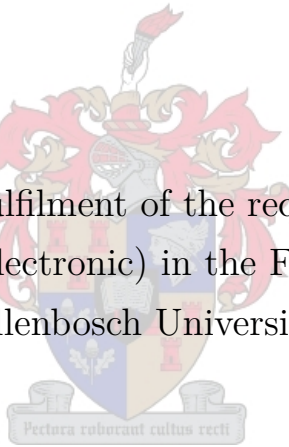
Data-Driven System Identification and Model Predictive Control of a Multirotor with an Unknown Suspended Payload

Jakobus Murray Louw
19977476



Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University.

Supervisor: Dr H. W. Jordaan
Department of Electrical and Electronic Engineering

March 2022

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

19977476 Studentenommer / *Student number*	 Handtekening / *Signature*
JM Louw Voorletters & van / *Initials & surname*	**1 November 2021** Datum / *Date*

Abstract

This thesis considers the problem of stabilised control for a multirotor with an unknown suspended payload. The controller assumes no prior knowledge of the payload dynamics. The swinging payload negatively affects the multirotor flight dynamics by inducing oscillations in the system. An adaptive control architecture is proposed to damp these oscillations and produce stable flight with different unknown payloads. A data-driven system identification method forms part of the architecture and is demonstrated in simulation and with practical flight data. Model Predictive Control (MPC) is applied for swing damping control and is verified with Hardware-in-the-Loop (HITL) simulations.

A parameter estimator and Linear Quadratic Regulator (LQR) is used in the baseline architecture. The LQR uses a predetermined model of the payload, which is completed with estimates of the payload mass and cable length. The architecture proposed in this work uses Dynamic Mode Decomposition with Control (DMDc) to estimate a linear state-space model and approximate the unknown dynamics. A short length of flight data is used for training. An MPC uses the data-driven model to control the multirotor and damp the payload oscillations.

A Simulink™ simulator was designed and verified with practical data. Within simulations, both the baseline and proposed architectures produced near swing-free multirotor control with different payload masses and cable lengths. Even with a dynamic payload producing irregular oscillations, both methods resulted in stabilised control. Both architectures also showed effective disturbance rejection. Despite the baseline method using an accurate predetermined model, the proposed method produced equal performances without prior knowledge of the dynamics. The baseline performance degraded significantly for a different multirotor mass because this parameter was not considered as an unknown. In contrast, the proposed method consistently produced good performances.

The accuracy of the DMDc models was verified with practical flight data. The proposed control architecture was also demonstrated in HITL simulations. The hardware executed the MPC at the desired speed, producing near swing-free control within a Gazebo simulator. Overall, it was shown that the proposed control architecture is practically feasible. Without knowledge of the payload dynamics, a data-driven model is used with MPC for effective swing damping control with a multirotor.

Uittreksel

Publication

Sections of this work form part of the following paper and was accepted for publication:

J.M. Louw and H.W. Jordaan, "Data-Driven System Identification and Model Predictive Control of a Multirotor with an Unknown Suspended Payload," in *Control Conference Africa (CCA)*, IFAC, Dec 2021, Magaliesburg, South Africa.

Acknowledgements

Contents

List of figures

List of tables

Abbreviations

$\mathcal{H}_\infty$	H-Infinity
2D	Two-Dimensional
3D	Three-Dimensional
6DOF	Six-Degrees-of-Freedom
ADC	Analog to Digital Converter
ADRC	Active Disturbance Rejection Control
AIC	Akaike's Information Criteria
ARC	Adaptive Robust Control
BBEL	Bidirectional Brain Emotional Learning
BFBEL	Bidirectional Fuzzy Brain Emotional Learning
BIC	Bayesian Information Criteria
BISMC	Back-Stepping Integral Sliding Mode Control
BS	Back-stepping
CAFVI	Continuous Action Fitted Value Iteration
CoM	Centre-of-Mass
CPU	Central Processing Unit
DCM	Direct Cosine Matrix
DMD	Dynamic Mode Decomposition
DMDc	Dynamic Mode Decomposition with Control
EB	Energy-Based
EI	Extra Insensitive
EKF	Extended Kalman Filter
ESL	Electronic System Laboratory
ESO	Extended State Observer
FC	Flight Controller
FFA	Federal Aviation Administration
FFT	Fast Fourier Transform
GPS	Global Positioning System
HAVOK	Hankel Alternative View Of Koopman
HAVOKc	Hankel Alternative View Of Koopman with Control

HITL	Hardware-in-the-Loop
ILC	Iterative Learning Control
iLQR	iterative Linear Quadratic Regulator
IMU	Inertial Measurement Unit
LPF	Low Pass Filter
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
MPC	Model Predictive Control
MRAC	Model Reference Adaptive Control
MRAE	Mean Relative Absolute Error
MSE	Mean Squared Error
MV	Munipulated Variable
NED	North-East-Down
NF	Notch Filter
NMAE	Normalised Mean Absolute Error
NZV	Negative Zero Vibration
OBC	On-Board Computer
ODE	Ordinary Differential Equation
PD	Proportional Derivative
PE	Percentage Error
PID	Proportional Integral Derivative
POD	Proper Orthogonal Decomposition
PWM	Pulse-Width Modulation
QGC	QGroundControl
QP	Quadratic Program
RAM	Random Access Memory
RC	Radio Control
RCAC	Retrospective Cost Adaptive Control
RISE	Robust Integral of the Sign of the Error
RL	Reinforcement Learning
RLS	Recursive Least Squares
ROS	Robot Operating System
SINDy	Sparse Identification of non-linear Dynamics

SISO	Single Input Single Output
SITL	Software-in-the-Loop
SMC	Sliding Mode Control
SVD	Singular Value Decomposition
UART	Universal Asynchronous Receiver Transmitter
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
USB	Universal Serial Bus
VTOL	Vertical Takeoff and Landing
ZV	Zero Vibration

Chapter 1

Introduction

1.1. Background

Recent years have seen a rise in the popularity of payload transportation with Unmanned Aerial Vehicles (UAVs) [11]. These payloads are usually categorised as either a sensor or freight [12]. Sensors like cameras or meteorological instruments can be carried by a UAVs for aerial photography or surveying. Payloads carried as freight include pesticides sprayed over agricultural land, medical parcels carried to remote areas or consumer deliveries.

Commercial package deliveries with UAVs have become especially popular. In 2015, the first Federal Aviation Administration (FFA) approved drone delivery was successfully completed by Flirtey in the United States [13]. Domino's pizza has also been delivered by Flirtey multirotors in New Zealand [14]. Another commercial example includes Wing food deliveries with multirotors in Australia [15].

Multirotor UAVs are commonly used for payload transportation tasks due to their hover and Vertical Takeoff and Landing (VTOL) abilities. In some applications, a payload is rigidly attached to the UAV. The flying characteristics of multirotors also allow them to transport suspended payloads, which is useful for arbitrarily shaped payloads or for delivering payloads without landing. In this configuration, the payload is suspended below the vehicle with a cable and the payload is free to swing during flight. This oscillatory motion affects the flight dynamics of the multirotor and makes stabilised control a challenging task.

Control becomes even more difficult with increased uncertainty of the payload dynamics. In some applications the payload dynamics are well known and constant, hence a controller can be designed based on an accurate predetermined model of the dynamics. However, package delivery applications often involve uncertainty of the payload parameters. Specific payloads such as elongated payloads or fluid containers add even more uncertainty to the system by inducing interesting dynamics which are also unknown before a flight. This significantly affects the flight dynamics of a multirotor and the controller may need to account for this uncertainty for effective control.

In summary, multirotor payload transportation is becoming increasingly popular. The suspended payload configuration offers strategic benefits but increases the difficulty of the control task. Furthermore, the uncertainty in payload dynamics makes the control task more challenging. In this study, a control architecture will be designed to address this problem.

1.2. Project definition and objectives

This project aims to design and implement a control architecture for stabilised control of a multirotor with an unknown suspended payload. The payload uncertainty should include parameter uncertainty and model uncertainty. Furthermore, the oscillatory motion of the payload significantly affects the multirotor dynamics. The proposed controller should be compared to previous work involving a swing damping controller for a suspended payload with an unknown mass and cable length.

In contrast to the architecture based on a predetermined model with only two unknown parameters, the proposed architecture should assume no prior knowledge of the suspended payload dynamics. A data-driven approach should be applied to estimate a dynamical model of the unknown dynamics. Based on the estimated model, a controller should stabilise the multirotor by actively damping the payload swing angles.

Therefore, the research objectives are stated as:

1. Investigate the literature regarding multirotor-payload controllers and specifically consider solutions for unknown suspended payload dynamics.
2. Derive a dynamical model to describe a multirotor with a suspended payload.
3. Identify and implement a baseline architecture with a system identification and control method for this system in simulation.
4. Design a data-driven system identification method for this system and implement it in simulation.
5. Design a controller based on the proposed system identification model and implement it in simulation.
6. Identify a hardware platform and software toolchain to implement the proposed control architecture.
7. Implement and verify the data-driven system identification method with experimental data from practical flights.

8. Implement, simulate and verify the controller algorithms on the practical hardware for effective swing damping control of the unknown suspended payload system.

1.3. Thesis outline

Chapter 1 provides the background of this research, the project definition and objectives, and the thesis outline.

Chapter 2 presents a study of the literature regarding multirotor payload transportation, with a focus on suspended payloads and uncertain payload dynamics.

Chapter 3 contains a derivation of a mathematical model for the multirotor and suspended payload dynamics, which is used for simulations and controller design.

Chapter 4 describes the baseline and the proposed system identification methods considered in this thesis. Furthermore, the performances of these methods are evaluated based on tests with simulation data.

Chapter 5 describes the different controllers and the corresponding controller design processes used in this project. Using the system identification models from the previous chapter, the controllers are also applied to the multirotor-payload system in simulation and the results are compared.

Chapter 6 provides an overview of the practical multirotor setup used for experimental work with the proposed algorithms. Thereby, the hardware components, software toolchain, and HITL simulations are discussed.

Chapter 7 presents and discusses the experimental results from implementing the system identification methods with practical flight data. HITL results are also presented to test the controller algorithms with the practical hardware and software systems.

Chapter 8 provides a summary of the work in this thesis. The major conclusions of this work are also presented and future recommendations are discussed.

Chapter 2

Literature study

This chapter will present a study of the literature regarding payload transportation with multirotors. Firstly, an overview of different payload configurations and control techniques for payload transportation will be discussed. Thereafter, the study will focus on control techniques that consider suspended payloads. A summary will be provided of different swing damping controllers proposed for the multirotor-payload system and a few literature trends will be highlighted. The chapter will conclude with a summary of the literature study and focus areas of this thesis.

2.1. Payload transportation with multirotors

The usage of Unmanned Aerial Vehicles (UAVs) for payload transportation has significantly grown in popularity over recent years [11]. Multirotor UAVs are specifically useful for many transportation applications due to their agility and Vertical Takeoff and Landing (VTOL) capability. The types of payloads attached to multirotors can usually be categorised as either a sensor (e.g. cameras and meteorological sensors), or freight (e.g. mail parcels or fire extinguishing material) [12]. Furthermore, the payload attachment is mainly categorised as either a rigid connection or a suspended connection [12]. In rare cases, a robotic actuator is attached to the multirotor to manipulate a payload [16, 17]. Figure 2.1 shows practical implementations of these three payload configurations.

(a) Rigid connection [18] **(b)** Suspended cable [19] **(c)** Robotic actuator [20]

Figure 2.1: Different multirotor-payload configurations

2.1.1. Rigid connection payloads

Payloads are often rigidly attached to a multirotor for transportation. This configuration is especially popular for commercial package deliveries [21]. In this use case, the mass and size of the payload are often unknown before a flight. There is minimal relative movement between the multirotor and the rigidly connected payload. Therefore, the payload only affects the Centre-of-Mass (CoM), the moment of inertia, and the aerodynamics of the vehicle.

Different control approaches have been proposed to deal with the altered flight dynamics in this applications, including Adaptive Robust Control (ARC) [22] and Model Reference Adaptive Control (MRAC) [23]. These control architectures mostly involve a parameter estimation algorithm to estimate the inertial parameters, and an adaptive control law based on the estimated parameters and the predetermined dynamical model of the system.

An advantage of rigidly connected payloads is that the flight dynamics is not altered significantly. The payload does not add a degree of freedom to the system and only the inertial parameters need to be accounted for. However, this configuration limits the shape and size of a potential payload, because the payload needs to be compatible with the vehicle gripper. The multirotor also needs to land or approach the payload closely to attach to the payload, which may be impractical in many applications.

2.1.2. Suspended payloads

Figure 2.2 shows an example of a practical application of a suspended payload used during search and rescue missions. The shape and mass of the payload affect the flight dynamics, but the payload parameters are often unknown before a flight. The control system should be able to account for these uncertainties and fly well despite the altered flight dynamics.

Various suspended payload configurations have been considered in literature. The classical suspended payload application involves a small payload suspended below the vehicle with a rigid link [5, 6, 24, 25, 26, 27]. Kotaru et al. [28] considered a suspended payload system with an elastic cable modelled as a spring-damper system. Tang and Kumar [29] modelled the multirotor-payload system with a hybrid dynamical model to consider aggressive manoeuvres where the cable transitions from taut to slack. The transportation of payloads with flexible cables have also been studied, where the cable is modelled as a set of serially connected rigid links [30, 31, 32]. Furthermore, the control of a group of multirotors cooperatively transporting a suspended payload was also considered in various studies [30, 33, 34, 35].

Figure 2.2: A practical suspended payload used for search and rescue missions [1]

From numerous examples in literature, it is clear that the control of multirotors with suspended payloads is a popular research topic. The suspended payload configuration is very useful in situations where a multirotor cannot land since the payload can be attached during hover. This configuration also has the advantage that a payload can have an arbitrary shape or size as long as it has an attachment point for a cable. However, the suspended payload increases the degrees of under-actuation of the system, which makes the control problem challenging [32].

2.2. Control of multirotors with suspended payloads

A major drawback of transporting a suspended payload is that the payload is free to swing during flight, which affects the dynamics of the multirotor. Two main control strategies are applied in the literature to stabilise a multirotor with a suspended payload, namely, trajectory generation and swing damping control. Some methods combine the two methods into a single control architecture. Trajectory generation methods involve determining multirotor trajectories that result in minimal oscillations or specific payload trajectories. Swing damping control involves feedback controllers that apply a control law to actively counteract the swing of a payload.

2.2.1. Trajectory generation

Trajectory generation methods for suspended payload systems are based on open-loop control techniques. The objective of these techniques is to determine a trajectory in which the multirotor motion would induce a specific payload trajectory to reduce oscillations or avoid obstacles. Numerous trajectory generation methods have been explored in the literature for suspended payload transportation [29, 36, 37, 38, 39, 40, 41, 42, 43].

Zeng et al. [37] and Tang and Kumar [29] applied differential flatness based trajectory planning methods for multirotors in obstacle-filled environments. Instead of only considering swing reduction of the suspended payloads, these studies consider specific payload trajectories to avoid obstacles during aggressive motion. Xian et al. [38] proposed an efficient online trajectory planning method without iterative optimizations. The swing-reduction performance of this method was verified with experimental results.

Dynamic programming methods have also been implemented to generate swing-free trajectories with suspended payloads [39, 40, 41]. These methods require accurate models of the plant dynamics and are sensitive to the accuracy of these models. Reinforcement Learning (RL) methods do not require prior models of the dynamics and have also been applied for swing-free trajectory generation [42, 43]. Faust et al. [43] implemented a RL method for minimal swing trajectories which provides sufficient criteria to allow the learned policy to be transferred to a variety of different models, starting positions, and trajectories. Furthermore, this RL trajectory generation method was verified with experimental results.

Input shaping is another open-loop control method applied for minimal swing control that is related to trajectory planning. This technique involves modifying a reference signal, usually with a set of timed impulses, to cancel the oscillatory modes of a system [44]. These techniques were originally designed for transporting suspended payloads using gantry systems [45, 46]. Later, these input shaping techniques were also applied for reduced swing control of helicopters [47, 48] and multirotors [49, 50, 51] that carry suspended payloads.

Ichikawa et al. [26] compared different input shaping techniques for velocity control of a multirotor with a suspended payload in simulations. The specific input shaping techniques considered were: Zero Vibration (ZV), Negative Zero Vibration (NZV), Extra Insensitive (EI), and 2-hump EI. These methods convolve a baseline input command with precisely timed impulses based on the length of the suspended cable. Simulation results showed that the input shapers significantly decreased the residual payload oscillations compared to a baseline velocity controller. The study highlighted that EI and 2-hump EI were more robust to cable length uncertainty than ZV and NZV.

Slabber and Jordaan [6] considered a system with unknown payload parameters and applied a notch filter to reduce payload oscillations for velocity control of a multirotor in simulation. The unknown payload mass and cable length were estimated with Recursive Least Squares (RLS) and Fast Fourier Transform (FFT) parameter estimators respectively and the natural frequency was calculated based on these estimates. The notch filter could then be designed to suppress the frequency band containing this natural frequency and was applied to the velocity setpoint signal. It was shown in simulation that the notch filter

attenuated the payload oscillations to a near swing-free motion despite large parameter estimation errors [6].

2.2.2. Swing damping controllers

Swing damping control is a closed-loop method where a feedback controller is applied to reduce the payload swing angles during a flight. This control method is also referred to as active vibration damping. Instead of finding a trajectory that reduces oscillations, these controllers follow a given trajectory as close as possible while trying to reduce the payload oscillations.

Linear Quadratic Regulator (LQR) is a popular optimal control technique and has often been used as a baseline controller to evaluate the performance of other swing damping controllers [6, 52, 53, 54, 55, 56]. Erasmus and Jordaan [57] proposed a Linear Quadratic Gaussian (LQG) controller for swing damping control of a multirotor with suspended cable. The payload state remained unmeasured and a Extended Kalman Filter (EKF) was implemented for full-state estimation of the multirotor-payload system. The EKF was combined with an LQR full-state feedback controller to produce LQG control. Simulation results showed good swing damping control with position step inputs despite the unmeasured payload state, external disturbances, sensor noise, and parameter uncertainty.

Slabber and Jordaan [6] implemented a LQR controller augmented with a notch filter input shaper for improved swing damping performance. The notch filter was applied to the velocity step reference and the LQR was then applied with the filtered reference signal for swing-damping control. The LQR was designed with integral action added to the velocity state to ensure zero steady-state velocity tracking. Furthermore, this work involved estimating the unknown payload state with a vision-based estimator for use in the full-state feedback controller. Simulation results showed that this controller provided good swing damping performance in the presence of external disturbances, sensor noise, and parameter uncertainty.

Model Predictive Control (MPC) is an optimal control technique related to LQR and can also be applied to suspended payload systems. Notter et al. [56] implemented an MPC for active swing damping control of a multirotor with a suspended payload. A non-linear model of the multirotor-payload system was linearised and discretised to apply a discrete, linear MPC formulation. The physical parameters of the multirotor, cable, and payload were assumed to be exactly known and the controller was tested with only one payload. The controller received a position trajectory reference and determined force setpoints to control the vehicle. Furthermore, constraints were applied to the heigh, attitude, and control inputs to ensure safe flight manoeuvres. Simulation results showed a superior

trajectory tracking performance with the MPC compared to a baseline LQR controller. The MPC simulation results were also verified with experimental results in an indoor environment.

Santos et al. [58] implemented a robust tube-based MPC for trajectory tracking and payload stabilisation of a tilt-rotor UAV and suspended payload. This approach consists of a pre-stabilising control policy for the nominal system and an additive control policy for the mismatch error. The MPC was applied as an outer-loop position controller and a mixed $\mathcal{H}_2/\mathcal{H}_\infty$ controller was applied for inner-loop attitude control. Integral action is applied in the MPC to the position states to ensure zero steady-state error despite external disturbances and modelling errors.

The tube-based MPC was designed to be robust against the additive uncertainties from the decoupling, linearization and discretisation modelling errors. However, the physical parameters of the model were assumed to be exactly known and non-additive parameter uncertainty was not considered. Simulation results showed successful stabilised control of the system along a square-like trajectory with sharp corners.

In other studies, more types of controllers were applied for active swing damping and these studies will be discussed in Section 2.3. Swing damping controllers generally perform better than open-loop, trajectory generation methods for systems with model uncertainties and external disturbances [59]. This is expected since trajectory generation requires accurate plant models and small modelling uncertainties can significantly alter a trajectory. The remainder of this study will focus on swing damping controllers.

Table 2.1: Summary of literature considered regarding the swing damping control of multirotors with suspended payloads

Author	Year	Proposed controller	Baseline controller	Plant model	Parameter uncertainty	Unknown dynamics	Different payloads	Practical data	Outdoor experiments
Muthusamy et al. [4]	2021	BFBEL	PID	-	✓	✓		✓	
Allahverdy et al. [60]	2021	BISMC with ILC	-	non-linear	✓	✓			
Faust et al. [61]	2014	RL (CAFVI)	-	-	✓	✓		✓	
Wang et al. [3]	2020	ADRC	PID	linear	✓	✓		✓	✓
Hua et al. [62]	2021	RL (non-linear)	BS, EB	non-linear	✓			✓	
Taylor et al. [63]	2020	$\mathcal{H}_\infty$ loop-shaping	LQR	linear	✓		✓		
Erasmus and Jordaan [5]	2020	MRAC	LQR	linear	✓		✓	✓	✓
Slabber and Jordaan [6]	2020	NF with LQR	PID	linear	✓		✓		
Dai et al. [64]	2014	RCAC	-	non-linear	✓				
Santos and Raffo [65]	2016	MPC	-	linear	✓				
Andrade et al. [66]	2016	MPC	LQR	linear	✓				
Zurn et al. [67]	2016	MPC	-	linear				✓	
Son et al. [68]	2019	MPC	-	linear				✓	
Son et al. [69]	2018	MPC	-	linear				✓	
Son et al. [70]	2017	MPC	-	linear					
Trachte et al. [52]	2014	MPC	LQR	non-linear					
Trachte et al. [71]	2015	MPC	LQR	non-linear					
Liang et al. [59]	2021	Non-linear	LQR, PD	non-linear			✓	✓	
Zeng and Sreenath [72]	2019	Geometric	-	non-linear					
Yang and Xian [73]	2018	RISE	-	non-linear					
Martinez-Vasquez et al. [74]	2020	SMC	-	linear					
Mosco-Luciano et al. [75]	2020	BS	-	linear				✓	
Rigatos et al. [76]	2018	$\mathcal{H}_\infty$	-	linear					
Alothman et al. [77]	2015	LQR	PD	linear					
Alothman and Gu [55]	2016	iLQR	LQR	linear					

Refer to Page (xvii) for the abbreviations used in this table.

2.3. Review of swing damping control studies

This section will discuss trends in the literature regarding the swing damping control of multirotors with suspended payloads. Table 2.1 lists other studies in the literature that consider this topic. The entries of this table are ordered to keep similar studies together, with a priority on unknown dynamics, parameter uncertainty, and proposed controllers. Studies that exclusively consider trajectory generation or cooperative transportation of a payload with multiple multirotors are excluded from this table. From Table 2.1, it is clear that suspended payload transportation with multirotors is a popular and current research topic.

For each study in Table 2.1, the type of proposed controller is listed along with a baseline controller if applicable. Baseline controllers are techniques considered to be well-known that are applied to a task for a reference performance used to evaluate a proposed technique. Other studies compare variations of the proposed controller to each other to highlight the effect of design decisions, but these comparisons are not considered as baseline comparisons.

Many studies in Table 2.1 do not consider a baseline controller, which makes it difficult to evaluate the performance of the proposed technique objectively. These studies can conclude that the proposed technique solves the considered problem, but can not determine whether the technique improves on the performance of known controllers. However, from Table 2.1 it is clear that LQR is a popular baseline controller for swing damping techniques and especially for optimal control techniques.

From the studies considered in Table 2.1, it also appears that MPC is a popular technique for the considered control problem. Historically, MPC was designed for slow-moving processes in the chemical industry because the computational intensity of this method limited the controller frequency on the available hardware [78]. However, due to improvements in the speed of computational hardware, MPC has become a viable controller for faster systems. Various MPC implementations were successfully applied in experimental flight tests which shows that MPC is suitable for practical multirotor implementations [67, 68, 69].

It is also noted that most proposed controllers, including MPC implementations, are based on a linearised plant model of the non-linear multirotor-payload dynamics. This shows that a linearised plant model can provide a sufficient representation of the suspended payload dynamics for effective swing damping control. Non-linear MPC implementations that depend on a non-linear plant model have also been studied for multirotors with suspended payloads [52, 71]. However, the non-linear MPC results were not compared to linear MPC results, therefore the studies do not conclusively justify the need for a

non-linear plant model. Non-linear MPC is more computationally intensive than linear MPC, which makes it more challenging to implement in practical flights. None of the studies considered in Table 2.1 that were implemented on practical systems used non-linear plant models. However, some practical implementations apply controllers which do not depend on a plant model [4, 61, 62].

Note from Table 2.1, that many studies only consider the proposed controllers in simulations and do not include practical data. Practical data may include sensor noise, modelling uncertainties, external disturbances, and other computational hardware effects such as latency that are often not considered in simulations. Unlike simulation results, experimental results clearly show that a proposed method is suitable for real-life applications. It also shows that the proposed algorithms can run in real-time on the available hardware, which is a challenge for complex techniques.

