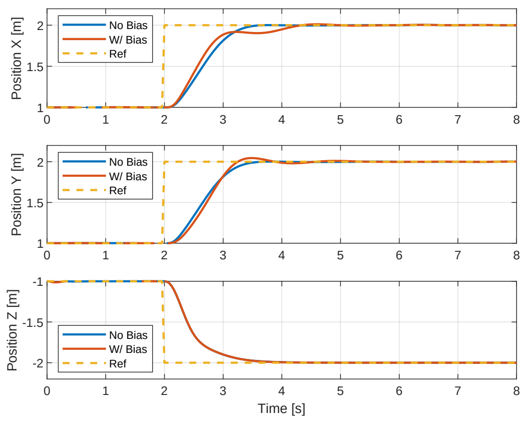
In the initial analysis, it is noticeable that the position z response is not affected by the yaw offset error, as expected. Moreover, it can be concluded that the presence of this yaw error in system measurements yields a very oscillatory LQR response. For the MPC controller, a deterioration in its performance is also observed, particularly in the position x response. Although not as obvious as in the LQR case, the response associated with the MPC controller becomes more oscillatory and with a longer settling time compared to its original response without a yaw calibration error. Lastly, it is possible to note that the DeePC algorithm provides robustness to this bias error since its response remains the same. Thus, it can be concluded once again that the DeePC algorithm has the ability to adapt to the unknown operating conditions of the system, in contrast to the other model-based control methods.



(a) Position response for DeePC controller



(b) Position response for LQR controller



(c) Position response for MPC controller

Figure 6.14: Comparison of the performance of DeePC, LQR, and MPC controllers when there is a yaw calibration error of 25° .

Chapter 7

Conclusions

The main goal of this thesis was to investigate and develop a data-driven control technique suitable for the control of a quadrotor system. For this purpose, the implementation of the DeePC algorithm was first proposed, and a very realistic simulation was conducted to evaluate the proposed method on models of different complexity. Finally, a detailed comparison between the DeePC and other model-based methods was performed, to situate this framework in relation to conventional control methods.

The implemented DeePC algorithm, presented in Chapter 4, uses a finite data set rearranged into Hankel matrices to learn the behaviour of the unknown system. Subsequently, it applies real-time output feedback to compute optimal input controls that guide the system towards a desired trajectory while satisfying the system's constraints. Moreover, for nonlinear systems that are corrupted by process noise, such as quadrotors, it is necessary to include some regularisations in the optimisation problem of the DeePC algorithm. At the end of this chapter, the data collection step, which is critical to the successful implementation of this data-driven control algorithm, was also detailed. The input signals used to excite the system in this stage consist of a PRBS excitation signal added to an existing simple controller that maintains the quadrotor in the hover state.

A detailed overview of the realistic simulation model used was presented in Chapter 5. This simulation model, which includes accurate modelling of rotor dynamics, actuation constraints, translational drag, measurement noise, and inner-loop controller dynamics, was developed using the software MATLAB/Simulink to evaluate the performance of the proposed method. In addition, the implementation of the DeePC controller demonstrates that it does not require access to full-state measurements, unlike traditional control methods. Furthermore, a basic simulation test was conducted to analyse the system response to a unit step in the position variables using the DeePC controller implemented in different control architectures. The obtained results demonstrate adequate nonlinear and linear responses, characterised by an acceptable settling time and minimal overshoot.

The detailed simulation results of the performance of the DeePC algorithm were reported in Chapter 6. Firstly, the analysis of the data collection phase concludes that the acquired data do not need to show a large variation in the z position to verify the persistence of excitation condition. In addition to the importance of adjusting the frequency band and amplitudes of the excitation signals, it is concluded that the acquisition of data from independent excitation signals is also critical to the proper functioning of the DeePC controller. Then, the effect of each hyperparameter on the performance of the DeePC controller was addressed. Consequently, the latter is evaluated by analysing some relevant metrics, such as the average overshoot, the maximum settling time, the average algorithm computation time, and the average static error. In the results obtained, it is important to highlight that the non-use of the regularisation parameters leads to a degradation of the controller performance, thus corroborating their important inclusion in the DeePC optimisation problem. Furthermore, it can also be concluded that the performance of the DeePC controller for a square wave reference is not adversely affected by the presence of drag and noise in the system.