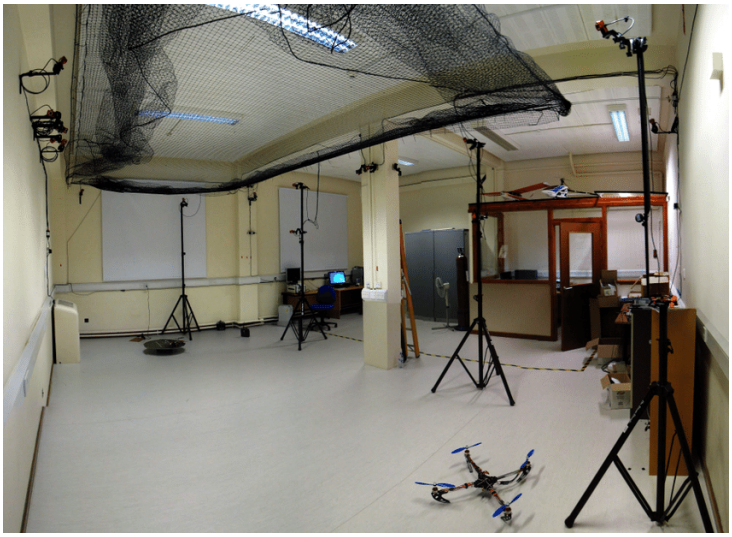


Figure 2.3: Optitrack motion capture system for multirotor experiments [2]

It is also noted that few studies in the literature consider outdoor flights. Most studies only consider practical flights performed in controlled indoor environments. Figure 2.3 shows an indoor motion capture setup used for multirotor experiments. In these experiments, motion capture systems like Vicon [4, 61, 62, 67, 68, 69], Qualisys [59] or Optitrack [75] provide high accuracy state feedback data. However, this setup is often impractical for real-life multirotor applications.

Outdoor payload transportation is dependant on inaccurate sensors like Global Positioning System (GPS) and potentiometers, which greatly increase the difficulty of the control problem. The multirotor-payload system may also be exposed to uncontrolled wind disturbances which further complicates the control problem. Figure 2.4 shows a multirotor in an outdoor practical experiment with a suspended payload [3].

Figure 2.4: Multirotor and suspended payload for outdoor experiments [3]

As observed by Hua et al. [62], most reported controllers in the literature are designed based on accurate plant models without considering dynamical uncertainties in the studies. This is also evident from the literature listed in Table 2.1. The *Parameter uncertainty* column identifies studies that account for parameter uncertainty in the considered plant model. These controllers either apply robust techniques [63] to ensure stability despite the parameter uncertainty, or adaptive techniques [64] to change the control law to result in improved control with the resultant dynamics. Other controllers combine robust and adaptive techniques into a single control architecture [5, 6].

The *Different payloads* column identifies studies that consider more than one payload. It is interesting to note that few studies test the proposed controllers on more than one payload. Some studies proposed controllers to account for parameter uncertainty, but only tested the controller on a single payload case. This does not conclusively demonstrate the adaptability or robustness of a controller, because the payload could be cherry-picked or the controller could be specifically tuned for that payload only. Therefore, it is noted that it is valuable to demonstrate a controller on multiple payload cases.

In Table 2.1, the *Unknown dynamics* column identifies studies that account for unknown dynamics of a multirotor with a suspended payload. This does not include the uncertainty due to modelling errors as a result of linearisation and discretisation methods. Studies identified by this column propose stabilising controllers that are not design without a priori knowledge of the payload dynamics. This approach is considered useful for complicated working conditions and model uncertainties [62]. This appears to be a promising research

area that has been considered in only a few different studies. It is also interesting to note that these studies are quite recently published.

2.4. Multirotor and suspended payload systems with unknown dynamics

Only a few studies have been identified that consider the stabilised control of a multirotor and suspended payload without prior knowledge of the payload dynamics [3, 4, 60, 61]. Some methods are not based on a plant model and learn a stabilising control law without prior knowledge of the dynamics [4, 61]. Other strategies control the multirotor with a model-based method and consider the effect of the suspended payload as an external disturbance [3, 60]

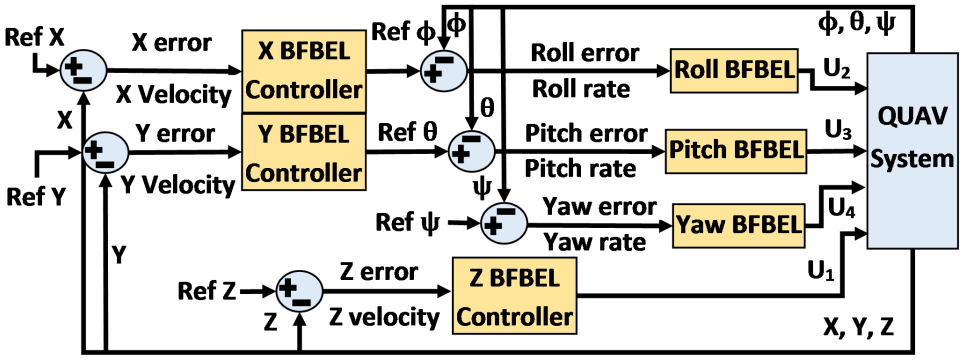


Figure 2.5: Controller structure proposed by Muthusamy et al. [4]

Muthusamy et al. [4] proposed a Bidirectional Fuzzy Brain Emotional Learning (BFBEL) controller which incorporated fuzzy inference, neural networks and the Bidirectional Brain Emotional Learning (BBEL) algorithm. A separate BFBEL feedback controller was applied to each degree of freedom of the Six-Degrees-of-Freedom (6DOF) multirotor system, as shown in Figure 2.5. The payload state remained unmeasured and the objective of the controller was to stabilise the multirotor system and provide accurate position tracking without knowledge of the multirotor-payload dynamics. Experimental results demonstrated the rapid adaptation capability and the trajectory tracking performance of the proposed BFBEL controller. Without prior knowledge of the system, the controller weights were autonomously tuned within 30 s of flight time to provide stable trajectory tracking with the multirotor. However, the payload state was not explicitly measured or damped, causing residual oscillations in the position data of the multirotor.

An Active Disturbance Rejection Control (ADRC) was proposed by Wang et al. [3] for the control of a multirotor with an unknown suspended payload. A transfer function model of

a practical multirotor without a payload was determined with a frequency sweep excitation method of each control channel. The effect of the suspended payload was considered as an external disturbance and Extended State Observers (ESOs) were applied to estimate the disturbance. An ADRC could then actively reject the disturbance caused by the payload and stabilise the system without prior knowledge of the payload dynamics. Experimental results showed that trajectory tracking was significantly improved compared to a standard Proportional Integral Derivative (PID) controller. It should be noted that this technique did not show significant swing damping performance but rather showed robustness against the disturbance effect of the swinging payload.

These methods provide stabilised control of the multirotor, but do not provide optimised control of the entire multirotor-payload system. These controllers focus on counteracting the current swinging payload disturbance but does not learn how to directly control the unknown payload.

System identification methods can determine a model of the unknown dynamics and a controller could be designed based on the identified plant model. This could provide improved control of the entire dynamical system. Control architectures involving data-driven model identification and resultant model-based controller have been proposed for multirotors [3, 79]. However, we were not able to find similar studies in the literature that involves an unknown suspended payload.

2.5. Summary

This chapter reviewed a range of different control solutions for multirotors with suspended payloads. A few observations were made regarding the literature on this subject. The main observations are:

1. Most of the studied controllers are based on accurate models of the system dynamics.
2. Swing damping methods perform better than trajectory generation methods when considering model uncertainty.
3. Many studies consider some parameter uncertainty, but only a few studies consider unknown system dynamics.
4. Only a few studies that account for parameter uncertainty are also demonstrated with different payload parameters.
5. Methods that consider unknown system dynamics usually consider the suspended payload as an unknown disturbance and do not attempt to actively control the payload.

6. No studies were found in the literature that consider data-driven system identification and control of multirotors with suspended payloads.

The focus of this thesis will involve the stabilised control of a multirotor with an unknown suspended payload. The dynamics of the suspended payload system will be considered unknown before a flight. Furthermore, the proposed controller will be based on an estimated model of the multirotor-payload dynamics and it will be tested on different payloads.

Chapter 3

Modelling

In this chapter, a non-linear mathematical model of a multirotor with a suspended payload will be derived. The model will be based on the multirotor named *Honeybee*, which was built by Grobler and Jordaan [10].

The chapter starts by defining the coordinate frames and types of rotations used in this work. After a discussion of the different forces and moments acting on this vehicle, a 6DOF model of the multirotor vehicle will be derived. The derivation of the suspended payload model will also be shown. Finally, practical flight data will be used to verify the simulated mathematical model.

3.1. Coordinate frames

Honeybee is a quadrotor vehicle, which is a type of multirotor with four propellers. The quadrotor X-configuration is considered in this work. Figure 3.1 shows the quadrotor schematic and the two coordinate frames used to describe this system.

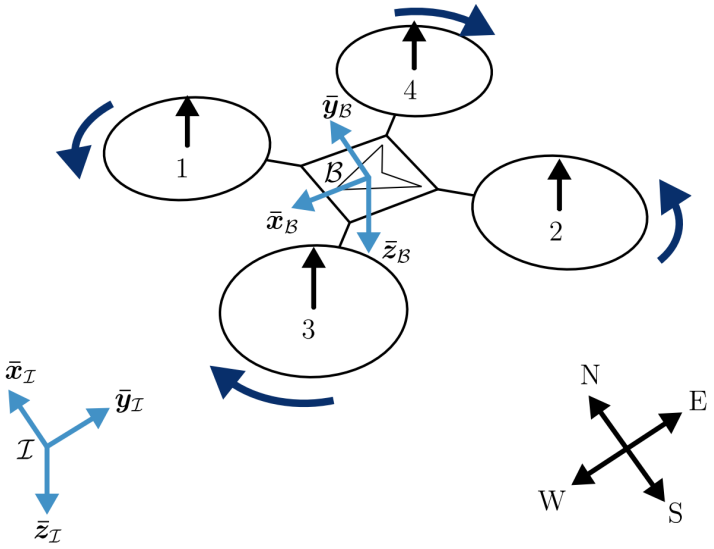


Figure 3.1: Inertial and body coordinate frames of a quadrotor [5]

The inertial frame is denoted by $\mathcal{I} = \{\bar{\boldsymbol{x}}_{\mathcal{I}}, \bar{\boldsymbol{y}}_{\mathcal{I}}, \bar{\boldsymbol{z}}_{\mathcal{I}}\}$ and describes a North-East-Down (NED) axis system. The x, y, and z axis, align with the North, East, and Down inertial directions respectively. The inertial frame assumes a flat, non-rotating earth since the multirotor travels small distances in comparison to the curvature of the earth. The origin of this frame is fixed at the takeoff location of the multirotor.

The body frame is denote by $\mathcal{B} = \{\bar{\boldsymbol{x}}_{\mathcal{B}}, \bar{\boldsymbol{y}}_{\mathcal{B}}, \bar{\boldsymbol{z}}_{\mathcal{B}}\}$ and is fixed to the multirotor body. The origin of this frame is at the CoM of the vehicle and the x, y, and z axes, align with the forwards, rightwards, and downwards directions of the multirotor body respectively. The body frame is defined by a translation and rotation relative to the inertial frame. The mathematical representation of rotations is be discussed in the section below.

3.2. Rotations

The rotation of the body frame relative to the inertial frame is referred to as the attitude of the multirotor. In this work, the attitude is defined in terms of Euler angles or quaternions.

3.2.1. Euler angles

Euler angles describe a Three-Dimensional (3D) rotation as a sequence of three consecutive elementary rotations [80]. The ZYX-sequence is commonly used for aircraft applications [80] and is also used in this work. The order of rotations are:

1. Rotate the body frame about the z-axis by the yaw angle, Ψ.
2. Rotate the resulting frame about the new y-axis by the pitch angle, Θ.
3. Rotate the resulting frame about the new x-axis by the roll angle, Φ.

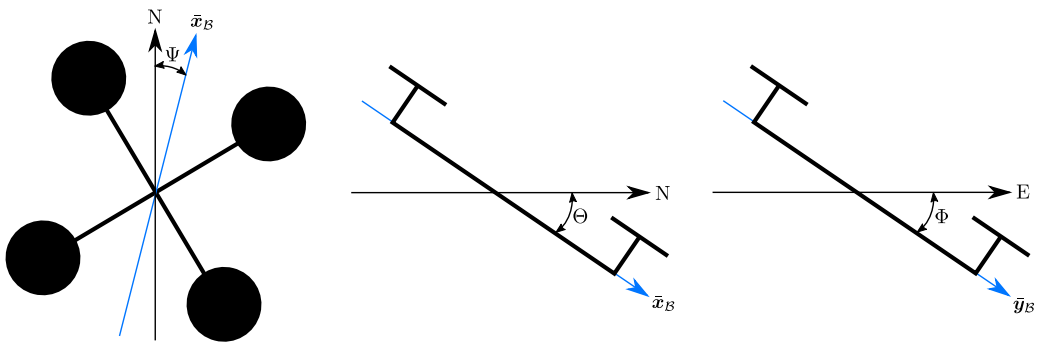


Figure 3.2: Illustration of Euler angles [6]

Figure 3.2 gives a simple illustration of the Euler-ZYX angles. For simplicity in the illustration, Θ and Φ are each shown as a pure pitch and roll angle, without a prior Euler rotation.

3.2.2. Quaternions

Quaternions provide a way of representing a rotation with four parameters. An advantage of quaternions is that it does not have mathematical singularities as Euler angles do [80]. A quaternion defines a 3D rotation as a single rotation about a fixed axis. This is parametrized by a rotation angle, α, and a unit vector, $\mathbf{r}$.

A unit quaternion is defined as,

$$\boldsymbol{q} = [q_0 \quad q_1 \quad q_2 \quad q_3]^T = \begin{bmatrix} q_0 \\ \boldsymbol{q}_v \end{bmatrix} = \begin{bmatrix} \cos(\frac{\alpha}{2}) \\ \mathbf{r}\sin(\frac{\alpha}{2}) \end{bmatrix}, \tag{3.1}$$

where q_0 is the magnitude component and $\boldsymbol{q}_v$ is the vector component of the quaternion.

A Euler-ZYX angle representation, $[\Theta \quad \Phi \quad \Psi]$, can be converted to a quaternion using [80],

$$\boldsymbol{q}(\Theta, \Phi, \Psi) = \begin{bmatrix} \cos\frac{\Phi}{2}\cos\frac{\Theta}{2}\cos\frac{\Psi}{2} & + & \sin\frac{\Phi}{2}\sin\frac{\Theta}{2}\sin\frac{\Psi}{2} \\ -\cos\frac{\Phi}{2}\sin\frac{\Theta}{2}\sin\frac{\Psi}{2} & + & \cos\frac{\Theta}{2}\cos\frac{\Psi}{2}\sin\frac{\Phi}{2} \\ \cos\frac{\Phi}{2}\cos\frac{\Psi}{2}\sin\frac{\Theta}{2} & + & \sin\frac{\Phi}{2}\cos\frac{\Psi}{2}\sin\frac{\Psi}{2} \\ \cos\frac{\Phi}{2}\cos\frac{\Theta}{2}\sin\frac{\Psi}{2} & - & \sin\frac{\Phi}{2}\cos\frac{\Psi}{2}\sin\frac{\Theta}{2} \end{bmatrix}. \tag{3.2}$$

The inverse of a quaternion is defined as,

$$\boldsymbol{q}^{-1} = \frac{\begin{bmatrix} q_0 \\ -\boldsymbol{q}_v \end{bmatrix}}{\sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2}}. \tag{3.3}$$

Furthermore, the multiplication of two quaternions, $\boldsymbol{q}$ and $\boldsymbol{q}'$, is given by [80]:

$$\boldsymbol{q} \cdot \boldsymbol{q}' = Q(\boldsymbol{q})\boldsymbol{q}', \tag{3.4}$$

where

$$Q(\boldsymbol{q}) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix}. \tag{3.5}$$

Successive rotations can therefore be represented mathematically by quaternion multiplication. These equations will be referred to and used in subsequent chapters.

3.3. Multirotor model

The multirotor is modelled as a 6DOF rigid body. This includes three translational and three rotational degrees of freedom. This modelling process is well described by [5] and [6], and the same general procedure is used here.

The system parameters describing the physical properties of the multirotor are listed in Table 3.1.

Table 3.1: System parameters of the multirotor model.

Symbol	Description
m_Q	Mass of multirotor
d	Distance from CoM to each motor
R_N	Virtual yaw moment arm
τ	Motor-propeller pair time constant
I_{xx}	Mass moment of inertia about $\bar{\boldsymbol{x}}_{\mathcal{B}}$
I_{yy}	Mass moment of inertia about $\bar{\boldsymbol{y}}_{\mathcal{B}}$
I_{zz}	Mass moment of inertia about $\bar{\boldsymbol{z}}_{\mathcal{B}}$
C_{Q_X}	Aerodynamic drag coefficient in $\bar{\boldsymbol{x}}_{\mathcal{B}}$ direction
C_{Q_Y}	Aerodynamic drag coefficient in $\bar{\boldsymbol{x}}_{\mathcal{B}}$ direction
C_{Q_Z}	Aerodynamic drag coefficient in $\bar{\boldsymbol{x}}_{\mathcal{B}}$ direction

The inertia tensor of the multirotor is defined as,

$$\boldsymbol{I_Q} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \approx \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \tag{3.6}$$

The multirotor is assumed to be symmetrical about the XZ- and YZ-plane, therefore the inertia tensor can be approximated as a diagonal matrix as shown in Equation 3.6.

The linear velocity and angular velocity of the multirotor within the body frame is denoted by,

$$\boldsymbol{V_{\mathcal{B}}} = \begin{bmatrix} V_{\mathcal{B}_X} & V_{\mathcal{B}_Y} & V_{\mathcal{B}_Z} \end{bmatrix}^T, \text{ and} \tag{3.7}$$

$$\boldsymbol{\Omega_{\mathcal{B}}} = \begin{bmatrix} \Omega_{\mathcal{B}_X} & \Omega_{\mathcal{B}_Y} & \Omega_{\mathcal{B}_Z} \end{bmatrix}^T. \tag{3.8}$$

Furthermore, the sum of forces and sum of moments acting on the multirotor in the body frame are denotes by,

$$\boldsymbol{F_B} = \begin{bmatrix} F_{B_X} & F_{B_Y} & F_{B_Z} \end{bmatrix}^T, \text{ and} \tag{3.9}$$

$$\boldsymbol{M_B} = \begin{bmatrix} M_{B_X} & M_{B_Y} & M_{B_Z} \end{bmatrix}^T. \tag{3.10}$$

As described by [5], these equations can be used with Newton's second law to derive the rigid body equations of motion as,

$$\boldsymbol{F_B} = m_Q \dot{\boldsymbol{V}}_{\boldsymbol{B}} + \boldsymbol{\Omega_B} \times m_Q \boldsymbol{V_B}, \tag{3.11}$$

$$\boldsymbol{M_B} = \boldsymbol{I_Q} \dot{\boldsymbol{\Omega}}_{\boldsymbol{B}} + \boldsymbol{\Omega_B} \times \boldsymbol{I_Q} \boldsymbol{V_B}, \tag{3.12}$$

This provides a set of Ordinary Differential Equations (ODEs) which fully describe the multirotor motion in 6DOF, given the forces and moments acting on the vehicle. With the equation derived in [81], the attitude of the multirotor can be obtained as a quaternion from the body angular rates using,

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \Omega_{B_X} \\ \Omega_{B_Y} \\ \Omega_{B_Z} \end{bmatrix}. \tag{3.13}$$

The Direct Cosine Matrix (DCM) is also derived in [81] and can be calculated from the attitude quaternion as,

$$\boldsymbol{R}_V = \begin{bmatrix} q_0^2 + q_1^2 + q_2^2 + q_3^2 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \tag{3.14}$$

$\boldsymbol{R}_V$ is the transformation matrix that describes the rotation from the body frame to the inertial frame such that,

$$\boldsymbol{V_I} = \boldsymbol{R}_V^{-1} \boldsymbol{V_B} \tag{3.15}$$

3.4. Forces and moments

Different phenomena apply forces and moments to the multirotor during flight. The total force and moment acting on the multirotor in the body frame are given by,

$$\boldsymbol{F}_{\mathcal{B}} = \boldsymbol{F}_{\mathcal{B}}^{T} + \boldsymbol{F}_{\mathcal{B}}^{A} + \boldsymbol{F}_{\mathcal{B}}^{G} + \boldsymbol{F}_{\mathcal{B}}^{P} \text{ and} \tag{3.16}$$

$$\boldsymbol{M}_{\mathcal{B}} = \boldsymbol{M}_{\mathcal{B}}^{T} + \boldsymbol{M}_{\mathcal{B}}^{A} + \boldsymbol{M}_{\mathcal{B}}^{G} + \boldsymbol{M}_{\mathcal{B}}^{P}. \tag{3.17}$$

The phenomena that cause the different forces and moments are denoted by the superscripts, T, A, G, and P, which refer to actuator thrust, aerodynamic drag, gravity, and the payload respectively. These phenomena are also considered and modelled by [5] and [6].

Actuator thrust

A multirotor has four rotors which each produce a thrust as shown in Figure 3.1. However, the actuators collectively apply a force, $\boldsymbol{F}_{\mathcal{B}}^{T}$, and moment, $\boldsymbol{M}_{\mathcal{B}}^{T}$, to the vehicle. This force and moment can be represented in terms of virtual actuator thrusts as,

$$\boldsymbol{F}_{\mathcal{B}}^{T} = \begin{bmatrix} 0 \\ 0 \\ \delta_T \end{bmatrix} \quad \text{and} \quad \boldsymbol{M}_{\mathcal{B}}^{T} = \begin{bmatrix} d \cdot \delta_A \\ d \cdot \delta_E \\ R_N \cdot \delta_R \end{bmatrix}, \tag{3.18}$$

where d is the distance from each motor to the multirotor CoM and R_N is the virtual yaw moment arm which is a property of the motor-propeller configuration. The virtual aileron, elevator, and rudder actuator thrusts are denoted by $d\delta_T$, $d\delta_A$, $d\delta_E$, and $d\delta_R$, respectively. These values are calculated with a mixing matrix, such that,

$$\begin{bmatrix} \delta_T \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}, \tag{3.19}$$

where $\boldsymbol{T} = [T_1 \;\; T_2 \;\; T_3 \;\; T_4]^T$ is a vector of actual thrust forces produced by the four individual rotors. Each motor-propeller pair receives a thrust setpoint and produces a corresponding thrust. This is modelled with a first order differential equation given by,

$$\dot{\boldsymbol{T}} = \frac{\boldsymbol{T}_{sp} - \boldsymbol{T}}{\tau}, \tag{3.20}$$

where $\boldsymbol{T}_{sp}$ is a vector of thrust setpoints corresponding to $\boldsymbol{T}$, and τ is the time constant of the motor-propeller configuration.

Aerodynamics

Multirotors experience aerodynamic forces due to the relative velocity of air over the vehicle. The aerodynamic model is based on work done by [82] and was also applied successfully by [5] and [6]. The model describes the aerodynamic forces as,

$$\boldsymbol{F}_{\mathcal{B}}^{A} = \frac{1}{2}\rho \boldsymbol{V}_{\mathcal{B}_w} |\boldsymbol{V}_{\mathcal{B}_w}| \boldsymbol{C}_Q \tag{3.21}$$

where ρ is the air density, $\boldsymbol{V}_{\mathcal{B}_w}$ is the relative velocity of air over the multirotor in the body frame, and $\boldsymbol{C}_Q = [C_{Q_X} \quad C_{Q_Y} \quad C_{Q_Z}]^T$ is the drag coefficients and reference areas lumped into a single damping coefficient per axis. The relative velocity, $\boldsymbol{V}_{\mathcal{B}_w}$, is calculated as

$$\boldsymbol{V}_{\mathcal{B}_w} = -\boldsymbol{V}_{\mathcal{B}} + \boldsymbol{R}_V \boldsymbol{V}_w, \tag{3.22}$$

where $\boldsymbol{V}_w$ is the wind velocity in the inertial frame. It is assumed that no moments are caused by aerodynamics, such that,

$$\boldsymbol{M}_{\mathcal{B}}^{A} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.23}$$

Gravity

Gravity applies a vertical force to the multirotor in the inertial Down axis. The inertial force is transformed into the body frame with the DCM. No moments are applied to the multirotor due to gravity. Therefore, the total force and moment acting on the multirotor due to gravity is,

$$\boldsymbol{F}_{\mathcal{B}}^{G} = \boldsymbol{R}_V \begin{bmatrix} 0 \\ 0 \\ m_Q g \end{bmatrix} \quad \text{and} \quad \boldsymbol{M}_{\mathcal{B}}^{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.24}$$

Suspended payload

The suspended payload applies a reaction force, $\boldsymbol{F}_{\mathcal{B}}^{P}$, and moment, $\boldsymbol{M}_{\mathcal{B}}^{P}$, to the multirotor which is dependant on the dynamics of the suspended payload. The payload and cable also experience gravity and aerodynamic forces which influence its motion. These forces and moments are considered in greater detail in the next section.

3.5. Suspended payload model

The dynamical model of the multirotor is now fully defined, except for the force and moment applied to the vehicle by the payload. The suspended payload can be modelled as a rigid body attached to the multirotor by a rigid link. Due to the acceleration of the

payload, the payload applies a force to the multirotor through the link. The suspended payload equations of motion are derived to define these forces and complete the dynamical model of the multirotor-payload system.

Figure 3.3 illustrates a multirotor with a suspended payload, where m_p is the mass of the payload, and l is the length of the suspended cable. The x-axis and y-axis Euler-ZYX angles in the inertial frame are denoted by θ and ϕ respectively. Furthermore, $\boldsymbol{r}_Q = [x_Q \;\; y_Q \;\; z_Q]^T$ defines the position of the multirotor CoM in the inertial frame. Likewise, $\boldsymbol{r}_p = [x_p \;\; y_p \;\; z_p]^T$ defines the position of the payload in the inertial frame.

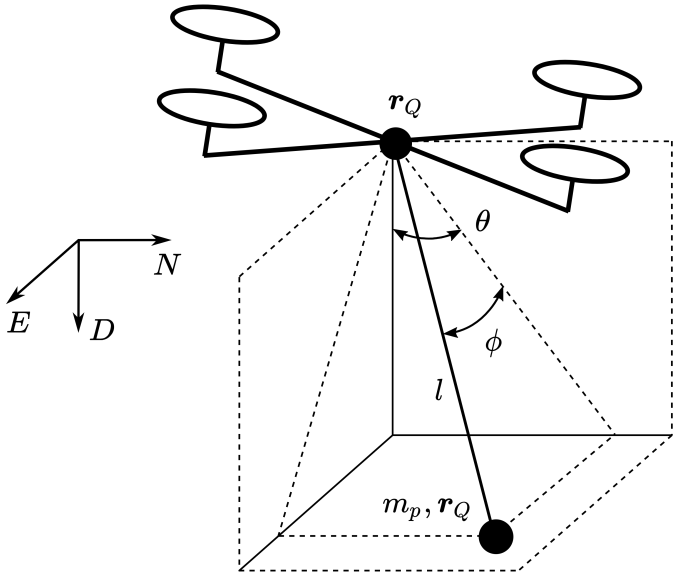


Figure 3.3: Schematic of a multirotor with suspended payload [6]

3.5.1. Payload assumptions

The following major assumptions are made regarding the payload model:

- The payload is a point-mass.
- The link is massless.
- The link is rigid.
- The link is attached to the CoM of the multirotor.

The payloads used in the practical setup (described in Chapter 6) are small relative to the multirotor and the attachment point is close to the payload CoM. The payloads are attached with a low-friction joint to the suspended cable and the rotation of the payload around the cable axis has a negligible effect on the multirotor. Therefore, modelling the payload as a point-mass appears to be a reasonable approximation.

The cable has a low mass in comparison to the payloads used and has a negligible amount of stretch. The cable remains straight and rigid during flight due to the tension applied by the payload. Aggressive manoeuvres could cause periods of zero cable tension where the load is in free-fall and the cable is slack [29]. However, such aggressive manoeuvres are not considered in this work and the assumption of a rigid, massless cable appears reasonable.

In the practical setup shown in Chapter 6, the cable attachment appears to be near to the CoM of the multirotor. It is assumed that even if the attachment point is slightly below the actual CoM, this has a negligible effect on the dynamics and can still be approximated by a CoM attachment. Due to this assumption, the payload cannot apply a moment to the multirotor, hence the multirotor attitude dynamics is decoupled from the payload dynamics. Therefore, the multirotor can be modelled as a point-mass when considering the payload equations of motion.

3.5.2. Lagrangian

Lagrangian mechanics is an energy-based modelling approach that can be used to derive differential equations to describe a system [5]. The dynamical equations of the suspended payload system were derived by [6] and the derivation in this work follows a similar approach. It is important to note that the derivation in this section considers the payload dynamics as a function of the motion of the multirotor CoM.

The payload position is defined as a function of the multirotor position in the inertial frame by the equation,

$$\boldsymbol{r}_p = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} x_Q + l\cos\phi\sin\theta \\ y_Q + \sin\phi \\ z_Q + l\cos\phi\cos\theta \end{bmatrix}. \tag{3.25}$$

The vector of generalised coordinates, $\boldsymbol{p}$, of the system can now be defined as,

$$\boldsymbol{p} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ \phi \\ \theta \end{bmatrix}. \tag{3.26}$$

The kinetic energy, $\mathcal{T}_p$, and potential energy, $\mathcal{V}_p$, of the payload can be determined as,

$$\mathcal{T}_p = \frac{1}{2} m_p |\dot{\boldsymbol{r}}_p|^2, \quad \text{and} \tag{3.27}$$

$$\mathcal{V}_p = -m_p g z_p, \tag{3.28}$$

where g is the acceleration due to gravity. Note that these energy equations describe the system modelled as two point-mass bodies and do not consider the attitude dynamics of the multirotor. The Lagrangian can now be determined as,

$$\mathcal{L} = \mathcal{T}_p - \mathcal{V}_p. \tag{3.29}$$

3.5.3. Non-conservative forces

Non-conservative forces and moments refer to effects that add or remove energy from the system. The non-conservative forces and moments acting on the payload include the aerodynamic drag force acting on the payload and the moment caused by the friction of the cable attachment. It is assumed that the cable is thin and short enough that the aerodynamic drag of the cable is negligible.

According to the aerodynamic model presented in Equation 3.21, the aerodynamic drag forces acting on the payload in the inertial frame are defined as,

$$\boldsymbol{F}_p^A = \begin{bmatrix} F_{p_x}^A \\ F_{p_y}^A \\ F_{p_z}^A \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\,\rho\,C_p\,\dot{x_p}^2 \\ \frac{1}{2}\,\rho\,C_p\,\dot{y_p}^2 \\ \frac{1}{2}\,\rho\,C_p\,\dot{z_p}^2 \end{bmatrix}, \tag{3.30}$$

where C_p is the lumped aerodynamic drag coefficient and reference area of the payload. It is assumed that the lumped payload drag coefficients are equal in all three axes, hence it is described by a single coefficient.

Friction at the cable attachment is modelled as linear damping. It is assumed that the coefficient of friction is equal in each axis of rotation. Therefore, the friction moments opposing the θ and ϕ rotations are given by,

$$M_\theta^F = -c\dot{\theta} \tag{3.31}$$

$$M_\phi^F = -c\dot{\phi} \tag{3.32}$$

respectively, where c is the rotational friction coefficient for θ and ϕ rotations. A vector, $\boldsymbol{Q}$, of non-conservative forces and moments corresponding to the system coordinates in $\boldsymbol{p}$ can now be defined as,

$$\boldsymbol{Q} = \begin{bmatrix} -F_{p_x}^A \\ -F_{p_y}^A \\ -F_{p_z}^A \\ M_\theta^F - F_{p_y}^A \cos\phi \\ M_\phi^F - F_{p_x}^A \cos\theta \end{bmatrix}. \tag{3.33}$$

3.5.4. Equations of motion

The set of Euler-Lagrange equations for this system are described by,

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{p}_j}\right) - \frac{\partial \mathcal{L}}{\partial p_j} = Q_j, \tag{3.34}$$

where p_j is an element in $\boldsymbol{p}$, Q_j is the corresponding element in $\boldsymbol{Q}$, and $j = \{1,\ 2,\ 3,\ 4,\ 5\}$. This set of coupled equations was solved with the Symbolic Maths Toolbox™ [83] in MATLAB® to determine the payload equations of motion. This yields a set of ODEs in the form,

$$\ddot{\boldsymbol{p}} = f(\dot{\boldsymbol{p}}, \boldsymbol{p}, \ddot{\boldsymbol{r}}_Q, \dot{\boldsymbol{r}}_Q), \tag{3.35}$$

which describes the motion of the payload as a function of the motion of the multirotor CoM in the inertial frame.

3.5.5. Payload forces acting on the multirotor

The reaction force applied to the multirotor due to the payload can now be determined in the inertial frame as,

$$\boldsymbol{F}_{\mathcal{I}}^{P} = m_Q \ddot{\boldsymbol{r}}_Q^{P}, \tag{3.36}$$

from Newton's second law, where $\ddot{\boldsymbol{r}}_Q^{P}$ represents the component of the multirotor acceleration caused by the suspended payload. This force can be represented in the body frame as,

$$\boldsymbol{F}_{\mathcal{B}}^{P} = \boldsymbol{R}_V \boldsymbol{F}_{\mathcal{I}}^{P}. \tag{3.37}$$

The set of ODEs represented by Equation 3.35 provides a coupling between the multirotor and payload dynamics. The multirotor-payload system can now be simulated from a given initial condition with an ODE solver in Simulink™.

3.6. Model verification

To verify whether the non-linear simulation model derived in this chapter is an accurate representation of the real-world dynamics, the simulation data is verified against practical data. A North velocity step response was performed in a practical flight with a multirotor to verify the multirotor model without a payload. This multirotor will be described in Chapter 6.

This flight was performed on a day with almost no wind to minimise the effect of wind disturbances in the comparison. Starting from the same initial condition, a velocity step setpoint was commanded in a simulation and the resulting simulation data was compared

to the practical flight data. The results compare the combined multirotor and controller simulation model to the practical flight data. The Simulink™ simulation environment used here includes the dynamics of the default multirotor controllers and will be further discussed in Chapter 5.

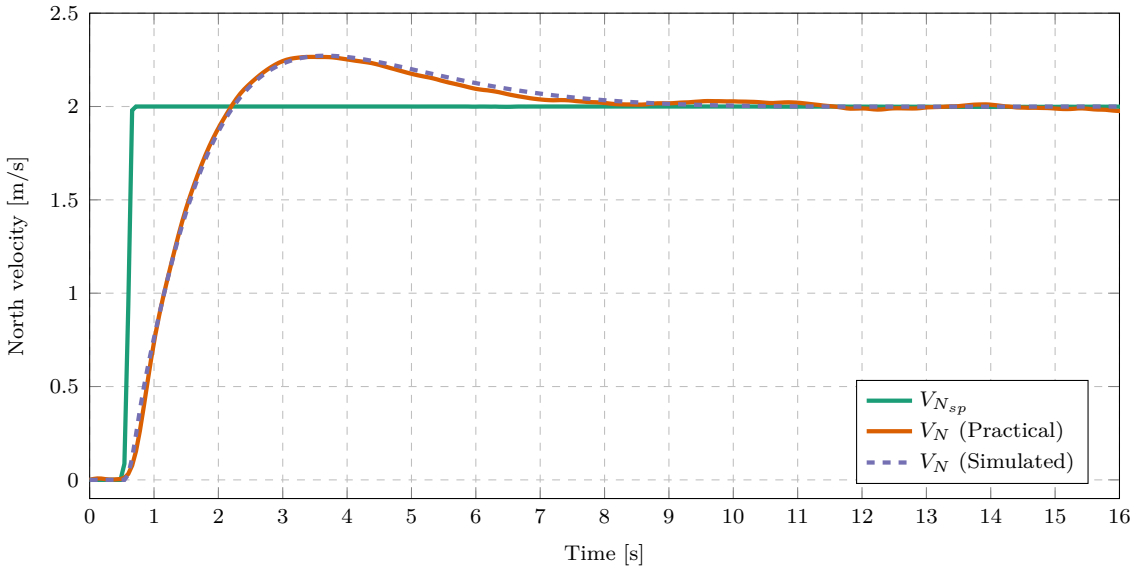


Figure 3.4: Comparison of simulated and practical data from Honeybee.

Figure 3.4 shows the velocity step responses of a practical and a simulated flight. From a visual inspection of the plots, it is clear that the velocity responses match well. Note that the practical velocity curve is slightly more irregular and not as smooth as the simulated curve. This may be due to slight wind disturbances or sensor noise, which are not considered in the simulation.

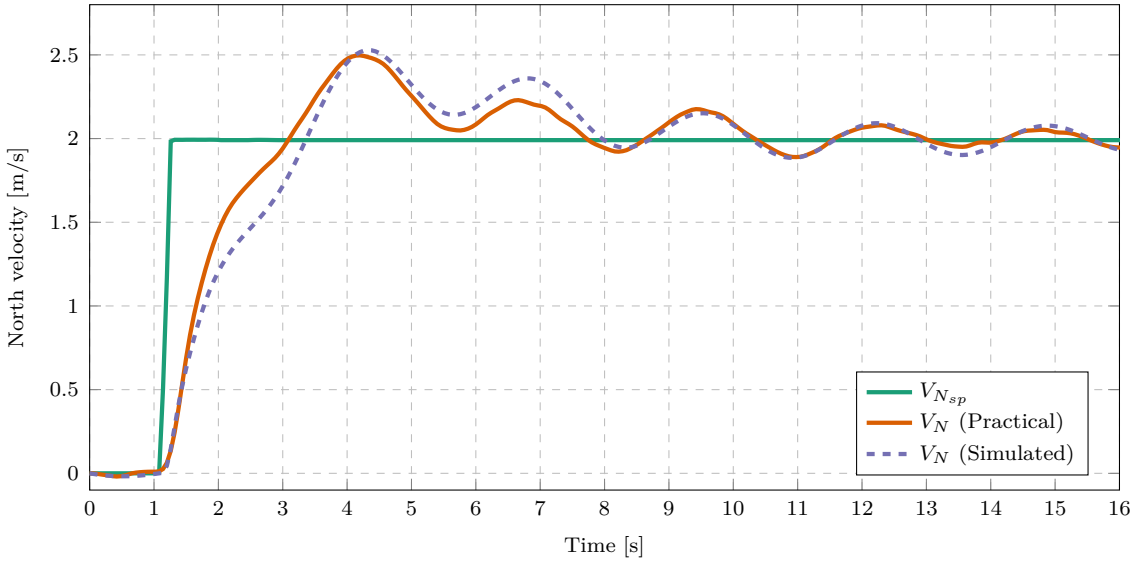


Figure 3.5: Velocity step comparison of simulated and practical data for Honeybee with a suspended payload.

The same procedure was followed to verify the multirotor with a suspended payload model. Figure 3.5 compares the practical and simulated velocity step responses with a suspended payload attached to the vehicle. The velocity curve shapes matches well in this comparison. This seems to be an impressive result for a complex system with many interlinking elements. Both systems provide an overshoot of a similar value. The amplitude and frequency of the velocity oscillations also appear to be very similar.

The slight difference between the curves is attributed to the high complexity of the multirotor-payload model. The suspended payload model adds modelling assumptions to the system which decreases the accuracy of the model. The system considers many state variables that each need to be assigned an accurate initial condition. Furthermore, the effect of wind disturbances is expected to be greater for the multirotor-payload system than for the multirotor without a load.

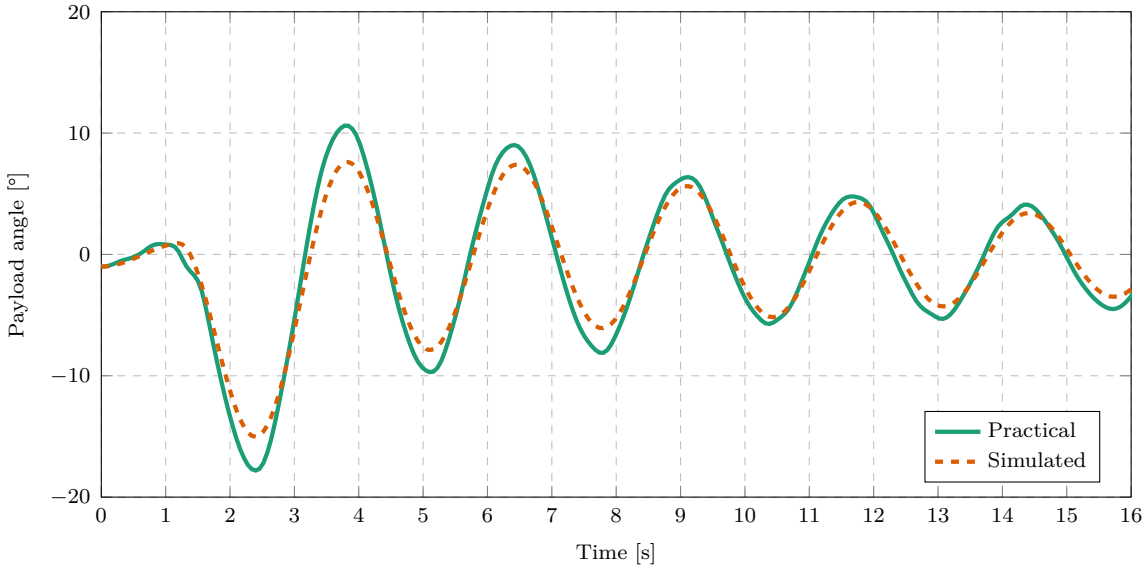


Figure 3.6: Payload angle comparison of simulated and practical data for Honeybee with a suspended payload.

Figure 3.6 shows the payload angle for the same flight as Figure 3.5. It is clear that the payload angle response of the simulated system matches the practical data well.

3.7. Summary

This chapter showed the derivation of a mathematical model representing the Honeybee multirotor with a suspended payload. A comparison of the simulation model and practical data showed that the model provides a good representation of the actual multirotor-payload system. This mathematical model will therefore be used for simulations in subsequent chapters.

Chapter 4

System identification

System identification is the process of creating a mathematical model of a dynamical system by using input and output measurements of that system. In this work a mathematical model of the multirotor and suspended payload system is required for swing damping control. The two major approaches to system identification are:

1. A priori mathematical modelling with parameter estimation
2. Data-driven system identification

This chapter will discuss these approaches and describe the differences between them. For each approach, specific estimation techniques will be applied to simulation data of the multirotor and payload system. The results of these techniques will then be discussed.

4.1. White-box and black-box techniques

Models determined from data-driven system identification methods are generally called black-box models. The user is only concerned with the inputs and output of the model and does not need to derive mathematical relationships from theoretical deductions. In contrast, white-box models are determined from a priori modelling and the user defines the physics of white-box models.

4.1.1. White-box techniques

The underlying physics of a white-box model is usually determined from first principles. This is done by modelling physical processes with techniques such as Lagrangian or Newton mechanics. Hence, the mathematical relations between system parameters in the model are predefined in the modelling phase. The system identification process is therefore reduced to parameter estimation which determines the best fit values of the system parameters.

This approach is used by [5] and [6] for system identification for swing damping control of a multirotor with an unknown suspended payload. Recall from Chapter 3 that the system was modelled as two connected rigid bodies with the following assumptions:

- The payload is a point-mass.
- The link is massless.
- The link is rigid.
- The link is attached to the CoM of the multirotor.

The only unknown parameters in the multirotor and payload model is the payload mass and the link length. These parameters are first estimated and then inserted into the predefined, linearised model. This model is used by a LQR controller to damp swing angles while also controlling the vehicle.

The main advantage of this approach is its simplicity. In the case considered by [5] and [6], only two parameters are estimated. In contrast, numerous values need to be estimated to reproduce the system dynamics with a black-box model. Therefore, white-box system identification methods are often less computationally complex and can easily be applied on low cost hardware. Due to the lower complexity, parameter estimation algorithms often require shorter lengths of training data than data-driven methods to produce accurate models.

Therefore the white-box approach works well for systems with predictable physics, however is not very adaptable to systems that deviate from the predefined dynamics. The payload considered by [5] and [6] is limited to a small rigid mass suspended from the multirotor by a non-stretching cable. In this configuration it was shown that a LQR controller successfully controls a multirotor and minimises the payload swing angles. However, if a payload or cable is used that violates one of the modelling assumptions, the predefined model no longer accurately represents the system. Many payloads considered for practical drone deliveries do not conform to these assumptions. Since the controller is dependent on this model, the mismatch between the model and actual dynamics may result in undesirable controller behaviour. Therefore a new model and parameter estimation technique will need to be derived for every use case that deviates significantly from the a priori model.

4.1.2. Black-box techniques

Data-driven system identification methods produce black-box models. These models do not require predefined mathematical relations between system parameters. No prior knowledge of the physics of the system are considered and no modelling assumptions are made. Black-box techniques determine the mathematical relationship between inputs and outputs of a system using information from measurement data only.

A disadvantage of the data-driven system identification approach is its computational complexity. Data-driven algorithms generally have a much higher computational complexity than parameter estimation techniques. This is expected since a lot more model parameter values are generated to populate a black-box model than a predefined white-box model. In the multirotor use case, this may mean that more expensive computational hardware is required to implement a data-driven method compared to parameter estimation methods. Also, most data-driven methods have hyperparameters that affect the performance of the method and need to be tuned for a specific use case. This can be done automatically, but this process increases the computational demand on the hardware. Furthermore, data-driven methods generally require more training data than parameter estimation methods. This means that more flight time is wasted on system identification before an updated controller can be activated.

However, black-box techniques are very adaptable and provide a general system identification solution for a broad range of different dynamics. This is a major advantage over white-box system identification techniques, which need to be manually redesigned for different use cases.

Dynamical models can be categorised as either non-linear or linear models. Non-linear models are often more accurate than linear models because real-world system mostly contain non-linear dynamics. The dynamics of a multirotor and suspended payload are also non-linear.

However, non-linear models are inherently more complex than linear models. Controllers based on non-linear models are usually more computationally complex than those with linear models. The control architectures used for multirotors in practical applications are mostly implemented on onboard hardware. Therefore there is value in low-complexity, linear models because these may be simple enough to execute on low cost hardware. Non-linear models may require control implementations that are too computationally intensive and may not be practically realisable on the available hardware on a multirotor.

Dynamic Mode Decomposition with Control (DMDc) and Hankel Alternative View Of Koopman with Control (HAVOKc) are the two data-driven system identification methods investigated in this work. These are linear regression techniques that produce linear models that approximate non-linear dynamics. Non-linear data-driven techniques like Neural Networks and SINDy [84] may produce models that are more accurate than linear techniques. However the gain in accuracy will be at the cost of a greater computational complexity. DMDc and HAVOKc are less computationally complex and their models are suitable for linear MPC, which is significantly faster than non-linear MPC. This is

desirable for a practical multirotor implementation, where onboard computational power is limited.

4.2. Plant considered for system identification

A specific subsystem of the multirotor-payload system will be considered for system identification. Because the proposed controllers in Chapter 5 will be applied for North velocity control, the longitudinal dynamics will be considered in this section, therefore the resulting input vector of the system identification plant is given by,

$$\boldsymbol{u} = \begin{bmatrix} A_{N_{sp}} \end{bmatrix}, \tag{4.1}$$

where $A_{N_{sp}}$ is the North acceleration setpoint in the inertial frame. $A_{N_{sp}}$ is used by the attitude controllers which will be explained in Section 5.2. For swing damping control, the controllers will require feedback from both the multirotor and payload variables, therefore the state vector of the considered plant is,

$$\boldsymbol{x} = \begin{bmatrix} V_N & \theta & \dot{\theta} \end{bmatrix}^T, \tag{4.2}$$

where θ and $\dot{\theta}$ are the payload angle and angular rate about the inertial East axis, and V_N is the North velocity of the multirotor in the inertial frame. A schematic of this Two-Dimensional (2D) plant is shown in Figure 4.1.

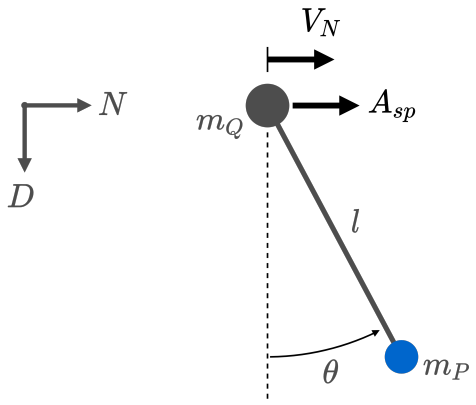


Figure 4.1: Schematic of a floating pendulum model considered for a North velocity controller

In subsequent sections, simulations of the full, non-linear multirotor and payload system will be performed and different methods will be applied to determine system identification

models of this plant.

4.3. Parameter estimation

The purpose of parameter estimation is to determine unknown values required by a predetermined, white-box model. The identified model is used to design a LQR controller and therefore needs to be in the form of a linear, continuous-time, state-space model. This model was derived by [6] and will be presented in the section below. The unknown parameters in the model include the payload mass, m_p and the cable length, l. Two separate methods are used to estimate each of these parameters. This approach was used successfully by [5] and [6] to implement a LQR swing damping controller for a multirotor with an unknown suspended payload.

4.3.1. White-box model

A white-box model was derived a priori by [6] to represent the dynamics of the multirotor-payload system. The non-linear dynamical equations derived in Chapter 3 were linearised to produce a linear state-space model. The equations were linearised around hover by applying small angle approximations. As discussed in Section 3.5.1, the payload is attached to the CoM of the multirotor. Therefore the vehicle attitude dynamics are decoupled from the payload dynamics and are not considered in this model. Aerodynamic drag forces are assumed to be negligible and are ignored in this model. It is also assumed that the attitude controllers have a large enough time-scale separation from the velocity controllers, such that $A_{N_{sp}} \approx A_N$.