Finally, the comparison between the DeePC controller and the conventional LQR and MPC controllers was described in Chapter 6. It is concluded that a simple controller, used to maintain the quadrotor in the hover state during the data collection step, does not need to be fine-tuned to yield a good performance of the DeePC controller. Furthermore, it is also confirmed that the DeePC algorithm performs similarly to MPC and LQR model-based control approaches when the quadrotor system is subjected to simple trajectories. This data-driven method demonstrates greater robustness to the performance degradation of inner-loop controllers and the presence of a yaw calibration error than the conventional approaches. Thus, it can be concluded that the DeePC algorithm shows the ability to adapt to the unknown operating conditions of the system, in contrast to the other model-based control methods. Nevertheless, it is important to note that, despite the positive results presented in this thesis, the implementation of the DeePC method becomes impractical when tracking aggressive trajectories. Moreover, for more complex nonlinear systems, the data collection step may fail to capture the essential dynamics of the system or, on the other hand, the dimension of the Hankel matrices may become excessively large, leading to a significant increase in the computational time of the solver.

7.1 Future Work

Future developments could be introduced to improve the performance evaluation of the proposed method. First, the presented algorithms could be validated through SITL simulations and experimental results. Although the simulation model used was realistic, conducting experiments in an environment closer to the real world would provide more guarantees of the performance of the proposed algorithm.

Second, to increase the applicability of the DeePC method, the data collected for the Hankel matrices could be updated online, resulting in an algorithm that is more adaptable to various unexpected scenarios. By performing the data collection step online, any changes in the environmental conditions around the system will be transmitted to the controller, leading to an improvement in its performance.

Finally, other data-driven control approaches that can outperform the DeePC method could be investigated and implemented. For this purpose, the development of a data-driven control method based on machine learning techniques is proposed, following the trend reported in the literature.

Bibliography

[1] S. A. H. Mohsan, M. Khan, F. Noor, I. Ullah, and M. Alsharif. Towards the unmanned aerial vehicles (uavs): A comprehensive review. *Drones*, 6, 2022. doi: https://doi.org/10.3390/drones6060147.

[2] M. Kunovjanek and C. Wankmüller. Containing the covid-19 pandemic with drones - feasibility of a drone enabled back-up transport system. *Transport Policy*, 106, 2021. doi: https://doi.org/10.1016/j.tranpol.2021.03.015.

[3] S. A. H. Mohsan, Q. U. A. Zahra, M. Khan, M. Alsharif, I. Elhaty, and A. Jahid. Role of drone technology helping in alleviating the covid-19 pandemic. *Micromachines*, 13:1593, 2022. doi: https://doi.org/10.3390/mi13101593.

[4] Z. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013. doi: https://doi.org/10.1016/j.ins.2012.07.014.

[5] J. Coulson, J. Lygeros, and F. Dörfler. Data-enabled predictive control: In the shallows of the deepc. *2019 18th European Control Conference (ECC)*, pages 307–312, 2019. doi: https://doi.org/10.23919/ECC.2019.8795639.

[6] H. Nguyen, T. Quyen, V.-C. Nguyen, A. Le, H. Tran, and M. Nguyen. Control algorithms for uavs: A comprehensive survey. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, 7, 2020. doi: https://doi.org/10.4108/eai.18-5-2020.164586.

[7] K. Åström and T. Hägglund. *PID Controllers: Theory, Design, and Tuning*. ISA - The Instrumentation, Systems and Automation Society, 1995. ISBN 1-55617-516-7.

[8] 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 Vol.5, 2004. doi: https://doi.org/10.1109/ROBOT.2004.1302409.

[9] S. Bouabdallah, A. Noth, and R. Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3:2451 – 2456 vol.3, 2004. doi: https://doi.org/10.1109/IROS.2004.1389776.

[10] S. Bouabdallah, P. Murrieri, and R. Siegwart. Towards autonomous indoor micro vtol. *Auton. Robots*, 18:171–183, 2005. doi: https://doi.org/10.1007/s10514-005-0724-z.

[11] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proc. of the AIAA Guidance, Navigation, and Control Conference*, 2007. doi: https://doi.org/10.2514/6.2007-6461.

[12] T. Zhang, Y. Kang, M. Achtelik, K. Kühnlenz, and M. Buss. Autonomous hovering of a vision/imu guided quadrotor. In *2009 International Conference on Mechatronics and Automation*, 2009. doi: https://doi.org/10.1109/ICMA.2009.5246422.

[13] I. Sadeghzadeh, A. Mehta, and Y. Zhang. Fault/damage tolerant control of a quadrotor helicopter uav using model reference adaptive control and gain-scheduled pid. In *AIAA Guidance, Navigation, and Control Conference*, 2011. ISBN 978-1-60086-952-5. doi: https://doi.org/10.2514/6.2011-6716.