The resultant linear state-space model of the longitudinal dynamics is given by,

$$\dot{\boldsymbol{x}}_{long} = \boldsymbol{A}_{long}\boldsymbol{x}_{long} + \boldsymbol{B}_{long}\boldsymbol{u}_{long}, \tag{4.3}$$

where

$$\boldsymbol{x}_{long} = \begin{bmatrix} V_N & \theta & \dot{\theta} \end{bmatrix}^T, \tag{4.4}$$

$$\boldsymbol{u}_{long} = \begin{bmatrix} A_{N_{sp}} \end{bmatrix}, \tag{4.5}$$

$$\boldsymbol{A}_{long} = \begin{bmatrix} 0 & \frac{m_p \cdot g}{m_Q} & \frac{c}{(l \cdot m_Q)} \\ 0 & 0 & 1 \\ 0 & \frac{(m_p + m_Q) \cdot g}{(m_Q \cdot l)} & \frac{-c \cdot (m_p + m_Q)}{(l^2 \cdot m_Q \cdot m_p)} \end{bmatrix}, \text{ and} \tag{4.6}$$

$$\boldsymbol{B}_{long} = \begin{bmatrix} 1 \\ 0 \\ -\frac{1}{l} \end{bmatrix}. \tag{4.7}$$

It is assumed that the coefficient of friction, c, does not change for different payload, therefore this parameter can be determine experimentally and will known before a flight. Furthermore, it is assumed that m_Q is known. The payload parameters, m_p and l, are unknown before a flight. These parameters will be estimated in the sections below.

4.3.2. Payload mass estimation

RLS is used by [5] and [6] to estimate the payload mass. It is assumed that the multirotor mass is known before a flight, therefore the payload mass can be estimated from the additional trust required during hover. In both [5] and [6] it is demonstrated that RLS is very accurate for a system nearly identical to the one considered in this work. To compare other aspects of the white-box and black-box techniques with more clarity, it will be assumed that the method estimates m_p with perfect accuracy. This isolates any inaccuracies in the white-box model to either the a priori modelling or the cable length estimation.

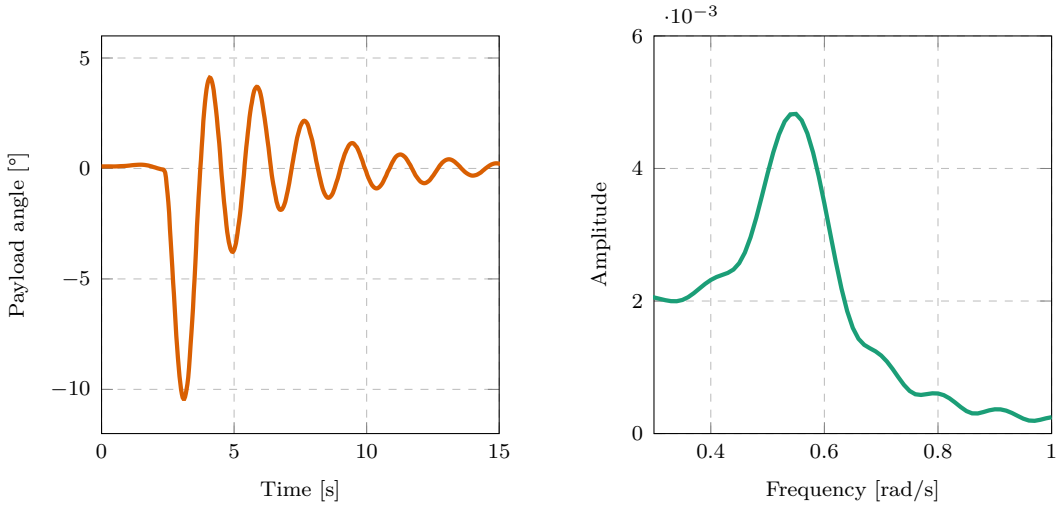
It should be noted however that this method is dependant on the assumption that the vehicle mass is known and remains unchanged. The method will clearly be inaccurate if an unknown mass is added to the multirotor in conjunction with the suspended payload. Common practical examples of this include adding a camera to the vehicle or using a different battery for a different flight range. In these cases, the mass estimation method will have to be redesigned. This is an inherent problem of the white-box techniques. The parameter estimation methods are designed for specific modelling assumptions and are not adaptable to different types of payload loadings. In contrast, data-driven techniques are adaptable to different payload loadings because it does not depend on a priori modelling assumptions.

4.3.3. Cable length estimation

The cable length is estimated from the measurement of the natural frequency of the swinging payload. As described by [47], the natural frequency is given by:

$$\omega_n = \sqrt{\frac{g}{l} \cdot \frac{m_Q + m_p}{m_Q}}. \tag{4.8}$$

The cable length can clearly be calculated from Equation 4.8 if the other parameters are known. The natural frequency is measured by performing a FFT on the payload swing angle response after a velocity step by the multirotor. The dominant frequency identified by the FFT during free swing is an approximate measurement of the natural frequency of the payload. Note that the measured frequency rather corresponds to the damped natural frequency. The swing angle damping is caused by the velocity controller,



(a) Position step response of the payload swing angle.

(b) The single-sided amplitude spectrum of the FFT.

Figure 4.2: Data from a velocity step response used for cable length estimation ($l = 1\,\text{m}$, $m_p = 0.3\,\text{kg}$.).

friction at the attachment of the cable to the drone, and air drag. However, it is assumed that the damping coefficient is small enough for the damped natural frequency to closely approximate the theoretical natural frequency.

Figure 4.2a shows the payload swing angle response to a position step setpoint. The first few seconds of the step response are excluded from the FFT to minimise the effect of the transient response and the multirotor controllers on the natural frequency measurement. Figure 4.2b shows the resulting single-sided amplitude spectrum of the FFT of this data.

The dominant frequency is clearly identified by the peak at $0.520\,\text{rad/s}$. Since m_Q, m_p and g are known, and ω_n has been measured, l can be determined from Equation 4.8. The estimated length for this simulation is $0.953\,\text{m}$. The actual cable length is $1\,\text{m}$, therefore this estimation has an error of 4.7%. As documented by [5] and [6], an error of this magnitude is acceptably small and still results in effective control with a LQR. It was also shown by [5] and [6] that this estimation method is effective for a range of different payloads in simulation.

4.4. Dynamic mode decomposition with control

Dynamic Mode Decomposition (DMD) is a regression technique that can be used to approximate a non-linear dynamical system with a linear model [85]. It uses temporal measurements of system outputs to reconstruct system dynamics without prior modelling assumptions. DMDc is an adaptation of DMD that also accounts for control inputs [86]. This section provides an overview of the specific implementation of DMDc used in this

work. Note that this implementation is an adaptation of DMDc, and includes time-delay-embedding of multiple variables. Enriching a DMD model with time-delay-embedding is a known technique and is also seen in other DMD adaptations [87, 88].

DMD produces a linear, discrete state-space model of system dynamics. Discrete measurements, $\boldsymbol{x}_k$, of the continuous time variable, $\boldsymbol{x}(t)$, are used, where $\boldsymbol{x}_k = \boldsymbol{x}(kT_s)$, and T_s is the sampling time of the model. Delay-coordinates (i.e. $\boldsymbol{x}_{k-1}, \boldsymbol{x}_{k-2}$, etc.) are also included in the state-space model to account for input delay and state delay in the system. Input delay refers to the time delay involved with transporting a control signal to a system, whereas state delay refers to time-separated interactions between system variables [89]. Hence, we define a state delay vector as:

$$\boldsymbol{d}_k = \begin{bmatrix} \boldsymbol{x}_{k-1} & \boldsymbol{x}_{k-2} & \cdots & \boldsymbol{x}_{k-q+1} \end{bmatrix}^T, \tag{4.9}$$

where q is the number of delay-coordinates (including the current time-step) used in the model, and $\boldsymbol{d}_k \in \mathbb{R}^{(n_x)(q-1)}$.

The discrete state-space model is therefore defined as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_{dmd}\boldsymbol{x}_k + \boldsymbol{A}_d\boldsymbol{d}_k + \boldsymbol{B}_d\boldsymbol{u}_k, \tag{4.10}$$

where $\boldsymbol{A}_{dmd} \in \mathbb{R}^{n_x \times n_x}$ is the system matrix, $\boldsymbol{A}_d \in \mathbb{R}^{(q-1)\cdot n_x \times (q-1)\cdot n_x}$ is the state delay system matrix and $\boldsymbol{B}_d \in \mathbb{R}^{n_x \times n_u}$ is the input matrix.

The training data consists of full-state measurements, $\boldsymbol{x}_k$, and corresponding inputs, $\boldsymbol{u}_k$, taken at regular intervals of $\Delta t = T_s$, during a simulated flight with Cascaded PID control. In a practical flight, these time-series measurements need to be saved in memory because it is processed as a single batch by DMD. Note that DMD can be applied in a recursive manner as described in [90]. However this implementation is not considered because an On-Board Computer (OBC) with significant memory size can be used.

The training data is collected into the following matrices:

$$\begin{aligned}
\boldsymbol{X'} &= \begin{bmatrix} \boldsymbol{x}_{q+1} & \boldsymbol{x}_{q+2} & \boldsymbol{x}_{q+3} & \cdots & \boldsymbol{x}_{w+q} \end{bmatrix}, \\
\boldsymbol{X} &= \begin{bmatrix} \boldsymbol{x}_{q} & \boldsymbol{x}_{q+1} & \boldsymbol{x}_{q+2} & \cdots & \boldsymbol{x}_{w+q-1} \end{bmatrix}, \\
\boldsymbol{X}_d &= \begin{bmatrix} \boldsymbol{x}_{q-1} & \boldsymbol{x}_{q+0} & \boldsymbol{x}_{q+1} & \cdots & \boldsymbol{x}_{w+q-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_{2} & \boldsymbol{x}_{3} & \boldsymbol{x}_{4} & \cdots & \boldsymbol{x}_{w+1} \\ \boldsymbol{x}_{1} & \boldsymbol{x}_{2} & \boldsymbol{x}_{3} & \cdots & \boldsymbol{x}_{w} \end{bmatrix}, \\
\boldsymbol{\Upsilon} &= \begin{bmatrix} \boldsymbol{u}_{q} & \boldsymbol{u}_{q+1} & \boldsymbol{u}_{q+2} & \cdots & \boldsymbol{u}_{w+q-1} \end{bmatrix},
\end{aligned} \tag{4.11}$$

where w is the number of columns in the matrices, $\boldsymbol{X'}$ is the matrix $\boldsymbol{X}$ shifted forward by one time-step, $\boldsymbol{X}_d$ is the matrix with delay states, and $\boldsymbol{\Upsilon}$ is the matrix of inputs. Equation 4.10 can be combined with the matrices in Equation 4.11 to produce:

$$\boldsymbol{X'} = \boldsymbol{A}_{dmd}\boldsymbol{X} + \boldsymbol{A}_d\boldsymbol{X}_d + \boldsymbol{B}_d\boldsymbol{\Upsilon}. \tag{4.12}$$

Note that the primary objective of DMDc is to determine the best fit model matrices, $\boldsymbol{A}_{dmd}$, $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$, given the data in $\boldsymbol{X'}$, $\boldsymbol{X}$, $\boldsymbol{X}_d$, and $\boldsymbol{\Upsilon}$ [86]. In order to group the unknowns into a single matrix, Equation 4.10 is manipulated into the form,

$$\boldsymbol{X'} = \begin{bmatrix} \boldsymbol{A}_{dmd} & \boldsymbol{A}_d & \boldsymbol{B}_d \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}_d \\ \boldsymbol{\Upsilon} \end{bmatrix} = \boldsymbol{G\Omega}, \tag{4.13}$$

where $\boldsymbol{\Omega}$ contains the state and control data, and $\boldsymbol{G}$ represents the system and input matrices.

A Singular Value Decomposition (SVD) is performed on $\boldsymbol{\Omega}$ resulting in: $\boldsymbol{\Omega} = \boldsymbol{U\Sigma V}^T$. Often, only the first p columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are required for a good approximation of the dynamics [91]. In many cases, the truncated form results in better models than the exact form when noisy measurements are used. This is because the effect of measurement noise is mostly captured by the truncated columns of $\boldsymbol{U}$ and $\boldsymbol{V}$. By truncating these columns, the influence of noise in the regression problem is reduced. Hence the SVD is used in the truncated form:

$$\boldsymbol{\Omega} \approx \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^T, \tag{4.14}$$

where $\tilde{}$ represents rank-p truncation. For the $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices, rank-p truncation refers to keeping only the first p number of columns and truncating the rest. Rank-p truncation of the $\boldsymbol{S}$ matrix refers to keeping the first p number of columns and rows and truncating the rest.

By combining Equation 4.14 with the over-constrained equality in Equation 4.13, the least-squared solution, $\boldsymbol{G}$, can be found with:

$$\boldsymbol{G} \approx \boldsymbol{X'}\tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{U}}. \tag{4.15}$$

By reversing Equation 4.13, $\boldsymbol{G}$ can now be decomposed into: $\boldsymbol{G} = \begin{bmatrix}\boldsymbol{A}_{dmd} & \boldsymbol{A}_d & \boldsymbol{B}_d\end{bmatrix}$ according to the required dimensions of each matrix. Thereby, the state-space model approximated by DMDc is complete.

4.5. Hankel alternative view of Koopman with control

Hankel Alternative View Of Koopman (HAVOK) is a data-driven, regression technique that provides a different approximation of the Koopman operator than DMD [91, 92]. The Koopman operator is a method of representing finite-dimensional non-linear dynamics in terms of an infinite-dimensional linear operator [91]. However, an infinite-dimensional linear operator is not very useful for practical implementation. DMD provides a limited approximation of the Koopman operator, because it is based on linear measurements only. HAVOK was developed to improve the Koopman approximation of DMD by using intrinsic measurement coordinates based on the time-history of the system [91].

The original formulation of HAVOK is only defined for uncontrolled dynamical systems [91]. In this work, we have adapted the standard HAVOK algorithm to account for the effect of control. This implementation well be referred to as HAVOKc. This algorithm results in a discrete, linear model that approximates the behaviour of a controlled dynamical system. In this section, a brief overview is provided of this implementation of HAVOKc.

A defining characteristic of HAVOK is that it uses multiple delay-coordinates (i.e. $\boldsymbol{x}_{k-1}, \boldsymbol{x}_{k-2}$, etc.) in the system identification process. To fit this into the standard state-space format, an extended state vector is defined as:

$$\boldsymbol{a}_k = \begin{bmatrix}\boldsymbol{x}_{k-(q-1)} & \cdots & \boldsymbol{x}_{k-1} & \boldsymbol{x}_k\end{bmatrix}^T, \tag{4.16}$$

where $\boldsymbol{a}_k \in \mathbb{R}^{(n_y)(q)}$, and the subscript of $\boldsymbol{a}$ denotes the highest subscript of $\boldsymbol{x}$ in the vector.

The resulting discrete state-space model is therefore in the form,

$$\boldsymbol{a}_{k+1} = \boldsymbol{A}_H\boldsymbol{a}_k + \boldsymbol{B}_H\boldsymbol{u}_k, \tag{4.17}$$

where $\boldsymbol{A}_H \in \mathbb{R}^{(q\cdot n_x)\times(q\cdot n_x)}$ is the system matrix, and $\boldsymbol{B}_H \in \mathbb{R}^{(q\cdot n_x)\times n_u}$ is the input matrix.

The original HAVOK algorithm, developed by Brunton et al. [91], constructs a Hankel matrix from output variables only. In this work, the standard HAVOK algorithm has been adapted to incorporate the effect of control. An extended Hankel matrix, $\mathbf{\Pi}$, is created by appending a matrix of inputs to a Hankel matrix of measurements:

$$\mathbf{\Pi} = \begin{bmatrix} \boldsymbol{a}_q & \boldsymbol{a}_{q+1} & \boldsymbol{a}_{q+2} & \cdots & \boldsymbol{a}_{w+q-1} \\ \boldsymbol{u}_q & \boldsymbol{u}_{q+1} & \boldsymbol{u}_{q+2} & \cdots & \boldsymbol{u}_{w+q-1} \end{bmatrix}, \tag{4.18}$$

where w is the number of columns in $\mathbf{\Pi}$. A truncated SVD of this Hankel matrix results in following approximation:

$$\mathbf{\Pi} \approx \tilde{\boldsymbol{U}}\tilde{\mathbf{\Sigma}}\tilde{\boldsymbol{V}}^T, \tag{4.19}$$

where $\tilde{}$ represents rank-p truncation. It is important to note that the model extracted by HAVOKc depends on the choice of hyperparameters (p and q), and the number of training samples ($N_{train} = w + q - 1$).

The columns of $\tilde{\boldsymbol{V}}$ are the most significant principal components of the system dynamics [93]. This matrix, $\tilde{\boldsymbol{V}}$, can be considered to contain a time-series of the pseudo-state, $\boldsymbol{v}$, such that $\tilde{\boldsymbol{V}}^T = \begin{bmatrix} \boldsymbol{v}_q & \boldsymbol{v}_{q+1} & \cdots & \boldsymbol{v}_w \end{bmatrix}$, characterises the evolution of the actual dynamics in an eigen-time-delay coordinate system [91]. Consider the following discrete, state-space formulation:

$$\boldsymbol{v}_{k+1} = \mathbf{\Lambda}\boldsymbol{v}_k. \tag{4.20}$$

HAVOKc determines the best fit linear operator $\mathbf{\Lambda}$ that maps the pseudo-state $\boldsymbol{v}_k$ to $\boldsymbol{v}_{k+1}$. In order to setup an over-determined equality for Equation 4.20, $\tilde{\boldsymbol{V}}^T$ is divided into two matrices:

$$\begin{aligned} \boldsymbol{V}_1 &= \begin{bmatrix} \boldsymbol{v}_q & \boldsymbol{v}_{q+1} & ... & \boldsymbol{v}_{w-1} \end{bmatrix}, \\ \boldsymbol{V}_2 &= \begin{bmatrix} \boldsymbol{v}_{q+1} & \boldsymbol{v}_{q+2} & ... & \boldsymbol{v}_w \end{bmatrix}, \end{aligned} \tag{4.21}$$

where $\boldsymbol{V}_2$ is $\boldsymbol{V}_1$ advanced a single step forward in time. The matrices from Equation 4.21 are now combined with Equation 4.20 and the best fit $\mathbf{\Lambda}$ is determined with the Moore-Penrose pseudoinverse:

$$\boldsymbol{V}_2 = \mathbf{\Lambda}\boldsymbol{V}_1 \qquad \Rightarrow \qquad \mathbf{\Lambda} \approx \boldsymbol{V}_1\boldsymbol{V}_1^{\dagger} \tag{4.22}$$

It can be shown from Equation 4.19 that Equation 4.20 is transformed from the eigen-time-delay coordinate system to the original coordinate system as the following:

$$\begin{bmatrix} \boldsymbol{a}_{k+1} \\ \boldsymbol{u}_{k+1} \end{bmatrix} = (\tilde{\boldsymbol{U}}\tilde{\mathbf{\Sigma}})\mathbf{\Lambda}(\tilde{\boldsymbol{U}}\tilde{\mathbf{\Sigma}})^{\dagger} \begin{bmatrix} \boldsymbol{a}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{4.23}$$

$\boldsymbol{A}_H$ and $\boldsymbol{B}_H$ can now be extracted from the matrix, $(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}})\boldsymbol{\Lambda}(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}})^{\dagger}$. This extraction is illustrated in Figure 4.3, where different blocks represent different groups of matrix entries.

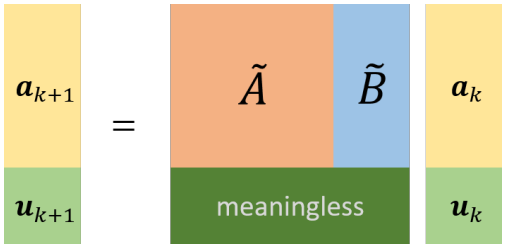


Figure 4.3: Illustration of the extraction of $\boldsymbol{A}_H$ and $\boldsymbol{B}_H$ from Equation 4.23

Note that the matrix entries in Figure 4.3 that map $\boldsymbol{u}_k$ to $\boldsymbol{u}_{k+1}$ are meaningless for state predictions and are discarded. Also note that the state vector, $\boldsymbol{a}_k$, includes delay-coordinates, therefore some matrix entries are independent of the dynamics. This is illustrated in Figure 4.4 for an example model with $q = 4$. For example, the mapping of $\boldsymbol{x}_k$ in the state vector to $\boldsymbol{x}_k$ in the predicted state vector corresponds to a entry of 1 in the $\boldsymbol{A}_H$ matrix. This is fixed by the model format and is not a function of the system dynamics. Due to the least-squares fitting and the coordinate transformation of the algorithm, HAVOKc does not produce these exact values in $\boldsymbol{A}_H$ and $\boldsymbol{B}_H$. By forcing each of these matrix entries to 1 or 0, the state-prediction performance of the model is improved. Finally, the improved $\boldsymbol{A}_H$ and $\boldsymbol{B}_H$ are inserted into Equation 4.17 to render the HAVOKc model.

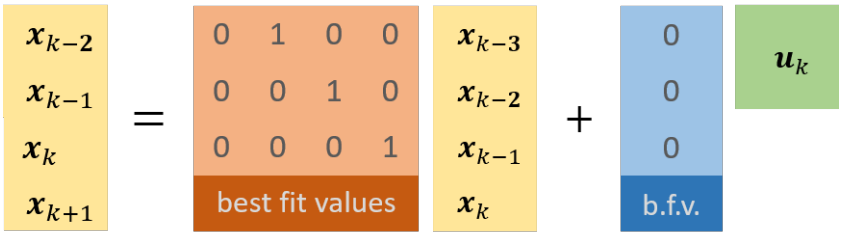


Figure 4.4: Illustration of forcing the known values in HAVOKc matrices

For a high-level comparison of the data-driven methods, recall from Section 4.4 that DMDc applies least-squares regression directly to the collected state and input data matrices. HAVOKc first applies an SVD to the state and input data, and truncates the insignificant modes. Least-squares regression is then applied to the pseudo-state data from the truncated SVD to determine a state-space model. This model is then transformed back into the original coordinate frame.

4.6. Implementation and results

In this section, the techniques introduced in Section 4.3, 4.4, and 4.5 will be applied to simulation data. Firstly, the influence of the design parameters on the algorithm performance will be discussed. These parameters include hyperparameters, the length of training data and the algorithm sample time. The effect of conditions that are not determined by algorithm design will also be explored, like measurement noise, and the physical properties of the payload. Finally, the white-box and black-box techniques will be tested on a dynamic payload which does not satisfy the assumptions of a simple pendulum.

4.6.1. Methodology

A Software-in-the-Loop (SITL) implementation of the PX4 Autopilot [94] using the Gazebo simulator [95] is used to generate data for system identification. Testing these techniques within simulations allows us to investigate a much larger range of system configurations than possible with practical flights. The simulation model used in Gazebo was verified in Chapter 3. Using PX4 in SITL also ensures that the controller dynamics in simulation is as close as possible to practical flights since the same flight stack is used in both cases. Gazebo applies realistic measurement noise to the signals received by PX4 and the PX4 flight stack applies an EKF for state estimation. Therefore the the data seen by the system identification techniques include the same EKF filtering as practical flight data.

The procedure used to evaluate the black-box techniques is as follows:

1. Takeoff and hover with the multirotor
2. Start logging input and output data
3. Command a series of velocity step setpoints with random step sizes and time intervals
4. Stop logging data
5. Split data into separate training and testing periods
6. Build a model from the training data
7. Calculate an error metric for the model from the testing data

The default PID velocity controller from PX4 is used during these simulation. The implemented controller gains are documented in Appendix A. A Robot Operating System (ROS) node is used to read and log the payload angle measurement from Gazebo and a different ROS node is used to send velocity setpoints to PX4 with the MAVLink protocol through the popular ROS package, 'mavros'.

A algorithm schedules the series of velocity step commands by assigning random step values and time-intervals within a specified range. These values are selected from a uniform distribution within the ranges specified in Table 4.1 The maximum velocity step is determined in simulation by iteratively increasing the maximum velocity step to a safe value where the multirotor remains in stable flight and the payload angles do not swing out of control. The time interval range is set iteratively to ensure that the generated data includes both transient and steady-state dynamics.

Table 4.1: Input data ranges.

	Velocity step [m/s]	Step time interval [s]
Minimum	0	10
Maximum	3	25

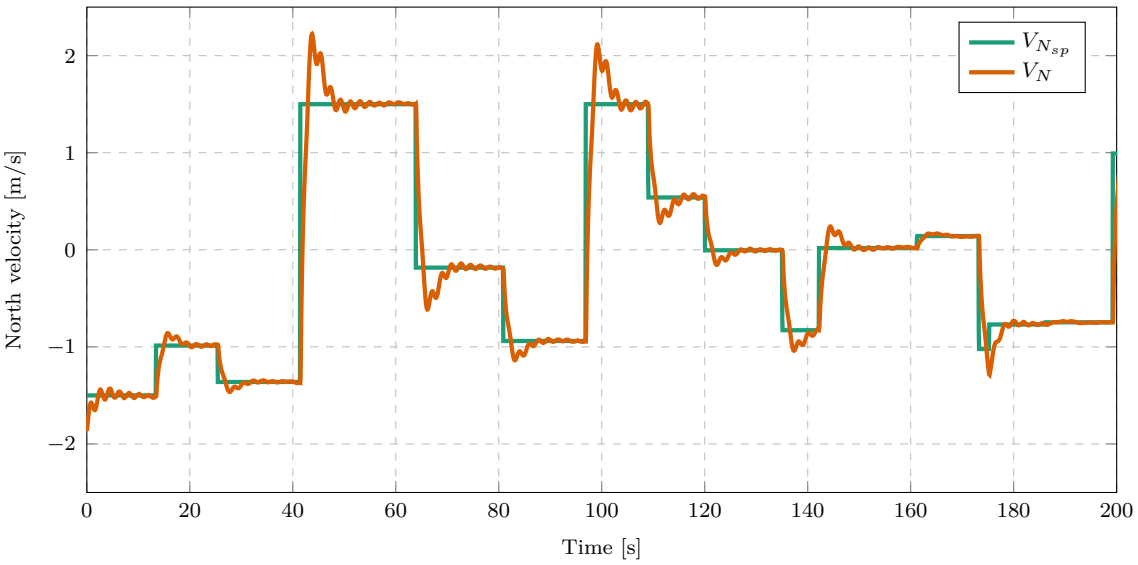


Figure 4.5: Example of training data with random velocity step inputs ($m_p = 0.2\,\text{kg}$, $l = 1\,\text{m}$).

Figure 4.5 shows an example of random velocity steps and the resulting velocity response used as training data. Using random velocity steps and time intervals prevents the system identification methods from overfitting to a specific set of control conditions. The method should rather determine a generalised model that works over a range of possible control conditions. The data logged for the duration of the training data shows minimal altitude fluctuations. These fluctuations appear to be due to GPS noise rather than the swinging payload, since they do not show high frequency oscillations as seen in the North velocity. This supports the assertion in 4.2 that the considered plant approximates a pendulum-on-a-cart model during longitudinal control.

The data logged from simulation is then divided into testing and training data. The training data is used by the system identification algorithms to generate a regression model and the model is then used to determine a prediction error metric over the unseen testing data. It is common practice in model evaluations to use separate sets of data for training and testing. This ensures that good prediction scores do not result from models that overfit the training data.

The testing data spans a fixed length of time and is taken from the start of the simulation period. The training data is then extracted from the remainder of the data. The same setpoint schedules are used to generate this data for different simulations to ensure that the metrics determined from different simulations are comparable. The error metric calculated from the testing data is used to evaluate and rank the performances of each model.

4.6.2. Error metric

It is common practice is to select a model for an MPC based on k-step-ahead prediction errors [96]. This is because the model is used to make k-step-ahead predictions during control optimisation. When model error is dominated by variance error (caused by disturbances), it may be better to use one-step-prediction error [96]. However for the multirotor and payload case it is assumed that variance error (caused by under modelling) dominates the model error.

Different metrics are used in literature to quantify prediction accuracy for different applications. Very common, scale-dependant error metrics are Mean Squared Error (MSE) and MAE. These metrics are dependant on the unit and scale of a variable, hence they cannot be used to compare predictions of different variables. MSE (l_2 norm of error values) penalises larger errors more than smaller errors, whereas MAE (l_1 norm) penalises errors equally. For our use case the l_1 norm provides a more intuitive metric than the l_2 norm because it has the same unit as the prediction variable and there is no motivation to penalise larger errors more than smaller errors for our use case.

MAE is calculated as:

$$\boldsymbol{MAE} = mean\left(\ |\hat{\boldsymbol{x}}_{\boldsymbol{k}} - \boldsymbol{x}_{\boldsymbol{k}}| \ \right), \tag{4.24}$$

where $\boldsymbol{MAE}$ is a vector with the MAE of each state, $\boldsymbol{x}_{\boldsymbol{k}}$ is the actual state vector at time-step k, $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ is the state prediction, and $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ is the state prediction.

Popular, scale-free error metrics, like Mean Absolute Percentage Error (MAPE), Mean Relative Absolute Error (MRAE) and Mean Absolute Scaled Error (MASE), are also based on the l_1 norm, but are independent of the scale and units of a variable [97]. These

metrics could therefore be used to compare predictions of different variables. However, these metrics provide misleading comparisons for our use case. MAPE expresses accuracy as the absolute ratio between the error and actual value at each time-step. This results in undefined or extremely large values for the payload angle predictions because the state has a zero mean. The velocity state variable has a non-zero mean, therefore the scale of the MAPE of velocity will significantly different from the MAPE of the payload angle.

MRAE is also popular metric for comparing predictions models used with an MPC [98], however, similarly to MAPE, it also results in undefined values for the payload swing angle. MASE does not have this problem and can compare predictions of different variables well, because it expresses accuracy as the ratio between the MAE of the model prediction and the MAE of an in-sample naive forecast [97]. However, an in-sample forecast is a naive prediction for a one-step-ahead prediction, but not for a k-step-ahead prediction. Therefore MASE is not a helpful ratio for our use case.

Normalised Mean Absolute Error (NMAE) (Normalised Mean Absolute Error) is a scale-free error metric which provides a fair comparison of different variables in our use case. In this work NMAE normalises the MAE of a variable by the range of that variable, thereby variables with different ranges or different means can be compared. This value is calculated as:

$$NMAE = \frac{MAE}{x_{max} - x_{min}} \tag{4.25}$$

where $x_{i,max}$ and $x_{i,min}$ are the maximum and minimum values of the considered variable in the testing data.

This results in an error metric for each predicted variable, but a single value is required per model to rank the overall accuracy of different models. Therefore the average of the NMAE of all state variables is used as the single value representing the overall accuracy of a model and is denoted as $\overline{NMAE}$. This is the final error metric used to evaluate the model predictions in the sections to follow.

Other criteria which are more statistically rigorous in model selection than error metrics are Akaike's Information Criteria (AIC) and Bayesian Information Criteria (BIC) scores. Thereby they provide a quantitative way of performing a Pareto analysis, which balances model complexity with model accuracy [99]. It is generally advantageous to use a parsimonious model, which has a low prediction error but is not overly complex, than a complex model with a slightly lower prediction error. This not only reduces overfitting, but also ensures that the MPC optimisation problem is not too computational intensive for the available hardware. However, these scores require the computation of the maximum log likelihood of each model over numerous simulations. This is computationally intractable

and unpractical for our use case because of the large number of hyperparameter combinations to compare, as explained in Section 4.6.3. Therefore an error metric will rather be used to evaluate model accuracy.

The error metric of one model may change significantly different starting conditions or prediction horizons. The prediction horizon used for model analysis is selected as 20 s which is at least twice as long as the desired MPC prediction horizon. Some models have very accurate transient predictions, but prove to be unstable over a longer time horizon. If the prediction horizon is too short, these models may score unreasonably low error metrics. Selecting these such a model could result in unstable control at certain control conditions. Therefore a long prediction horizon is used for testing so that marginally unstable models are penalised heavily in model selection.

Different starting conditions also have a large influence on the prediction score of a model. Some models may accurately predict transient behaviour, while being extremely bad at steady-state predictions. This would result in an MPC controlling the plant well during the initial step response, but becoming unstable during steady-state control. In order to have an MPC that can control the plant during the different stages of a flight, a model needs to be selected with accurate predictions over a range of different control conditions.

Therefore the error metric needs to include predictions from multiple starting conditions in the testing data. The resulting testing procedure is to first specify a number of equispaced starting conditions within the testing data. The model is then run multiple times for the length of the prediction horizon, stating with different initial conditions each time. The NMAE is determined for each run, whereafter the average of these scores gives the final NMAE score of the model. In order to balance the variety of testing conditions with the computational time per error metric calculation, 10 prediction runs with different initial conditions used in the final NMAE score.

4.6.3. Hyperparameters

As discussed in Section 4.4 and 4.5 DMDc and HAVOKc models are dependent on two hyperparameters: the number of delay-coordinates, q, and the SVD truncation rank, p. For each system identification run with different system parameters or a different length of training data, a hyperparameter search is performed to find the combination of hyperparameters that results the lowest prediction error. Firstly, a coarsely spaced grid search is performed with large intervals between tested hyperparameter values. The range of tested hyperparameters is then reduced and a finer hyperparameter search is performed. From numerous simulation iterations, the range of significant hyperparameter values is

conservatively determined to be,

$$5 < q < 30, \qquad 5 < p < 50 \tag{4.26}$$

.

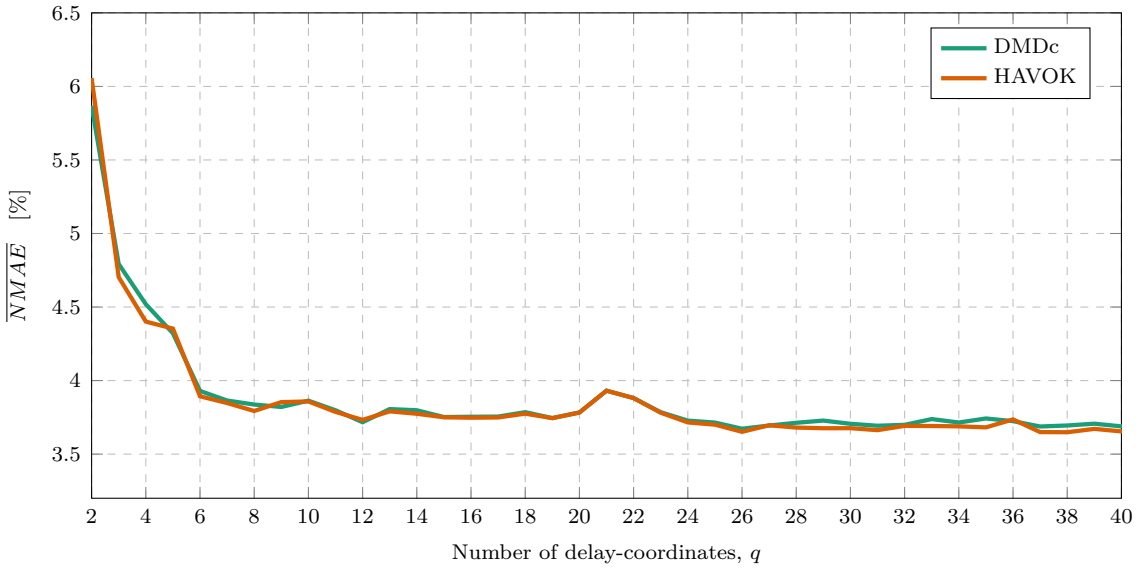


Figure 4.6: DMDc and HAVOKc predictions error for different lengths of noisy training data ($m_p = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$, $T_{train} = 60\,\text{s}$.).

Figure 4.6 shows the prediction error of DMDc and HAVOKc models for different values of q. For each value of a q, a new model is generated with every p in the considered range and the lowest prediction error is plotted. There is only a slight difference between the results of DMDc and HAVOKc. As expected, the models with the least number of terms have the highest prediction errors. As the number of terms available to the model increases, the error decreases. It is clear that there is a sharp decrease in prediction error for $2 < q < 6$, however there is no longer a significant decrease in error as model complexity increases past $q > 12$.

This 'elbow' in the plot can be considered as the Pareto front, where there is a balance between model complexity and accuracy [99]. It is desirable to select a parsimonious model on this front that has just enough free parameters to capture the plant dynamics and have good accuracy, without being overly complex [100]. These models are less prone to overfitting and also lead to lower computational complexity in the MPC optimisation problem.

Figure 4.7 plots the singular values of the SVD from a HAVOKc model in a log scale. The singular values of the SVD can be loosely interpreted as a measure of significance of

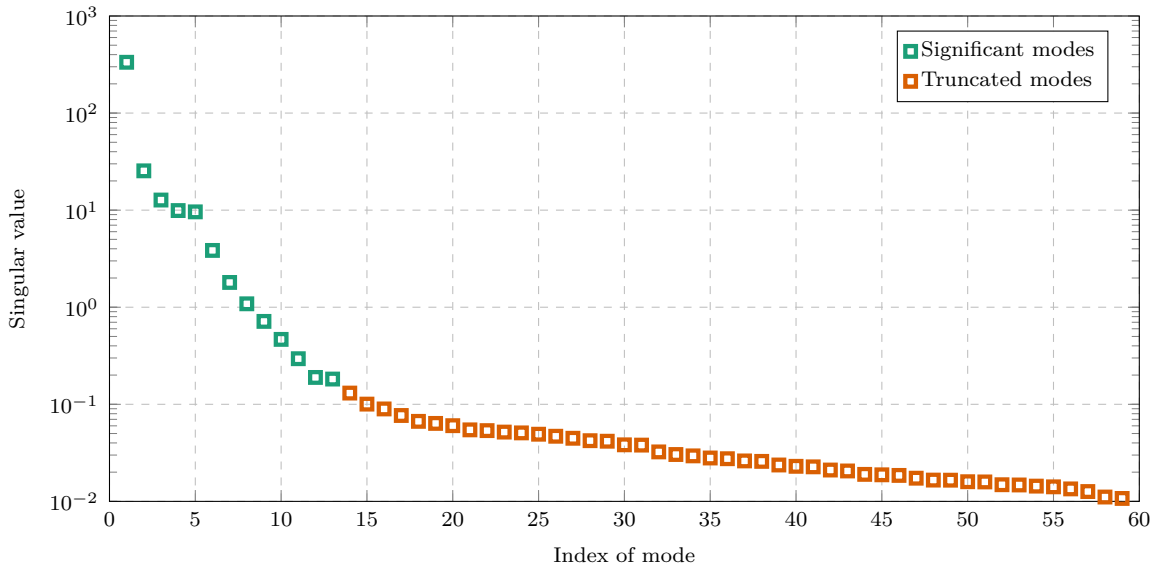


Figure 4.7: Significant and truncated singular values of a HAVOKc model produced from noisy data ($m_p = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$, $T_{train} = 60\,\text{s}$.).

the corresponding Proper Orthogonal Decomposition (POD) mode to the plant dynamics [101]. That is, modes with higher singular values contain more relevant information about the plant dynamics than modes with lower singular values. The p number of significant singular values, which correspond to the POD modes used to reconstruct the observed dynamics, are specifically shown in the plot. The truncated singular values are also shown, which correspond to the discarded modes.

The Pareto 'elbow' is also visible in this plot where there is noticeable change in gradient roughly at the split between significant and truncated values. This change in gradient shows that after p modes, there is a significant drop in information contributed per remaining mode. Note that the number p was selected from a hyperparameter search using an error metric which did not consider the singular values. This seems to confirm the notion that the Pareto optimal solution is often the most accurate representation of the actual dynamics [101].

4.6.4. Sample time

The sample time, T_s, used for system identification is the sample time of the discrete model, which determines the sample time of the MPC. Resampling strategies can enable the MPC to run at a different frequency to the discrete model but this adds unnecessary complexity to the control architecture.

The MPC acts in the velocity loop and commands an acceleration setpoint. The default PID velocity controller runs at 50 Hz which corresponds to $T_s = 0.02\,\text{s}$. Due to the

computational complexity of an MPC, the optimiser will struggle to run at 50 Hz on an OBC on a multirotor. However, the controller needs to run as fast as possible to have significant time-scale separation from the multirotor dynamics. If the controller runs too slowly, it may result in poor flight performance or unstable control. The highest natural frequency of the payload based on the range of physical parameters considered is 8.39 Hz corresponding to a period of 0.119 s.

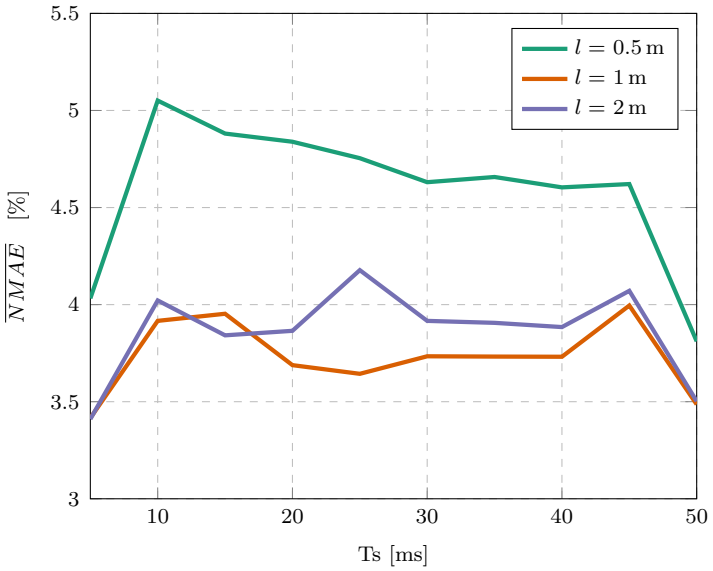


Figure 4.8: DMDc prediction error using different cable lengths with a range of different sample times of noisy training data ($m_p = 0.2\,\mathrm{kg}$).

Figure 4.8 shows the prediction error of different DMDc models generated with a range of different sample times. The natural frequency of the payload pendulum is dependant on the cable length and influences the frequency response of the plant. Therefore Figure 4.8 plots the experiment result for different cable lengths to see if it has an effect on the prediction error of the models.

Note that for all considered cable lengths, the prediction error has a sharp decrease for $T_s > 0.045\,\mathrm{s}$. This may be because the model does not try to capture the small, high-frequency oscillations in the dynamics at such slow sample times. Hence the long term prediction of the models fits the general shape of the dynamics well and results in low errors. However, this sample time is too slow for controlling the practical multirotor. $T_s = 0.03\,\mathrm{s}$ is selected as the sample time for system identification because it provides a good balance between being fast enough for the multirotor dynamics and slow enough for a practical MPC implementation.

4.6.5. Choice of payload variable in the state vector

As discussed in Section 4.2, the equations of motion in continuous-time of a floating pendulum are dependent on $\dot{\theta}$ and V_N, but are not dependent on θ. Therefore it is expected that $\boldsymbol{x} = \begin{bmatrix} V_N & \dot{\theta} \end{bmatrix}^T$ will be used as the state vector for system identification. However, if $\dot{\theta}$ is not included in the state vector of a discrete model, it can still be represented with numerical differentiation of θ. An example of this is the backward Euler form,

$$\dot{\theta}_k = (\frac{1}{T_s}) \cdot \theta_k - (\frac{1}{T_s}) \cdot \theta_{k-1}. \tag{4.27}$$

Therefore the original state vector can also be replaced by, $\boldsymbol{x} = \begin{bmatrix} V_N & \theta \end{bmatrix}^T$ for system identification.

Based on the floating pendulum equations, it is expected that a model derived from $\dot{\theta}$ data will better approximate the actual dynamics than one using θ. This is because $\dot{\theta}$ is directly related to the dynamics, compared to θ which needs to be related to $\dot{\theta}$ to be relevant for the dynamics. However, an experiment to compare the performances of these models shows that this has a negligible effect.

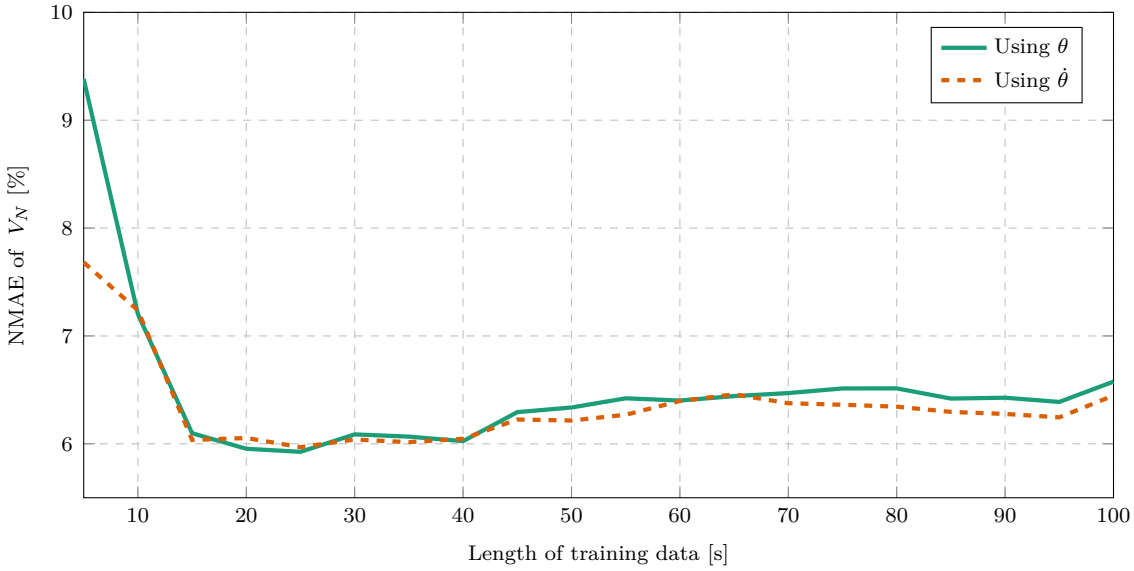


Figure 4.9: Prediction NMAE for HAVOKc models using either angle or angular rate measurements ($m_p = 0.2\,\text{kg}$, $l = 1\,\text{m}$, $T_s = 0.03\,\text{s}$).

Figure 4.9 shows the prediction error of HAVOKc models using $\dot{\theta}$ or θ for a range training data lengths. Only for very short lengths of training data, do models using $\dot{\theta}$ outperform those using θ. For longer lengths of training data there is a negligible difference in prediction error between the methods. Therefore θ will be used for system identification to

avoid unnecessary complexity, since there is no direct measurement of $\dot{\theta}$ on the practical multirotor.

4.6.6. Noise

Measurement noise is bad for system identification because it adds high-frequency information to the output signals which are not part of the actual dynamics. On the practical multirotor the Inertial Measurement Unit (IMU), barometer, magnetometer and GPS sensors experience measurement noise. The EKF performs sensor fusion and smooths out most of the measurement noise to provide a state estimate that is less noisy than raw sensor values. Therefore the output from the EKF is used for system identification.

The potentiometer and Analog to Digital Converter (ADC) which measure the payload angle on the multirotor also has experience measurement noise. This signal is not smoothed by an onboard EKF. In simulation noise is applied to the payload angle as band-limited white-noise. The applied noise power was iteratively adjusted to match that of the practical payload measurements. The noisy signals from both the multirotor EKF and payload swing angle are smoothed with a quadratic regression smoother from MATLAB® before applying system identification. The smoother uses a fixed window length of 20 samples which was selected iteratively to remove high-frequency variation without loosing the general shape of the data.

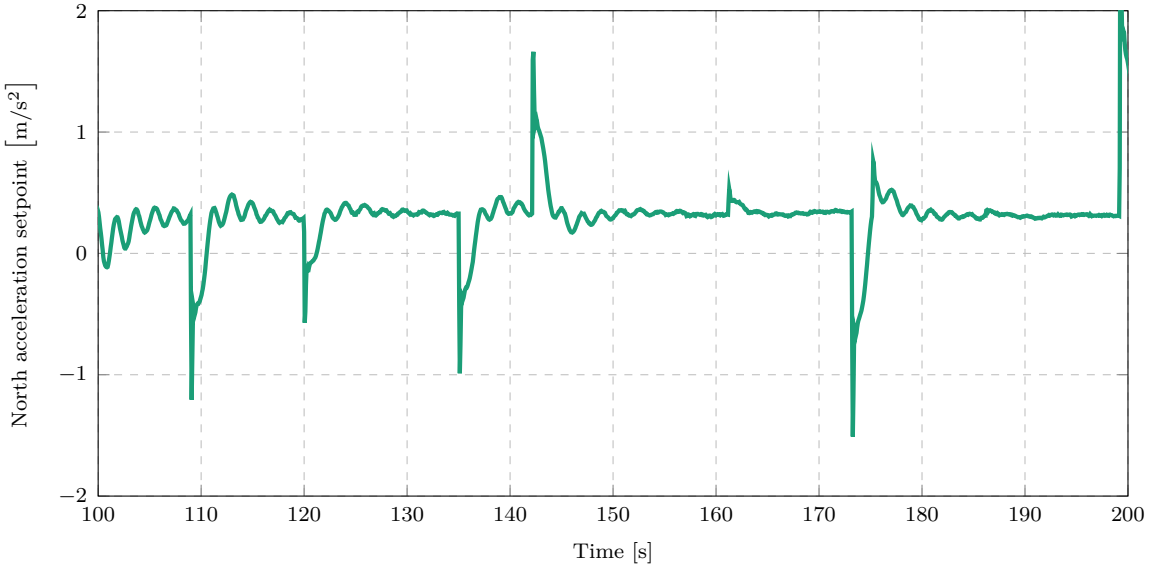


Figure 4.10: Accelleration setpoint training data from random velocity step inputs ($m_p = 0.2\,\text{kg}$, $l = 1\,\text{m}$).

The input signal also needs to be smoothed to remove high-frequency noise from the logged signal. The quadratic regression smoother does not fit the shape of the input data well

because of the sharp, non-differentiable edges in the acceleration setpoint signal. Therefore a Gaussian-weighted moving average smoother from MATLAB® is used for the input signal.

Figure 4.10 shows the North acceleration setpoint for a period of training data. Without noise the acceleration setpoint should have a zero mean, however the signal mean shows a constant offset. The is due to a measurement offset in the IMU which causes a offset in the orientation vector and therefore affects the control signals. The setpoint mean is calculated from the training data and subtracted from the signal to correct for the offset. This results in a input signal with a zero mean. The calculated mean is reapplied to the MPC control signal during implementation to readjust for the required offset.