[14] F. Goodarzi, D. Lee, and T. Lee. Geometric nonlinear pid control of a quadrotor uav on se(3). In *Proceeding of European control conference*, pages 3845–3850, 2013. doi: https://doi.org/10.23919/ECC.2013.6669644.

[15] H. Yang, L. Cheng, Y. Xia, and Y. Yuan. Active disturbance rejection attitude control for a dual closed-loop quadrotor under gust wind. *IEEE Transactions on Control Systems Technology*, PP: 1–6, 2017. doi: https://doi.org/10.1109/TCST.2017.2710951.

[16] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006. ISBN 978-1-62410-046-8. doi: https://doi.org/10.2514/6.2006-6200.

[17] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke. A prototype of an autonomous controller for a quadrotor uav. In *2007 European Control Conference (ECC)*, pages 4001–4008, 2007. doi: https://doi.org/10.23919/ECC.2007.7068316.

[18] B. Yu, Y. Zhang, I. Minchala, and Y. Qu. Fault-tolerant control with linear quadratic and model predictive control techniques against actuator faults in a quadrotor uav. In *2013 Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 661–666, 2013. doi: https://doi.org/10.1109/SysTol.2013.6693925.

[19] L. Martins, C. Cardeira, and P. Oliveira. Linear quadratic regulator for trajectory tracking of a quadrotor. *IFAC-PapersOnLine*, 52:176–181, 2019. doi: https://doi.org/10.1016/j.ifacol.2019.11.195.

[20] M. Chen and M. Huzmezan. A combined mbpc/2 dof h infinity controller for a quad rotor uav. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2003. ISBN 978-1-62410-090-1. doi: https://doi.org/10.2514/6.2003-5520.

[21] A. Mokhtari, A. Benallegue, and B. Daachi. Robust feedback linearization and gh[infinity] controller for a quadrotor unmanned aerial vehicle. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 57, pages 1198 – 1203, 2005. doi: https://doi.org/10.1109/IROS.2005.1545112.

[22] G. V. Raffo, M. G. Ortega, and F. R. Rubio. Backstepping/nonlinear h[infinity] control for path tracking of a quadrotor unmanned aerial vehicle. In *2008 American Control Conference*, pages 3356–3361, 2008. doi: https://doi.org/10.1109/ACC.2008.4587010.

[23] G. Raffo, M. Ortega, and F. Rubio. An integral predictive/nonlinear h[infinity] control structure for a quadrotor helicopter. *Automatica*, 46:29–39, 2010. doi: https://doi.org/10.1016/j.automatica.2009.10.018.

[24] H. K. Khalil. *Nonlinear Systems*. Prentice-Hall, Upper Saddle River, NJ, 3nd edition, 2002.

[25] S. Bouabdallah and R. Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2247–2252, 2005. doi: https://doi.org/10.1109/ROBOT.2005.1570447.

[26] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 153–158, 2007. doi: https://doi.org/10.1109/IROS.2007.4399042.

[27] T. Madani and A. Benallegue. Backstepping control for a quadrotor helicopter. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3255–3260, 2006. doi: https://doi.org/10.1109/IROS.2006.282433.

[28] V. Lippiello, F. Ruggiero, and D. Serra. Emergency landing for a quadrotor in case of a propeller failure: A backstepping approach. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4782–4788, 2014. doi: https://doi.org/10.1109/IROS.2014.6943242.

[29] R. Xu and U. Ozguner. Sliding mode control of a quadrotor helicopter. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 4957–4962, 2006. doi: https://doi.org/10.1109/CDC.2006.377588.

[30] R. López-Gutiérrez, A. Rodriguez-Mata, S. Salazar, I. González, and R. Lozano. Robust quadrotor control: Attitude and altitude real-time result. *Journal of Intelligent & Robotic Systems*, 88, 2017. doi: https://doi.org/10.1007/s10846-017-0520-y.

[31] J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[32] E. Altug, J. Ostrowski, and R. Mahony. Control of a quadrotor helicopter using visual feedback. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 72–77 vol.1, 2002. doi: https://doi.org/10.1109/ROBOT.2002.1013341.

[33] D. Lee and S. Sastry. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. international journal of control, automation and systems, 7(3), 419-428. *International Journal of Control, Automation and Systems*, 7:419–428, 2009. doi: https://doi.org/10.1007/s12555-009-0311-8.

[34] H. Voos. Nonlinear control of a quadrotor micro-uav using feedback-linearization. In *2009 IEEE International Conference on Mechatronics*, pages 1–6, 2009. doi: https://doi.org/10.1109/ICMECH.2009.4957154.