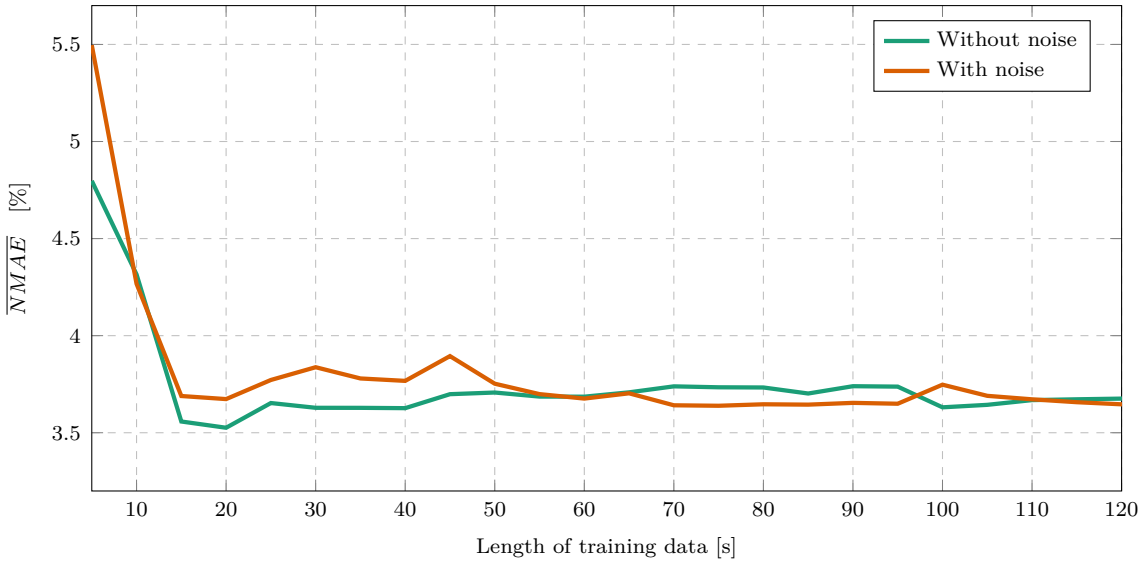


Figure 4.11: HAVOKc prediction errors for different lengths of training data with and without noise ($m_p = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).

Figure 4.11 compares the prediction errors of HAVOKc models generated from data with or without noise. The plot shows that when using short lengths of training data, the prediction errors are smaller for model generated with noiseless signals. However, it appears that the prediction errors are almost equal with longer lengths of training data. This is because with a short length of data, the signal variation or energy contributed of the noise is a significant part of the data and has a strong influence on the model. However, with longer lengths of data, the variation caused by the actual plant dynamics dominates the low energy contribution of the measurement noise. Hence, the noise has a smaller influence on the model. It also appears that at long training data lengths noise has a negligible effect on the prediction error of the resulting models.

Figure 4.12 compares the performance of HAVOKc and DMDc model when using noisy data. The prediction error curves of the two methods are very similar, with HAVOKc producing slightly lower prediction errors than DMDc. However, this difference in error

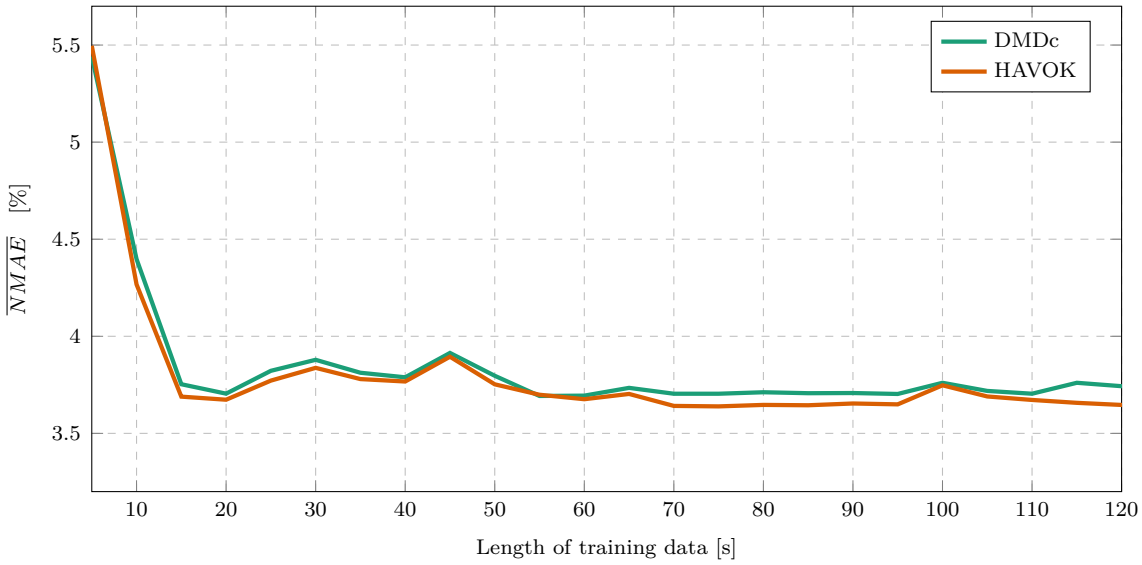
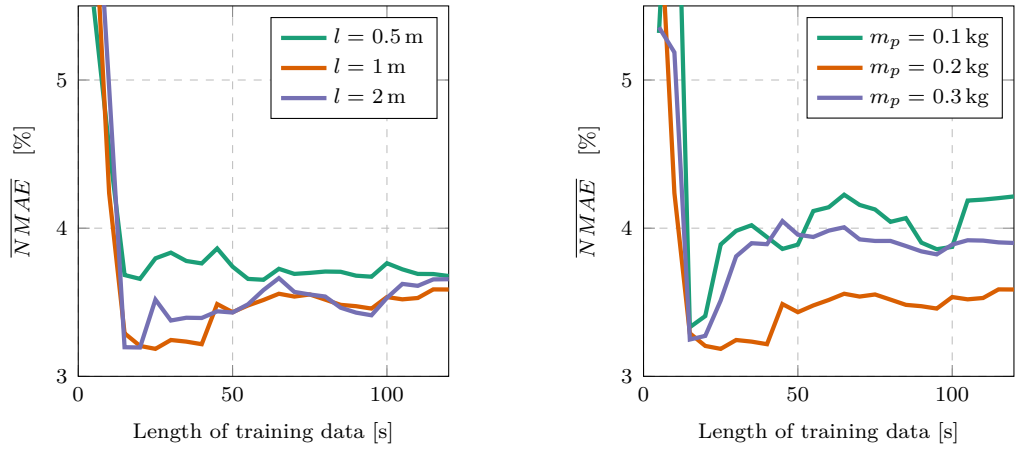


Figure 4.12: DMDc and HAVOKc prediction errors for different lengths of noisy training data ($m_p = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).



(a) $m_p = 0.2\,\text{kg}$, with varying l.

(b) $l = 1\,\text{m}$, with varying m_p.

Figure 4.13: HAVOKc prediction errors for different system parameters.

may be so small that it has a negligible effect on control.

4.6.7. System parameters

The suspended payload, as described in Section 4.2, has two system parameters, m_p and l. For the practical multirotor considered, the payload mass is limited to, $0.1 \leq m_p \leq 0.3\,\text{kg}$, and the cable length is limited to: $0.5 \leq l \leq 2\,\text{m}$. Figure 4.13 shows the prediction error of HAVOKc models build from simulations with various values of m_p and l. The plots are not shown for DMDc models because they are so similar to the HAVOKc results.

From Figure 4.13a it seems that there is not a great difference in prediction error for different cable length setups. From Figure 4.13b it appears that m_p has a greater effect on prediction error, since there is a bigger difference in prediction error between plots of

different m_p values. However, it is clear that the system identification method works for a range of different payload parameters.

4.6.8. Length of training data

From the plots discussed in previous sections, it is obvious that the accuracy of a model is dependant on the length of training data exposed to the system identification algorithm. The general relationship between length of training data and prediction error is illustrated in Figure 4.13. For very short lengths of training data the prediction error is large, but as training length increases, the prediction error improves up to a point. After this point, the prediction error slowly worsens with increasing lengths of training data.

This trend may be counter-intuitive, because it is generally expected that more training data leads to better models. The logic follows that more training data leads to less overfitting which leads to better test data predictions. However, a phenomenon named 'double-descent' occurs when the dimension of a regression model, D is near the number of training samples, N_{train} [102]. In this critical region at the transition between over-parameterized and under-parameterized models, the prediction error initially decreases, then increases to a peak whereafter it decreases again [102].

A rough calculation confirms that the critical region of 'double-descent' is applicable to our use case. The highest q in the considered range is 30, which corresponds to a model dimension of

$$D = (q \cdot n_x)^2 + (q \cdot n_x)(n_u) = 3660. \tag{4.28}$$

The length of training data corresponding to $D = N_{train} \cdot n_x$ at the transition between over- and under-parameterization is therefore:

$$T_{train} = N_{train} \cdot T_s = 54.9\,\text{s}. \tag{4.29}$$

This value is indeed within the range of considered training data lengths, which explains why our training experiments experience this phenomenon.

It should be noted that the plot for $l = 0.5\,\text{m}$ in Figure 4.13a does not follow the 'double-descent' trend. This may be because the short cable length corresponds to large swing angles and a high natural frequency. The onboard PID controllers do not damp these oscillations as quickly as the smaller angle and lower frequency oscillations of longer cables. Hence there is not enough information exposed in the first few step responses of the training data and the algorithm needs more step responses to accurately capture the steady-state behaviour of the plant.

In a practical implementations, training data is costly and it is desirable to use less training data. Less training data means less flight time will be wasted on training a model before the multirotor can fly with a updated controller. Less training data also corresponds to lower memory usage on the multirotor hardware and lower time-complexity for the algorithm. Therefore it is not practical to increase the amount of training data to the under-parameterized region. Hence, the critical region of training data lengths will be used and the data length corresponding to the lowest prediction error will be selected per simulation.

4.6.9. Dynamic payload

In Section 4.3 it was shown that the white-box system identification method perform well for the suspended payload use case. It was also shown in [6] and [5] that this method can be used in conjunction with LQR control to minimize swing angles of an unknown payload. However, many payloads do not satisfy the assumptions made in for the a priori model which is detrimental to performance of white-box method. For example, for an elongated payload attached to the cable the point-mass assumption does not hold. The CoM of the payload is well below the attachment point of the cable, which creates a double pendulum model that differs significantly from a simple pendulum. Such a payload is better represented in 2D by the double pendulum modelled in Figure 4.14, than the model defined in Figure 4.1.

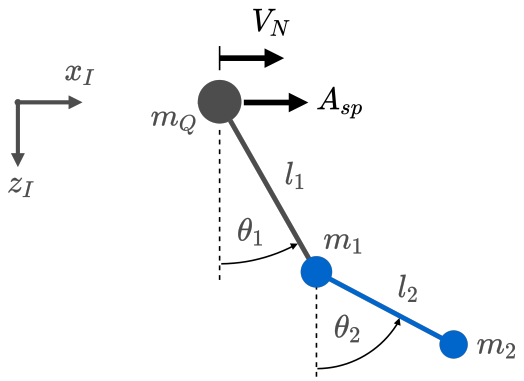


Figure 4.14: Double pendulum model representing an elongated suspended payload

Figure 4.15 shows the k-step-ahead prediction of a white-box models for a single pendulum simulation. The exact m_p is used for the model, but the cable length is estimated from a FFT of the payload swing angle as described in Section 4.3.3. It is clear that the prediction is accurate for the first few oscillations, but the slight difference in frequency accumulates over time causing an increasingly large offset. However, the general shape of the dynamics is still captured by the model. Also note that the prediction oscillations are damped

linearly but the actual oscillations experience non-linear damping. This difference is an approximation error because the non-linear plant is modelled with a linear model.

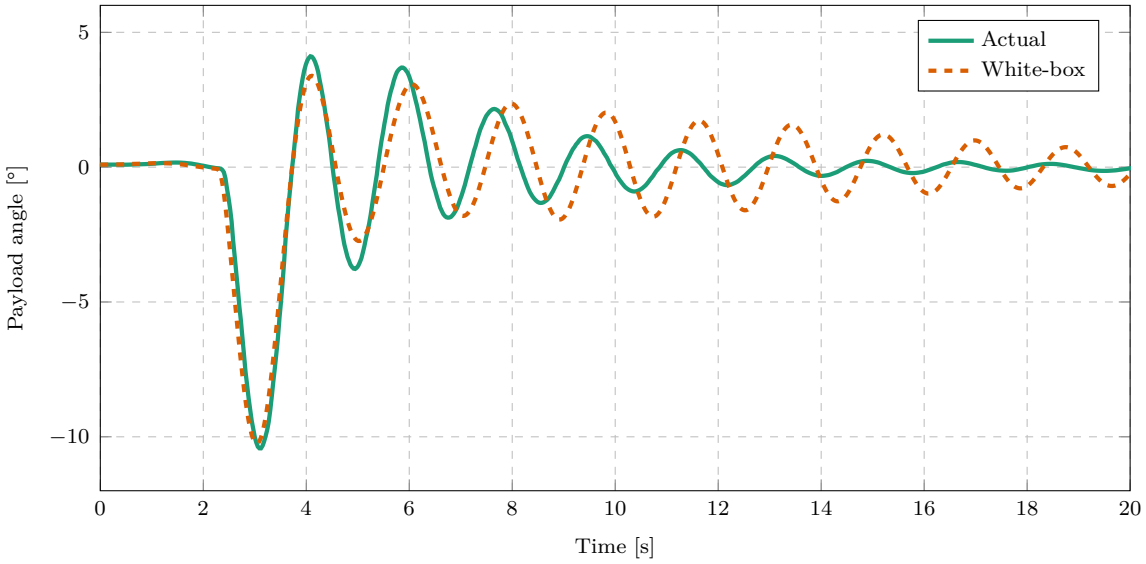


Figure 4.15: White-box model predictions of a single pendulum for a North velocity step input ($l = 1\,\text{m}$, $m_p = 0.3\,\text{kg}$.).

Figure 4.16 also shows the k-step-ahead prediction of a white-box model, but this prediction is for a double pendulum simulation. The cable length was estimated from the same method by calculating the FFT of the cable swing angle and identifying the dominant frequency. Note how the prediction of the first oscillation is quite accurate and is similar to the initial swing of the single pendulum. However, by the second swing, the double pendulum dynamics differ significantly from the model prediction. The single pendulum oscillations seen in Figure 4.15 are regular compared to double pendulum oscillations in Figure 4.16 which are noticeably irregular. The a priori model expects regular, single frequency oscillations. This can be seen in Figure 4.16 where the predicted peaks are equidistant. However, the actual dynamics have a superposition of two frequencies due to the double pendulum payload.

The FFT amplitude spectrum of the single pendulum is shown in Figure 4.17b. This plot shows a single peak which corresponds to the natural frequency of the suspended payload. Figure 4.17a shows the FFT amplitude spectrum of the double pendulum. Two peaks are revealed in this plot which correspond to the two superimposed frequencies caused by the double pendulum. The frequency content of the two plants are clearly different so the white-box model and parameter estimation techniques would need to be redesigned for these payloads specifically. This is the great disadvantage of the white-box system identification technique. For every model with different dynamics, a new technique needs to be designed and used. In contrast, the proposed data-driven method provides a general

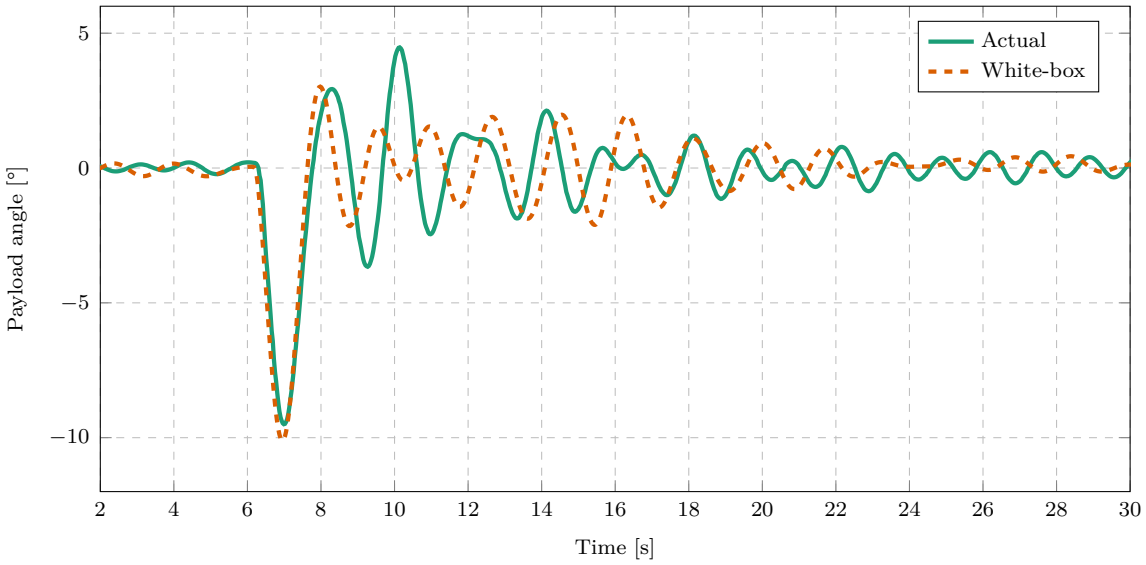


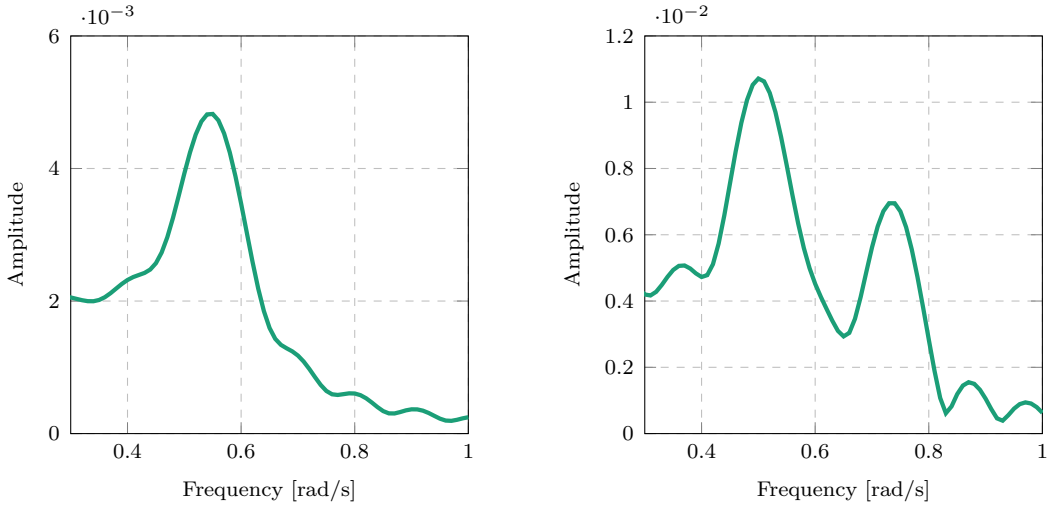
Figure 4.16: White-box model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).

solution for a large range of different payloads and dynamics.

Figure 4.18 shows the prediction of the two data-driven methods for a single pendulum simulation. Note that there is less of a frequency difference in this prediction than the white-box prediction in Figure 4.15. The shape of the predicted damping is also more similar to the actual dynamics than the white-box prediction. This may be because the data-driven methods effectively fit a higher order damping model to the dynamics, compared to the simple, linear damping applied in the white-box model.

The damping seen in these plots is a complicated effect which depends on the payload connection, the aerodynamic drag, and the controller gains. An advantage of data-driven system identification techniques is that the effect of damping is inherently included in the estimated model without specifically estimating a damping coefficient. In contrast, the white-box estimation technique requires the effect to be modelled a priori and a designed algorithm to estimate every parameter that effects the dynamics.

Figure 4.19 shows the prediction of the same data-driven methods but for a double pendulum simulation. The same cable length and effective payload mass is used here as in the single pendulum simulation. Notice how accurate the prediction is for the first 20 s of the plot. In contrast to the white-box model, the black-box model oscillations follow the irregular, multi-frequency response of the actual dynamics. The state-space model can approximate the multi-frequency dynamics of the plant because of the delay-coordinates in the model. As expected, the prediction accuracy is also much better than the white-box models for both the single and double pendulum simulations.



(a) Single pendulum. **(b)** Double pendulum.

Figure 4.17: The single-sided amplitude spectrum of the swing angle FFT.

The double pendulum plant involves a hidden state variable, because the unmeasured angle of the second pendulum is required to fully describe the state of the system. However, the DMDc and HAVOKc models are still able to approximate the dynamics quite accurately without measuring this state variable. This is due to the extent of the delay embedding of the models [93]. Figure 4.20 shows the prediction error as a function of the number of delays in the model for a double pendulum. Note how much more delays is required before the prediction error reaches steady-state than for a single pendulum showed in Figure 4.6. This is because the dynamics are more complex and model needs a lot more parameters to account for the hidden state variable.

Overall, the data-driven system identification approaches were shown to work well for both the single and double pendulum payloads. In contrast, the white-box method describes the general shape of the single pendulum dynamics well, but do not perform well for a double pendulum simulation because it was specifically designed for the single pendulum payload. The data-driven approaches therefore provide an accurate system identification method for a much larger range of payload types without needing a redesign for specific payload dynamics.

4.7. Summary

DMDc and HAVOKc produce very similar prediction errors for a range of different simulation conditions. HAVOKc generally has slightly lower errors, but this may have a negligible effect on control. DMDc will be used in the remainder of this work because the algorithm has been studied more, is less computationally complex and provides similar performance to HAVOKc. These data-driven methods were shown to produce accurate

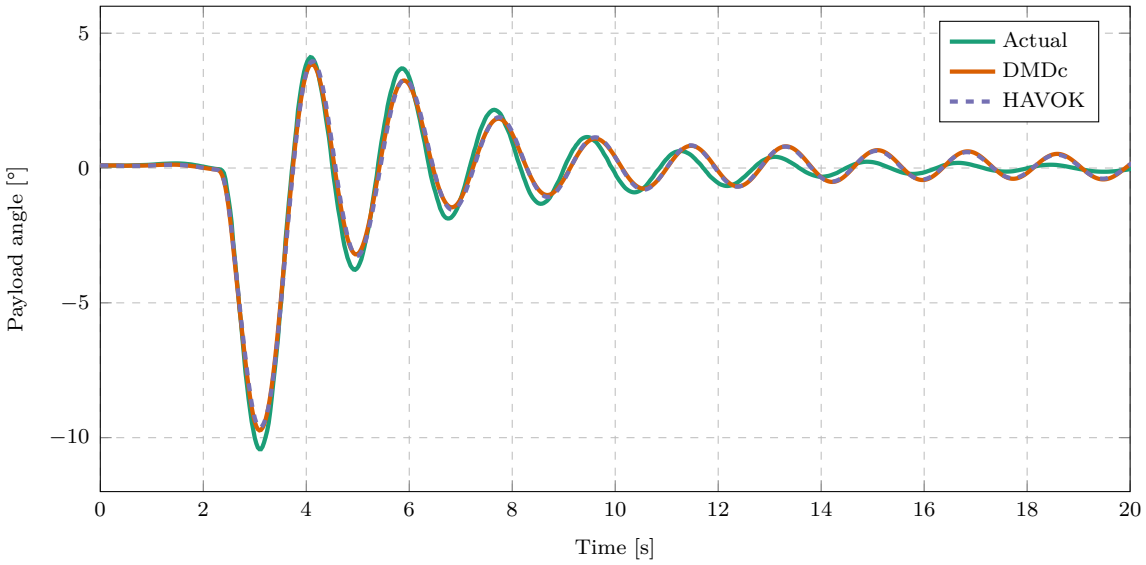


Figure 4.18: Data-driven model predictions of a single pendulum for a North velocity step input ($m_p = 0.3\,\text{kg}$, $l = 1\,\text{m}$).

predictions of the payload dynamics with a practical length of training data, sample time and noise level. This proves to be promising for practical implementation with an MPC on a multirotor OBC. The data-driven methods also proved to provide accurate prediction with a range of different payload parameters and even with payloads that act as a double pendulum. In contrast, the considered white-box system identification technique failed to identify a relevant model for the double pendulum dynamics. Therefore the data-driven approach provides a more general system identification method that can be used for a range of different suspended payloads without being redesigned for specific dynamics.