[35] E. F. Camacho and C. Bordons. *Modern Predictive Control*. Springer-Verlag, 2nd edition, 2007.

[36] K. Alexis, G. Nikolakopoulos, and A. Tzes. Model predictive quadrotor control: Attitude, altitude and position experimental studies. *IET Control Theory and Applications*, 6(12):1812–1827, 2012. ISSN 1751-8644. doi: https://doi.org/10.1049/iet-cta.2011.0348.

[37] M. Abdolhosseini, Y. Zhang, and C. Rabbath. An efficient model predictive control scheme for an unmanned quadrotor helicopter. *Journal of Intelligent & Robotic Systems*, 70, 2013. doi: https://doi.org/10.1007/s10846-012-9724-3.

[38] J. Pinto. Model predictive control strategies for aggressive parcel relay maneuvers using drones. Master's thesis, Instituto Superior Técnico, Lisbon, Portugal, 2021.

[39] F. Nan, S. Sun, P. Foehn, and D. Scaramuzza. Nonlinear mpc for quadrotor fault-tolerant control. *IEEE Robotics and Automation Letters*, 7, 2022. doi: https://doi.org/10.1109/LRA.2022.3154033.

[40] R. J. Heaston. *Modern Control Theory; State-of-the-Art Review*. TACTICAL WEAPONS GUIDANCE AND CONTROL INFORMATION ANALYSIS CENTER CHICAGO IL, 1995.

[41] G. Gremillion and J. Humbert. System identification of a quadrotor micro air vehicle. In *AIAA Atmospheric Flight Mechanics Conference*, 2010. ISBN 978-1-62410-151-9. doi: https://doi.org/10.2514/6.2010-7644.

[42] I. Lopez-Sanchez, J. Montoya-Cháirez, R. Pérez-Alcocer, and J. Moreno-Valenzuela. Experimental parameter identifications of a quadrotor by using an optimized trajectory. *IEEE Access*, 8:167355–167370, 2020. doi: https://doi.org/10.1109/ACCESS.2020.3023643.

[43] D. Six, S. Briot, J. Erskine, and A. Chriette. Identification of the propeller coefficients and dynamic parameters of a hovering quadrotor from flight data. *IEEE Robotics and Automation Letters*, 5(2):1063–1070, 2020. doi: https://doi.org/10.1109/LRA.2020.2966393.

[44] M. Burri, M. Bloesch, Z. Taylor, R. Siegwart, and J. Nieto. A framework for maximum likelihood parameter identification applied on mavs. *Journal of Field Robotics*, 35, 2017. doi: https://doi.org/10.1002/rob.21729.

[45] M. Rigter, B. Morrell, R. G. Reid, G. B. Merewether, T. Tzanetos, V. Rajur, K. Wong, and L. H. Matthies. An autonomous quadrotor system for robust high-speed flight through cluttered environments without gps. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5227–5234, 2019. doi: https://doi.org/10.1109/IROS40897.2019.8968127.

[46] 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. doi: https://doi.org/10.1109/ICCA.2013.6565145.

[47] J. Angarita, K. Schroeder, and J. Black. Quadrotor model generation using system identification techniques. In *2018 AIAA Modeling and Simulation Technologies Conference*, 2018. doi: https://doi.org/10.2514/6.2018-1917.

[48] N. Abas, A. Legowo, Z. Ibrahim, N. Rahim, and A. Kassim. Modeling and system identification using extended kalman filter for a quadrotor system. *Applied Mechanics and Materials*, 313-314: 976–981, 2013. doi: https://doi.org/10.4028/www.scientific.net/AMM.313-314.976.

[49] 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. doi: https://doi.org/10.1016/j.sysconle.2004.09.003.

[50] I. Markovsky and P. Rapisarda. Data-driven simulation and control. *International Journal of Control*, 81(12), 2008. doi: https://doi.org/10.1080/00207170801942170.

[51] M. Campi, A. Lecchini, and S. Savaresi. Virtual reference feedback tuning: a direct method for the design of feedback controllers. *Automatica*, 38:1337–1346, 2002. doi: https://doi.org/10.1016/S0005-1098(02)00032-8.

[52] P. Panizza, D. Invernizzi, F. Riccardi, S. Formentin, and M. Lovera. Data-driven attitude control law design for a variable-pitch quadrotor. In *2016 American Control Conference (ACC)*, pages 4434–4439, 2016. doi: https://doi.org/10.1109/ACC.2016.7525620.