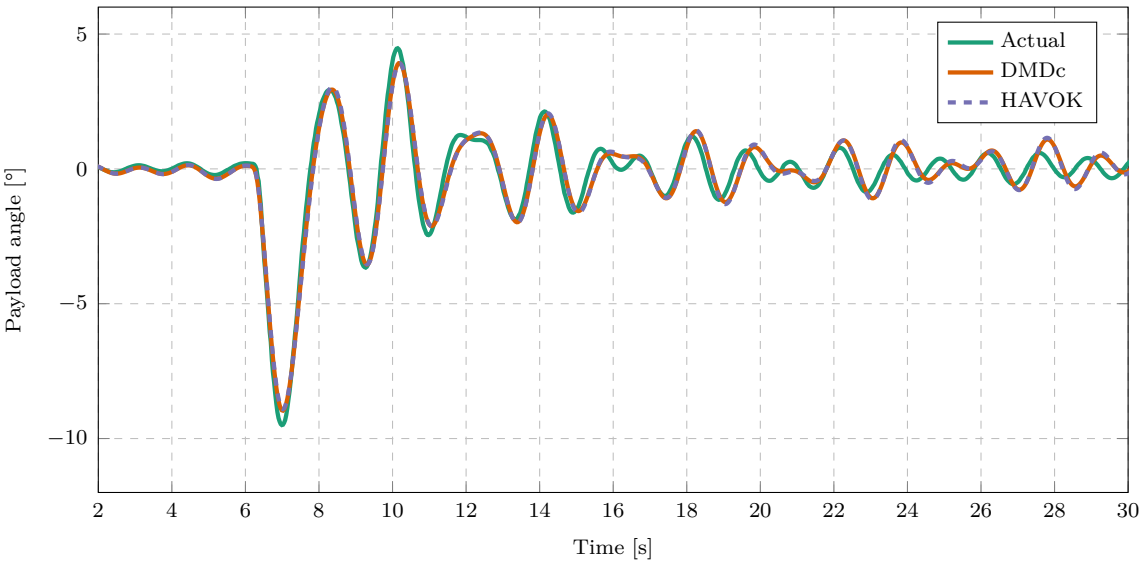


Figure 4.19: Data-driven model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).

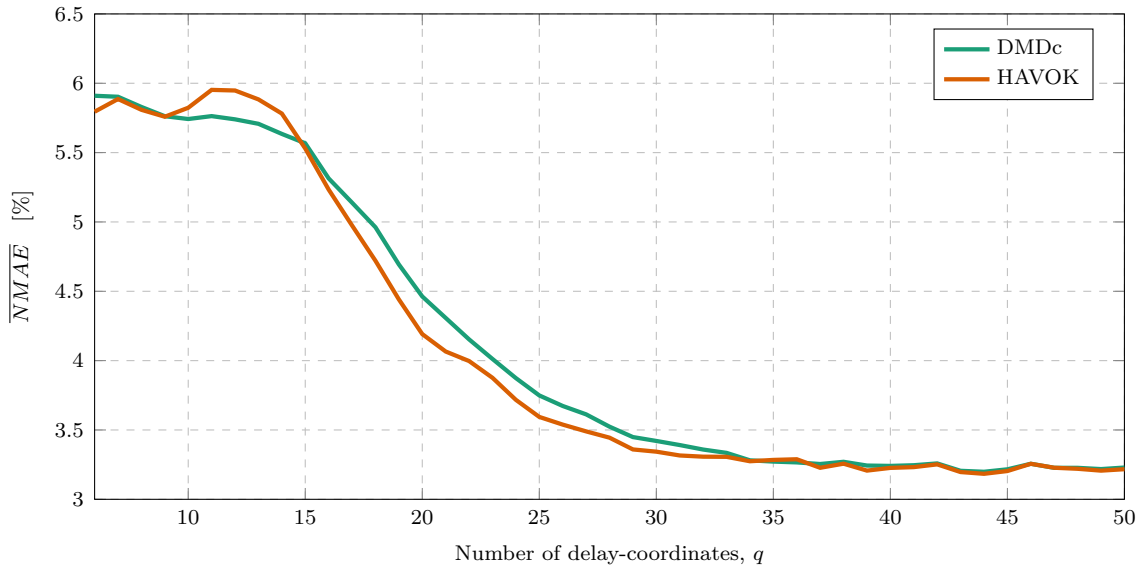


Figure 4.20: DMDc and HAVOKc predictions error of double pendulum for different numbers of delay-coordinates ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$ $T_{train} = 70\,\text{s}$.).

Chapter 5

Control systems

Three different types of controllers are considered in this work, namely PID, LQR, and MPC. PID control does not provide active swing-damping control of the payload, but it is used in the training data stage discussed in Chapter 4. LQR is a popular and well-known optimal control technique. In previous work by [5] and [6], an LQR implementation with a parameter estimator was designed for active swing-damping control of a multirotor with an unknown suspended payload. This LQR architecture was shown to be an effective swing-damping controller and will be applied in this work as the baseline controller.

A data-driven system identification method was introduced in Chapter 4 to estimate a linear model of unknown dynamics. The models generated by this method will be used in an MPC for active swing-damping control of the multirotor and payload system. The proposed MPC architecture will be compared to the baseline LQR controller. These swing-damping control architectures are summarised in Table 5.1, where each controller is paired with a system identification method.

Table 5.1: System identification techniques paired with the corresponding controllers.

System identification		**Controller**
White-box model	RLS mass estimator, and FFT cable length estimator	LQR
Black-box model	DMDc, or HAVOKc	MPC

In this chapter, a Simulink™ simulation environment will be introduced to test the proposed control architectures. An overview of the different controllers will be given and the design process of each controller will be explained. The controllers will then be tested in simulations with different system parameters and disturbances. Finally, the simulation results will be shown and discussed.

5.1. Simulation Environment

The controllers in this chapter are tested within a Simulink™ simulation environment. This is used rather than the SITL and Gazebo simulation environment because it allows us to iterate designs faster. Simulink™ provides a graphical interface for control system design which enables us to change the control system design with ease. In contrast, the SITL and Gazebo simulation environment requires text-based control laws with C++ and requires ROS nodes to interface an MPC with PX4. This requires a lot more development time for control system design than the graphical tools in Simulink. The SITL and Gazebo simulation also has a longer runtime per simulation, which further adds to the development time of the iterative control system design process.

The multirotor and suspended payload system is modelled in Simulink™ with the differential equations derived in Chapter 3. The resulting controllers are also applied in Simulink. Using the cascade PID control architecture (discussed in the sections below), this simulation environment was verified against practical data with and without a payload. The plots in Figure 3.4, 3.5, and 3.6 from Chapter 3 show how well the simulations match the actual system dynamics. The controllers will therefore be designed, tested and compared using this simulation environment.

5.2. Cascaded PID control

PID control is a popular linear control technique that applies a control signal proportional to the error signal, the integral of the error signal, and the derivative of the error signal. This is the default control architecture used in the PX4-Autopilot flight stack [94]. PX4-Autopilot was chosen as the multirotor flight-stack because it is open-source and widely used in industry and research. The PID implementation of PX4 does not provide active swing-damping of the payload, however, it is used in the system identification flight stage discussed in Chapter 4.

The default PX4 control architecture consists of multiple cascaded PID controller loops. This is divided into two main sections, the inner-loop attitude controllers and the outer-loop translational controllers. Figure 5.1 shows a high-level overview of the PX4 control architecture without showing state feedback.

The setpoint vectors in Figure 5.1 are denoted by $\boldsymbol{X}_{sp}$ for position, $\boldsymbol{V}_{sp}$ for velocity, $\boldsymbol{A}_{sp}$ for acceleration, $\boldsymbol{q}_{sp}$ for the attitude quaternion, $\boldsymbol{\Omega}_{sp}$ for angular rate, and $\boldsymbol{T}_{sp}$ for motor thrust. The virtual actuator commands are denoted by $\delta_{A_{sp}}$, $\delta_{E_{sp}}$, $\delta_{R_{sp}}$, and $\delta_{T_{sp}}$, for the virtual aileron, elevator, rudder, and thrust commands.

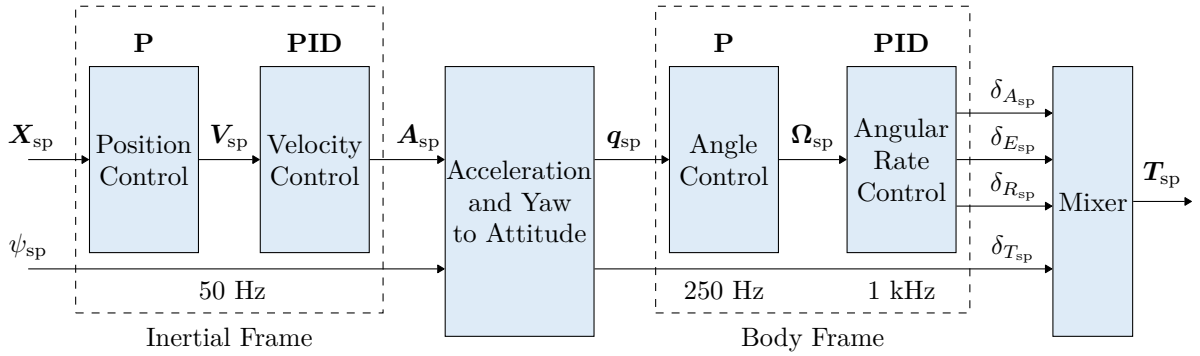


Figure 5.1: Cascaded PID control architecture of PX4 [7]

The inner-most control element is the mixer, which converts virtual actuator commands to actuator thrust commands. The attitude controller includes the angle and angular rate controllers, which send commands to the mixer. The translational controller consists of the position and velocity controllers, which send commands to the attitude controller. The rate of each controller is also shown in Figure 5.1. Note that the outer-loop controllers deal with slow dynamics and therefore run at slower rates than the inner-loop controllers.

5.2.1. Angular rate controller

Three separate linear PID controllers are used to control angular rates in the pitch, roll, and yaw axes of the multirotor body frame. The angular rate controller receives angular rate estimates from the PX4 state estimator and outputs virtual actuator commands to the mixer. Figure 5.2 shows a diagram of the pitch angular rate controller. The yaw and roll controllers are not shown here, but they replicate the same structure.

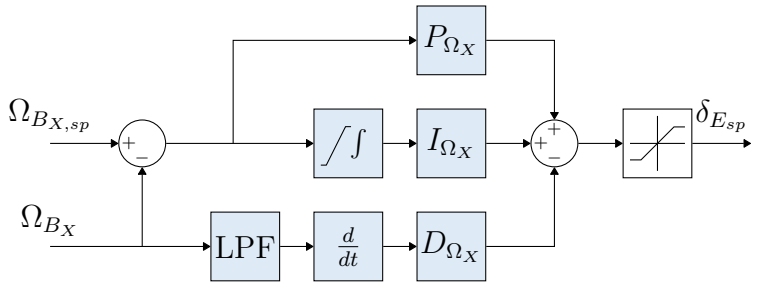


Figure 5.2: Angular rate controller diagram [7].

In Figure 5.2, P_{Ω_X}, I_{Ω_X}, and D_{Ω_X} are the PID gains for the pitch angular rate controller. As defined in Chapter 3, Ω_{B_X} is the pitch angular rate in the body axis, $\Omega_{B_{X,sp}}$ is the pitch angular rate setpoint, and $\delta_{E_{sp}}$ is the virtual elevator setpoint.

Some controller elements improve the practical implementation, but have a negligible effect on the design of the gains, namely:

- A Low Pass Filter (LPF) is added in the derivative path to reduce the effect of sensor noise.
- The derivative path is implemented on the plant output signal, instead of the error signal to eliminate derivative kick.
- Saturation is applied to the integral path to eliminate integral wind-up.
- The control signal is saturated to avoid dangerously large setpoint commands.

These effects are also described in detail by [5].

Classical control theory was used by [10] to design the controller gains of the practical multirotor, Honeybee. This same process is also explained in detail by [5]. The controller gains were designed for a transient response that is as fast as possible while retaining fast disturbance rejection, and minimal overshoot. It was determined by [10] that the default PX4 angular rate controller gains of the ZMR250 airframe provide excellent control for Honeybee and satisfy these design requirements. These gains are documented in Appendix A.

For a 1 rad/s step response, the pitch rate controllers result in a 3.6 % overshoot, 0.024 s rise time, 0.8 s 2 % settling time, and 138 rad/s bandwidth [10]. These gains are well suited for Honeybee and will also be implemented in this work. The default roll-rate and yaw-rate controller gains are also implemented for the same reason.

5.2.2. Angle controller

The pitch, roll, and yaw angles are controlled by the angle controller in the body frame. A quaternion based controller is implemented by PX4 for angle control based on work by [80]. The structure and design of this controller is well explained by [10], [5], and [6] and is only briefly discussed here.

For the control law proposed by [80], the error quaternion is calculated as:

$$\boldsymbol{q}_e = \boldsymbol{q}^{-1} \cdot \boldsymbol{q}_{sp}, \tag{5.1}$$

where $\boldsymbol{q}_e$ is the quaternion error, $\boldsymbol{q}$ is the current attitude quaternion of the multirotor, and $\boldsymbol{q}_{sp}$ is the attitude quaternion setpoint. The resulting control law is given as:

$$\boldsymbol{\Omega}_{sp} = 2\boldsymbol{P}_q \operatorname{sgn}(q_{0_e})\boldsymbol{q}_{v_e}, \qquad \operatorname{sgn}(q_{0_e}) = \begin{cases} 1, & q_{0_e} \geq 0 \\ -1, & q_{0_e} < 0 \end{cases}, \tag{5.2}$$

where $\boldsymbol{\Omega}_{sp}$ is a vector of the roll, pitch, and yaw angular rate setpoints, $\boldsymbol{P}_q$ is a vector of the corresponding proportional gains, q_{0_e} is the error of the quaternion magnitude component, and $\boldsymbol{q}_{v_e}$ is the error of the vector component of the quaternion. The implementation of this control law is illustrated in Figure 5.3 for a single element of $\boldsymbol{\Omega}_{sp}$.

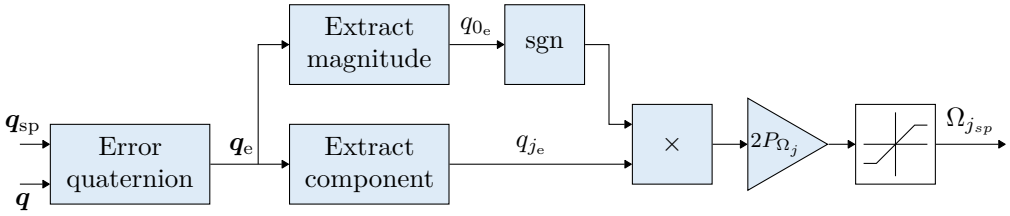


Figure 5.3: Quaternion based angle controller diagram [7].

In Figure 5.3, the j subscript denotes a specific element in the vectors $\boldsymbol{q}_e$, $\boldsymbol{P}_q$, and $\boldsymbol{\Omega}_{sp}$, where $j = \{1, 2, 3\}$. The attitude quaternion setpoint is taken as input, the current attitude is received from the PX4 estimator, and the resulting angular rate setpoint is produced as the control signal.

Only a proportional term is applied in this control law. The proportional gains were designed for Honeybee by [10] for a transient response that is faster than the default ZMR250 response. These gains are documented in Appendix A. This results in a 1 rad step response with little to no overshoot, 0.3 s rise time, 0.47 s for a 2 % settling time, and 11 rad/s bandwidth [10]. This has a large time-scale separation from the angular rate controller, which has a bandwidth of 138 rad/s. A fast angle response is also desired in this work, hence the same gains designed by [10] will be used.

5.2.3. Translational controller

The translational controller consists of the position and velocity controllers. Position control is not required for training data in this work, therefore it will not be discussed in this section. A diagram of the North velocity controller is shown in Figure 5.4. The West and Down velocity controllers duplicate the same configuration.

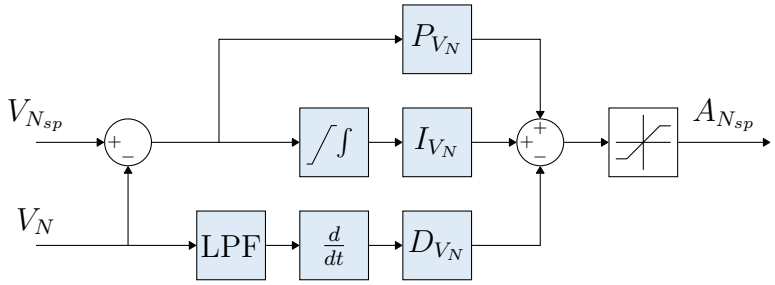


Figure 5.4: Velocity controller diagram [7]

In order to drive the multirotor to a given velocity setpoint, the velocity controller commands an acceleration setpoint in the inertial frame. This acceleration setpoint is transformed to an attitude setpoint which is used by the angle controller. This transformation is based on work done by [103].

Recall from Chapter 4, that the cascaded PID controller is used for velocity step inputs in the training data flight stage for system identification. The system identification methods produce linear models, which are used in swing damping controllers to minimise the payload angles during flight. The velocity controller gains used for Honeybee by [10] result in aggressive velocity responses, which produces large payload swing angles. Such large payload angles are undesirable for safe flights. Therefore the velocity controller gains are redesigned for a slower transient response and smaller payload swing angles.

The derivation of the transfer function, $G_{V_N}(s)$, of the North velocity dynamics of a multirotor with a suspended payload was derived by [6] and is described here briefly. Firstly, the non-linear dynamics were derived with Lagrangian mechanics as described in Chapter 4. The equations were then linearised around hover with the small-angle approximation. The linearised equations for the longitudinal or North velocity dynamics can be presented in the state-space form as,

$$\dot{\mathbf{X}}_{long} = \mathbf{A}_{long}\mathbf{X}_{long} + \mathbf{B}_{long}\mathrm{U}_{long} \text{ and} \tag{5.3}$$

$$\mathrm{Y}_{long} = \mathbf{C}_{long}\mathbf{X}_{long}, \tag{5.4}$$

where

$$\mathbf{X}_{long} = \begin{bmatrix} V_N & \dot{\theta} & \theta \end{bmatrix}^T, \tag{5.5}$$

$$U_{long} = A_N, \tag{5.6}$$

$$\mathbf{A}_{long} = \begin{bmatrix} 0 & \frac{c}{lm_Q} & \frac{gm_p}{m_Q} \\ 0 & -\frac{c(m_Q+m_p)}{l^2 m_Q m_p} & -\frac{g(m_Q+m_p)}{lm_Q} \\ 0 & 1 & 0 \end{bmatrix}^T, \tag{5.7}$$

$$\mathbf{B}_{long} = \begin{bmatrix} 1 & -\frac{1}{l} \end{bmatrix}^T, \tag{5.8}$$

$$\mathbf{C}_{long} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}. \tag{5.9}$$

For the definition of these symbols, refer to Chapter 4. The input and output of the plant are $\mathrm{U}_{long} = A_N$ and $\mathrm{Y}_{long} = V_N$, respectively. Note that the actual input of the plant is $A_{N_{sp}}$, however, it is assumed that $A_{N_{sp}} \approx A_N$ due to the large time-scale separation between the velocity controller and attitude controller. The transfer function can therefore

be calculated as,

$$G_{V_N}(s) = \frac{V_N(s)}{A_N(s)} = \mathbf{C}_{long}\left(s\mathbf{I} - \mathbf{A}_{long}\right)^{-1}\mathbf{B}_{long} \tag{5.10}$$

$$G_{V_N}(s) = \frac{s^2 + \frac{c}{m_p l^2}s + \frac{g}{l}}{s\left[s^2 + \frac{c(m_Q + m_p)}{m_Q m_p l^2}s + \frac{g(m_Q + m_p)}{m_Q l}\right]} \tag{5.11}$$

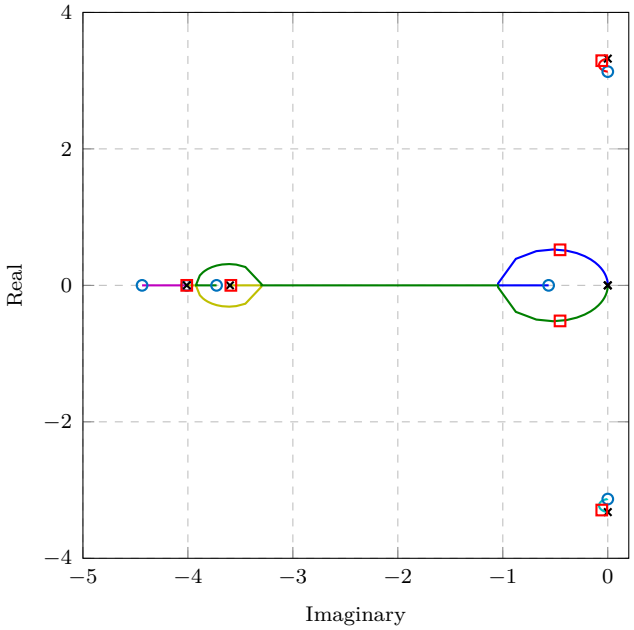
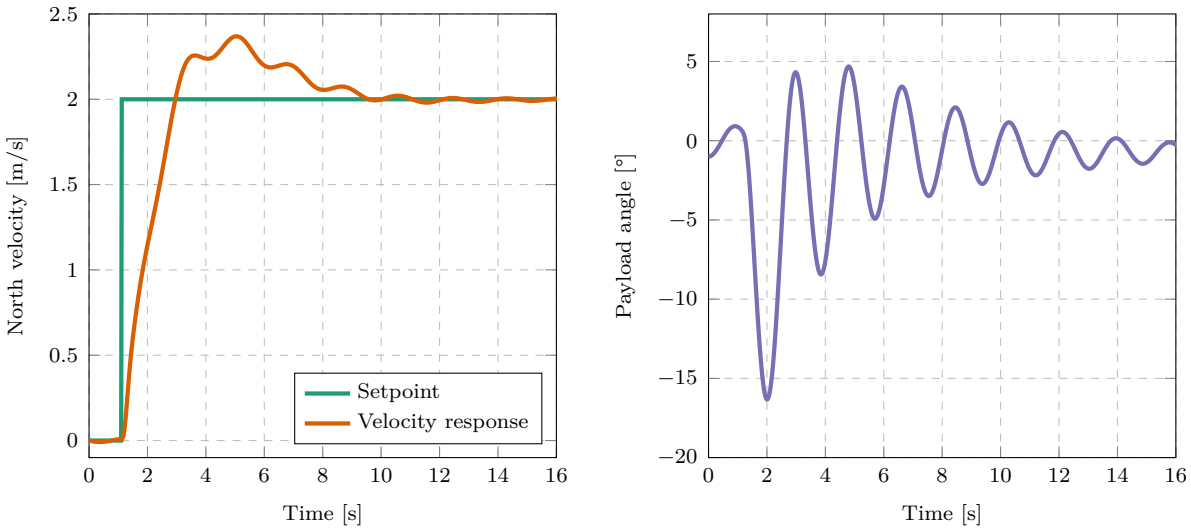


Figure 5.5: Root locus plot of the North velocity dynamics including PID controller.

A payload with $m_p = 0.2\,\text{kg}$ and $l = 1\,\text{m}$ is considered for the controller design. Figure 5.5 shows a root locus plot of the closed loop system which includes the PID controller. The gains were tuned in an iterative process to produce a slow step response that stimulates the payload dynamics enough for system identification. These gains were also tested iteratively with different payload parameters in the ranges, $0.1 \leq m_p \leq 0.3$ kg and $0.5 \leq l \leq 2$ m, to establish safe flights with different payloads.

Figure 5.6 shows a velocity step response for the resulting cascaded PID controller with a specific payload. The gains used are documented in Appendix A. Note that the response results in a large overshoot, however, this aids in keeping the payload angles low. The initial swing angle is quite large, however, this attenuates to lower angles quickly enough for stable flight. The velocity oscillations are clearly visible in Figure 5.6a. The current PID controller does not provide an adequate way of actively damping the payload oscillations, hence an active swing damping controller is required.



(a) North velocity. **(b)** Payload angle.

Figure 5.6: PID velocity step response ($l = 1\,\text{m}$, $m_p = 0.2\,\text{kg}$)

5.3. Linear Quadratic Regulator

LQR is a popular optimal control technique and is an effective swing-damping controller for multirotors with a suspended payload [5, 6, 104]. The LQG technique combines LQR with a full-state estimator to control systems where some state variables cannot be measured [105]. An LQG was effectively used by [6] for swing-damping control of a multirotor and suspended payload system.

However, to focus on the controller performance without considering the effect of state estimation, it is assumed that full-state feedback is available in this work. Therefore, an LQR for North velocity control with full-state feedback is presented in this section. This controller structure can be duplicated for East velocity control due to the symmetry of a multirotor.

An LQR does not inherently apply integral action and does not achieve zero steady-state tracking error in the presence of disturbances. Therefore the state vector is augmented with an integral state, $\mathcal{V}_N$, of the velocity error such that:

$$\dot{\mathcal{V}}_N = V_{N_{sp}} - V_N \tag{5.12}$$

The North velocity dynamics were derived and linearised in Chapter 3 to produce a state-space model which will be used to design the LQR. It is assumed that all model parameters are known before a flight, except for the payload parameters, l and m_p, which are estimated in the system identification phase. The augmented state space model which

includes the integral state is given by:

$$\dot{\boldsymbol{x}}_A = \boldsymbol{A}_A \boldsymbol{x}_A + \boldsymbol{B}_A \boldsymbol{u}_A, \tag{5.13}$$

where

$$\boldsymbol{x}_A = \begin{bmatrix} \mathcal{V}_N & V_N & \theta & \dot{\theta} \end{bmatrix}^T, \tag{5.14}$$

$$\boldsymbol{u}_A = \begin{bmatrix} A_{N_{sp}} \end{bmatrix}, \tag{5.15}$$

$$\boldsymbol{A}_A = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & \frac{m_p \cdot g}{m_Q} & \frac{c}{(l \cdot m_Q)} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(m_p + m_Q) \cdot g}{(m_Q \cdot l)} & \frac{-c \cdot (m_p + m_Q)}{(l^2 \cdot m_Q \cdot m_p)} \end{bmatrix}, \text{ and} \tag{5.16}$$

$$\boldsymbol{B}_A = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \frac{-1}{l} \end{bmatrix}. \tag{5.17}$$

The LQR control law is defined as:

$$\boldsymbol{u}_A = \boldsymbol{K}_{lqr}(\boldsymbol{x}_{A_{sp}} - \boldsymbol{x}_A), \tag{5.18}$$

where $\boldsymbol{K}_{lqr}$ is the LQR gain, and $\boldsymbol{x}_{A_{sp}}$ is the augmented state vector setpoint. Only $V_{N_{sp}}$ has a non-zero value, hence $\boldsymbol{x}_{A_{sp}} = \begin{bmatrix} 0 & V_{N_{sp}} & 0 & 0 \end{bmatrix}^T$.