[53] Y. Al Younes, A. Drak, H. Noura, A. Rabhi, and A. El Hajjaji. Model-free control of a quadrotor vehicle. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1126–1131, 2014. doi: https://doi.org/10.1109/ICUAS.2014.6842366.

[54] Y. Younes, A. Drak, H. Noura, A. Rabhi, and A. El hajjaji. Robust model-free control applied to a quadrotor uav. *Journal of Intelligent & Robotic Systems*, 84, 2016. doi: https://doi.org/10.1007/s10846-016-0351-2.

[55] W. Favoreel, B. D. Moor, and M. Gevers. Spc: Subspace predictive control. *IFAC Proceedings Volumes*, 32(2):4004–4009, 1999. ISSN 1474-6670. doi: https://doi.org/10.1016/S1474-6670(17)56683-5. 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July.

[56] E. Elokda, J. Coulson, P. N. Beuchat, J. Lygeros, and F. Dörfler. Data-enabled predictive control for quadcopters. *International Journal of Robust and Nonlinear Control*, 31:8916 – 8936, 2021. doi: https://doi.org/10.1002/rnc.5686.

[57] J. Coulson, J. Lygeros, and F. Dörfler. Distributionally robust chance constrained data-enabled predictive control. *IEEE Transactions on Automatic Control*, 67(7):3289–3304, 2022. doi: https://doi.org/10.1109/TAC.2021.3097706.

[58] L. Huang, J. Lygeros, and F. Dörfler. Robust and kernelized data-enabled predictive control for nonlinear systems. *arXiv preprint arXiv:2206.01866*, 2022. doi: https://doi.org/10.48550/arXiv.2206.01866.

[59] H. Waarde. Beyond persistent excitation: Online experiment design for data-driven modeling and control. *IEEE Control Systems Letters*, PP:1–1, 2021. doi: https://doi.org/10.1109/LCSYS.2021.3073860.

[60] T. Dierks and S. Jagannathan. Neural network output feedback control of a quadrotor uav. In *2008 47th IEEE Conference on Decision and Control*, pages 3633–3639, 2008. doi: https://doi.org/10.1109/CDC.2008.4738814.

[61] T. Dierks and S. Jagannathan. Output feedback control of a quadrotor uav using neural networks. *IEEE Transactions on Neural Networks*, 21(1):50–66, 2010. doi: https://doi.org/10.1109/TNN.2009.2034145.

[62] S. Bansal, A. Akametalu, F. Jiang, F. Laine, and C. Tomlin. Learning quadrotor dynamics using neural network for flight control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016. doi: https://doi.org/10.1109/CDC.2016.7798978.

[63] 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. doi: https://doi.org/10.1109/TNNLS.2019.2891257.

[64] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza. Data-driven mpc for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776, 2021. doi: https://doi.org/10.1109/LRA.2021.3061307.

[65] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza. Neurobem: Hybrid aerodynamic quadrotor model. In *Robotics: Science and Systems 2021*, 2021. doi: https://doi.org/10.15607/RSS.2021.XVII.042.

[66] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023. doi: https://doi.org/10.1109/LRA.2023.3246839.

[67] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 528–535, 2016. doi: https://doi.org/10.1109/ICRA.2016.7487175.

[68] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 908–919, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[69] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza. A benchmark comparison of learned control policies for agile quadrotor flight. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10504–10510, 2022. doi: https://doi.org/10.1109/ICRA46639.2022.9811564.

[70] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine*, 19(3):20–32, 2012. doi: https://doi.org/10.1109/MRA.2012.2206474.

[71] C. Chen. *Linear System Theory and Design*. HRW series in electrical and computer engineering. Holt, Rinehart, and Winston, 1984. ISBN 9780030602894.

[72] K. Ogata. *Modern Control Engineering*. Prentice Hall, 5th edition, 2010.

[73] R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, 2002.

[74] L. Ljung. *System identification*. Springer, 1998.

[75] MATLAB. *9.12.0.2039608 (R2022a)*. The MathWorks Inc., Natick, Massachusetts, 2022.

[76] S. Documentation. Simulation and model-based design, 2022. URL `https://www.mathworks.com/products/simulink.html`.

[77] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. doi: 10.1007/s12532-020-00179-2. URL `https://doi.org/10.1007/s12532-020-00179-2`.

[78] W. Z. Fum. Implementation of simulink controller design on iris+ quadrotor. Master's thesis, Naval Postgraduate School, Monterey, California, USA, 2015. URL `https://hdl.handle.net/10945/47258`.

[79] PX4 Autopilot. Open source autopilot for drones. [Online] Available: `https://px4.io/`, Last accessed on 2023-05-22.