Furthermore, the LQR considers the cost function:

$$J(\boldsymbol{u}_A) = \int_0^\infty \left(\boldsymbol{X}_A^T \boldsymbol{Q} \boldsymbol{X}_A + \boldsymbol{u}_A^T \boldsymbol{R} \boldsymbol{u}_A \right) dt, \tag{5.19}$$

where $\boldsymbol{Q}$ is the state weighting matrix, and $\boldsymbol{R}$ is the input weighting matrix. The LQR gain, $\boldsymbol{K}_{lqr}$, is therefore determined by substituting Equation 5.18 into Equation 5.19 and minimising the cost function.

The LQR control performance can be manually tuned by changing the state and input weights in $\boldsymbol{Q}$ and $\boldsymbol{R}$ respectively. The weighting of each state variable can be determined according to tracking importance. For example, if the weighting of V_N is small in comparison to the weighting of θ, the controller will produce a slow velocity response with small payload swing angles. For the values in $\boldsymbol{R}$, if the weighting of an input variable is large, the controller will output lower control values for that variable.

The general design requirements of the LQR are to produce a fast velocity response with zero steady-state error while damping the payload oscillations quickly. The priority is to produce a smooth trajectory with minimal oscillation, however, a reasonably fast response time is still required. An iterative tuning approach was used to determine the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrix entries that produce a good performance. The final LQR weightings were selected as:

$$\boldsymbol{Q} = diag([0.1 \quad 10 \quad 0 \quad 100]), \quad \boldsymbol{R} = 13. \tag{5.20}$$

The term $diag()$ refers to a diagonal matrix with the given vector as the principal diagonal. Note that the weight of the payload angle variable is 0. This is because the payload angle will have a non-zero steady-state value at a constant velocity, V_N, due to aerodynamic drag. The payload angle is therefore damped by placing a heavy weighting on the derivative of the payload angle instead.

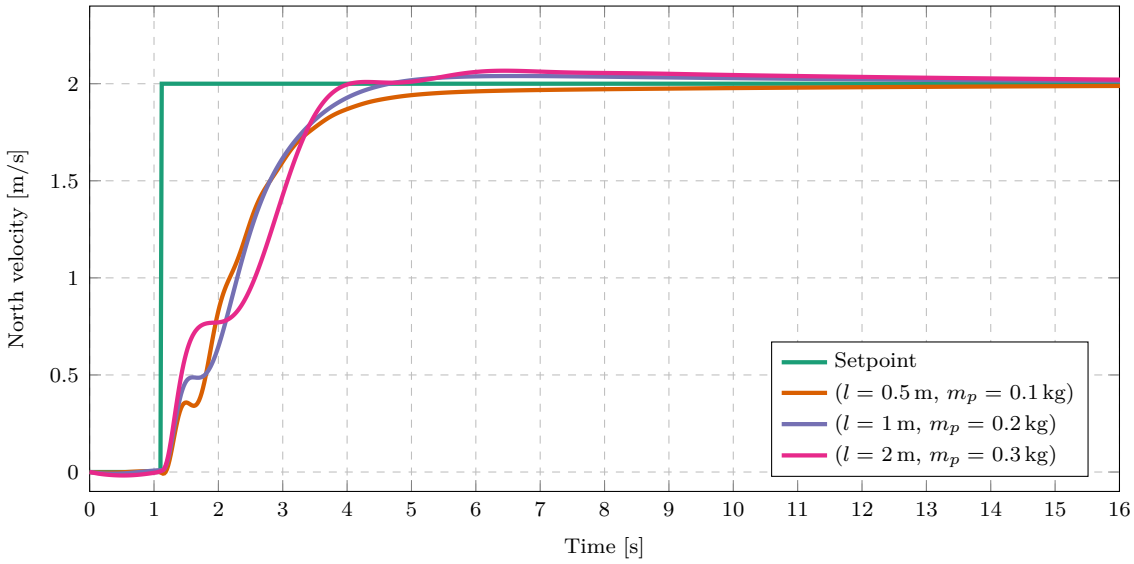


Figure 5.7: LQR velocity step responses with different payloads.

Figure 5.7 shows a plot of the simulated LQR velocity response with different payload parameters. It is clear that the controller damps the payload angles well and produces a smooth trajectory with different payloads. For each different payload, the payload parameters are estimated, the state-space model is populated, and the LQR gain is calculated using the same weighting matrix values. The controller performance, including disturbance rejection, will be further discussed in Section 5.5.

5.4. Model Predictive Control

Model Predictive Control (MPC) refers to a control system approach that determines the control action at each time-step by solving an open-loop optimal control problem over a

finite prediction horizon [106]. MPC does not refer to a specific algorithm implementation, but rather to the general control system approach.

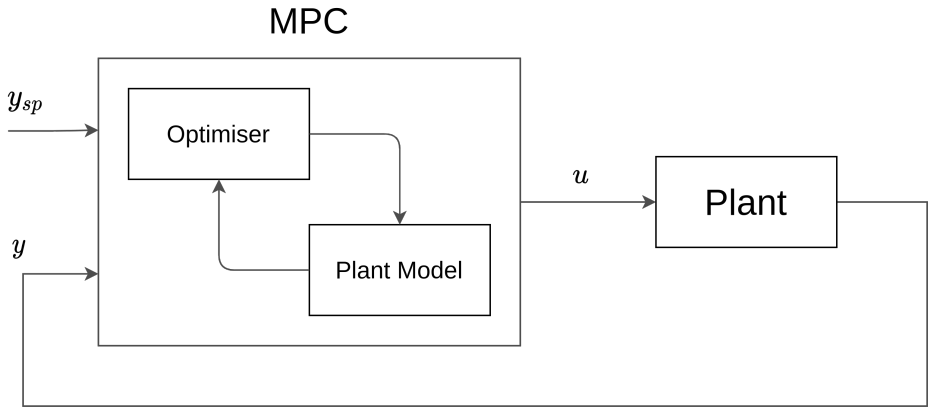


Figure 5.8: Diagram of the structure of a typical MPC

Figure 5.8 shows the structure of a typical MPC implementation. As a high-level overview, the MPC receives the measured output vector, $\boldsymbol{y}$, and the output setpoint $\boldsymbol{y}_{sp}$, and determines the control action, $\boldsymbol{u}$, to drive the values in $\boldsymbol{y}$ to the values $\boldsymbol{y}_{sp}$. An optimiser uses an internal plant model to determine an optimal control sequence over a prediction horizon [106]. Therefore $\boldsymbol{y}_{sp}$ may be replaced by a target trajectory for the prediction horizon. In many implementations, only the first control action of the optimal sequence is executed and the optimisation is re-calculated at every time-step.

Each specific MPC implementation is dependant on the plant model representation [107]. In this section, an overview will be given of the specific MPC implementation used in this work. The MPC objective function will be explained and the design process to tune the controller will be discussed. It will also be discussed how integral action is achieved with the MPC. Finally, the control response of the tuned MPC will be shown and discussed for the simulated system.

5.4.1. Receding horizon

As stated before, an MPC considers an open-loop optimal control problem over a finite prediction horizon [106]. Using a plant model for predictions, an optimiser determines the optimal control sequence that will minimise an objective function over the prediction horizon. MPC is also referred to as receding horizon control because the control sequence is determined for the next prediction horizon period at every time-step.

Figure 5.9 illustrates the receding horizon concept for a Single Input Single Output (SISO) system. One of the main objectives of the optimiser is to minimise the error between

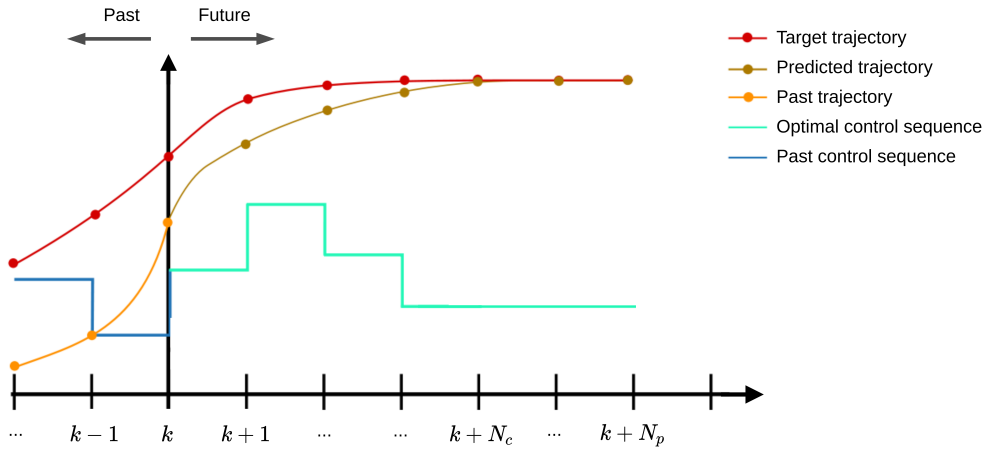


Figure 5.9: Illustration of the receding horizon of an MPC [8]

the predicted trajectory and the target trajectory. Starting at time-step k, a prediction horizon of N_p time-steps is considered, and the optimiser suggests a controller decision of N_c unique control values. N_c is referred to as the control horizon and is subject to the condition, $N_c \leq N_p$. The control sequence, or controller decision, at time-step k is denoted by,

$$\boldsymbol{z}_k{}^T = [\boldsymbol{u}(k|k)^T \quad \boldsymbol{u}(k+1|k)^T \quad \ldots \quad \boldsymbol{u}(k+p-1|k)^T], \tag{5.21}$$

which minimizes a specific objective function over the prediction horizon. The controller decision produces the predicted trajectory when applied to the plant model. Note that if $N_c < N_p$, the remaining control actions are set to,

$$\boldsymbol{u}(i|k) = \boldsymbol{u}(N_c|k), \quad i > N_c. \tag{5.22}$$

5.4.2. Plant model

An important characteristic of MPC is that it uses a separately identifiable plant model in the control optimisation process [107]. An estimated model from the data-driven techniques discussed in Chapter 4 will be used as the plant model for the proposed MPC architecture. DMDc produces a discrete, linear state-space model of the system dynamics. Hence, an MPC implementation that corresponds to such a model will be applied. The following state-space model representation will be used,

$$\boldsymbol{x}_{mpc}(k+1) = \boldsymbol{A}_{mpc}\boldsymbol{x}_{mpc}(k) + \boldsymbol{B}_{mpc}\boldsymbol{u}_{mpc}(k), \tag{5.23}$$

where $\boldsymbol{A}_{mpc}$ is the system matrix and $\boldsymbol{B}_{mpc}$ is the input matrix applied by the MPC. $\boldsymbol{x}_{mpc}(k)$ is the state vector and $\boldsymbol{u}_{mpc}(k)$ is the input vector at time-step k for this state space model. It is assumed that full-state feedback is available, therefore $\boldsymbol{y}_{mpc} = \boldsymbol{x}_{mpc}$.

DMDc applies multiple delay-coordinates to account for input delay and state delay in the system. In Section 4.4, it was shown that the adapted DMDc algorithm produces three matrices, $\boldsymbol{A}_{dmd}$, $\boldsymbol{A}_{d}$, and $\boldsymbol{B}_{dmd}$. However, the MPC requires a single system matrix, $\boldsymbol{A}_{mpc}$. Therefore the DMDc system is converted into:

$$\begin{bmatrix} \boldsymbol{x}_{dmd}(k+1) \\ \boldsymbol{d}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{dmd} & \boldsymbol{0} \\ \boldsymbol{I}_{d} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{dmd}(k) \\ \boldsymbol{d}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{dmd} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{u}_{dmd}(k), \tag{5.24}$$

$$\boldsymbol{x}_{mpc}(k+1) = \boldsymbol{A}_{mpc} \cdot \boldsymbol{x}_{mpc}(k) + \boldsymbol{B}_{mpc} \cdot \boldsymbol{u}_{mpc}(k), \tag{5.25}$$

where $\boldsymbol{I}_d$ is the identity matrix that links the corresponding entries in $[\boldsymbol{x}_{dmd}(k) \;\; \boldsymbol{d}(k)]^T$ to $\boldsymbol{d}(k+1)$. This produces large state-space matrices with many output variables (represented in $\boldsymbol{d}(k+1)$) that are necessary for state predictions but do not require setpoint tracking. To ignore these variables in the control optimisation, they are assigned a zero weight in the MPC objective function. The objective function will be further discussed in the sections below.

Recall from Section 4.6.5 that $\dot{\theta}$ is not used in the estimated model. However, as discussed in Section 5.3, it is better for a controller to minimise $\dot{\theta}$, rather than θ. This is because θ has a non-zero steady-state value during a velocity step response due to aerodynamic drag. The state vector is therefore augmented with the $\dot{\theta}$ variable, such that the state vector is rather defined by,

$$\boldsymbol{x}_{mpc}{}^{T} = \begin{bmatrix} V_N & \theta & \dot{\theta} & \boldsymbol{d} \end{bmatrix}, \tag{5.26}$$

where $\boldsymbol{d}$ is the delay state vector defined in Equation 4.9. The $\boldsymbol{A}_{mpc}$ matrix is also augmented with a Backwards Euler numerical differentiation equation, such that,

$$\dot{\theta}(k) = \frac{1}{T_s}\theta(k) - \frac{1}{T_s}\theta(k-1) \tag{5.27}$$

In this way, a weight can be applied to the $\dot{\theta}$ variable in the MPC objective function to control this variable. $\boldsymbol{B}_{mpc}$ is augmented with zeros so that the state space matrix dimensions agree and so that $\boldsymbol{u}_{mpc}(k)$ does not directly influence $\dot{\theta}(k)$.

5.4.3. Algorithm implementation

A specific algorithm needs to be selected or designed to implement MPC in this work. There are numerous open-source methods available for this purpose. An extensive list of options is provided in the survey by [108] and the most promising ones are summarised here. A custom MPC implementation can be developed in MATLAB® or C++ with the aid of software packages like CVXGEN [109], ACADO [110], YALMIP [111], Multi-Parametric

Toolbox [112], and do-mpc [113]. Other open-source MPC implementations are also available as ROS packages from work done by [114] or [115].

The Simulink™ implementation of an MPC from the Model Predictive Control Toolbox™ [116] was selected for this work. It was selected because it integrates well with our simulation environment in Simulink and it specifically uses a discrete state-space plant model. The Model Predictive Control Toolbox™ also supports C++ code generation for stand-alone ROS nodes. Therefore, it can integrate with the SITL implementation of PX4 and it can be run on an OBC for practical implementations.

The Model Predictive Control Toolbox™ solves the control optimisation problem as a Quadratic Program (QP) at each time interval [116]. To do this, it applies an active-set QP solver using the KWIK algorithm from [117]. This QP usually consists of three features, namely,

- the objective function,
- the constraints, and
- the controller decision

The objective function provides a scalar value that quantifies the controller performance. The controller decision is the set of $\boldsymbol{u}_{mpc}$ values determined by the QP solver that minimises the objective function. The constraints are conditions that the controller decision should satisfy, such as bounds on $\boldsymbol{x}_{mpc}$, $\boldsymbol{u}_{mpc}$, and $\Delta\boldsymbol{u}_{mpc}$ values.

A powerful advantage of MPC is that constraints are easily included in the optimal control implementation. However, constraints are not necessary for this work and will not be applied. This is advantageous for practical implementations because unconstrained MPC is less computationally complex than constrained MPC.

The objective function used by the Model Predictive Control Toolbox™ is documented well by the corresponding user manual [116], and an overview of this implementation will be presented. The objective function consists of the sum of three terms that each quantify a specific aspect of the control performance, and is calculated as,

$$J(\boldsymbol{z}_k) = J_y(\boldsymbol{z}_k) + J_u(\boldsymbol{z}_k) + J_{\Delta u}(\boldsymbol{z}_k), \tag{5.28}$$

where $\boldsymbol{z}_k$ is the controller decision at time-step k. The three scalar performance measures are denoted as $J_y(\boldsymbol{z}_k)$ for output setpoint tracking, $J_u(\boldsymbol{z}_k)$ for Munipulated Variable (MV) tracking, and $J_{\delta u}(\boldsymbol{z}_k)$ for MV move suppression. Each performance measure includes

weights that balance the competing objectives of the different terms. These weights need to be manually tuned for a desired controller performance.

Output setpoint tracking

The performance measure of the output setpoint tracking is calculated as,

$$J_y(\boldsymbol{z}_k) = \sum_{j=1}^{n_y} \sum_{i=1}^{N_p} \left\{ w^y{}_j \left[r_j(k+i|k) - y_j(k+i|k) \right] \right\}^2, \tag{5.29}$$

where the symbols are denoted as,

k	Current control interval time-step.
n_y	Number of output variables.
N_p	Prediction horizon.
$y_j(k+i\|k)$	Predicted value of j^{th} output variable at i^{th} time-step from k.
$r_j(k+i\|k)$	Reference value of j^{th} output variable at i^{th} time-step from k.
$w^y{}_j$	Tuning weight for j^{th} output variable.

The controller receives the reference values, $r_j(k+i|k)$, for the prediction horizon starting at time-step k. Using the internal plant model, the predicted output values, $y_j(k+i|k)$, is determined based on the controller decision, $\boldsymbol{z}_k$. The values of N_p and $w^y{}_j$ are design choices that are constant controller specifications. The value of n_y are also constant and are determined from the plant model.

Manipulated variable tracking

In some control applications, it is desirable to keep specific MV variables close to a target value. In the multirotor use case, lower MV values are preferred because this corresponds to lower energy use. The MV target values are therefore set. The performance measure of manipulated variable tracking is calculated as,

$$J_u(\boldsymbol{z}_k) = \sum_{j=1}^{n_u} \sum_{i=0}^{N_p-1} \left\{ w^u{}_j \left[u_j(k+i|k) - u_{j,sp}(k+i|k) \right] \right\}^2, \tag{5.30}$$

where the symbols are denoted as,

n_u	Number of manipulated variables.
$u_j(k+i\|k)$	Control decision of j^{th} MV at i^{th} time-step from k.
$u_{j,target}(k+i\|k)$	Target value of j^{th} MV at i^{th} time-step from k.
$w^u{}_j$	Tuning weight for j^{th} MV.

The desired $u_{j,target}(k+i|k)$ values can be received for the prediction horizon starting at time-step k. However, for our use case, all $u_{j,target}(k+i|k)$ values are constant and zero.

The value of n_u is fixed by the plant model. The ${w^u}_j$ values are also constant and are determined as a design decision.

Manipulated variable increment suppression
Large increments or moves in the MV values are often undesirable for good control performance. In the multirotor use case, large increments of the acceleration MVs result in aggressive jerks which may cause the system to go beyond the accurate domain of the linear approximation model. High frequency moves in the acceleration MVs may also cause jittery flight dynamics because acceleration setpoint changes correspond to attitude changes. The performance measure of MV tracking is used to penalise increments in the MVs. This is calculated as,

$$J_\Delta u(\boldsymbol{z}_k) = \sum_{j=1}^{n_u} \sum_{i=0}^{N_p-1} \left\{ {w^{\Delta u}}_j \left[u_j(k+i|k) - u_j(k+i-1|k) \right] \right\}^2, \tag{5.31}$$

where the symbols are denoted as,

${w^{\delta u}}_j$ Tuning weight for movement in the j^{th} MV.

The values of ${w^{\Delta u}}_j$ are constant and are determined as a design choice.

It is important to note the similarities and differences between the LQR and MPC objective functions. The LQR implementation described in Section 5.3, did not included penalisation for MV increments. However, if $J_\Delta u(\boldsymbol{z}_k)$ is removed from Equation 5.28, the MPC and LQR optimiser consider the same variables.

The LQR optimisation corresponds to solving the unconstrained MPC optimisation problem for $N_p = \infty$. However, the LQR optimisation is run only once to determine the LQR gain, whereas the MPC optimisation is re-run at every time-step. Also note that the LQR uses a continuous-time model, but the MPC considered in this work uses a discrete-time model.

5.4.4. Integral action

A simple implementation of predictive control with multiple output variables does not inherently apply integral action or disturbance rejection. For the multirotor and suspended payload use case, MPC control without integral action results in a non-zero steady-state error of the multirotor velocity, due to wind disturbance and inaccuracies in modelling the drag.

Different methods have been proposed to apply integral action with an MPC. A commonly used method involves applying an integrator to the control action determined by the MPC

[118]. In this implementation, the MPC determines the optimal control action increment, $\Delta u_k{}^*$, and calculates the control action, $u_k = u_{k-1} + \Delta u_k{}^*$ which is then applied. Hence, integral action is applied to the plant input. This method is also described by [106].

Another commonly used strategy involves estimating an input disturbance that influences an output variable in the plant model [118]. In this way, integral action is applied to the plant output. This integral action strategy will be applied in this work. Since integral action is required for V_N in the North velocity controller, a disturbance model that influences V_N is augmented to the input matrix.

The resulting state-space matrix is,

$$\boldsymbol{x}_{mpc}(k+1) = \boldsymbol{A}_{mpc}\boldsymbol{x}_{mpc}(k) + \begin{bmatrix}\boldsymbol{B}_{mpc} & \boldsymbol{B_{ud}}\end{bmatrix}\begin{bmatrix}\boldsymbol{u}_{mpc}(k) \\ \hat{u}_{ud}(k)\end{bmatrix}, \tag{5.32}$$

where $\boldsymbol{B_{ud}}$ is the input disturbance model and $\hat{u}_{ud}$ is the estimated input disturbance value. The input disturbance model is designed as,

$$\boldsymbol{B_{ud}} = \begin{bmatrix}b_{ud} & 0 & 0 & \boldsymbol{0}\end{bmatrix}^T, \tag{5.33}$$

such that an input disturbance only influences V_N in the state vector,

$$\boldsymbol{x}_{mpc}{}^T = \begin{bmatrix}V_N & \theta & \dot{\theta} & \boldsymbol{d}\end{bmatrix}. \tag{5.34}$$

The variable b_{ud} is a tunable value that quantifies the effect of the input disturbance on V_N. This variable will be tuned in Section 5.4.5

The specific value of the non-zero matrix entry in $\boldsymbol{B_{ud}}$ has only a slight effect on the control performance, hence the iterative tuning process for this value was simple. The value of $\hat{u}_{ud}(k)$ is estimated by the default Kalman filter estimator from the Model Predictive Control Toolbox™. This filter is based on the state-space model from Equation 5.32. The value of $\hat{u}_{ud}(k)$ is then used in the QP solver to determine the optimal control action of the MPC.

It was determined from simulations with different payload parameters and disturbances that the MPC with the default input disturbance estimator provides acceptable controller performance without additional tuning. Hence, the default Kalman filter is used in the final control implementation. Zero steady-state error for velocity tracking was achieved for different payloads and different input disturbances, showing that integral action is achieved. The simulation results showing the MPC integral action will be shown and discussed in Section 5.5.

5.4.5. Tuning

An MPC can easily be tuned for a range of different design requirements. The same general design requirements will be applied to the MPC as to the LQR in Section 5.3, namely, to produce a fast velocity response with zero steady-state error while damping the payload oscillations quickly. The MPC is firstly tuned to have a similar response time to the LQR so that the controller performances can be roughly compared. Thereafter, it is tuned to produce a trajectory that is as smooth as possible.

The MPC parameters that are determined in the controller design are, N_p, N_c, $\boldsymbol{w}^y$, $\boldsymbol{w}^u$, $\boldsymbol{w}^{\Delta u}$, and b_{ud}. T_s is fixed by the system identification phase, since the sample time of the discrete model and the controller should match.

In the controller tuning process, a large value of ${w^y}_j$ corresponds to aggressive control of the j^{th} output variable, because the tracking error of that variable will be heavily penalised. In contrast, small values of ${w^u}_j$ or ${w^{\Delta u}}_j$, correspond to aggressive manoeuvres, because the control values are not heavily penalised in the objective function.

The computational complexity of the QP problem increases significantly with larger values of N_p [119]. The computational complexity also increases with larger values of N_c. Therefore the smallest values of N_p and N_c that still provide acceptable controller performance will be used. In the tuning process, the initial values were set to $T_p = T_c = 2 \times t_p$ where $T_p = N_p \times T_s$, $T_c = N_c \times T_s$, and t_p is the peak-time of the velocity step response with a PID controller. The objective function weights were then tuned for a desired controller performance.

The objective function weights were iteratively tuned for the desired control performance in a similar way to the LQR. All the weights corresponding to the delay-coordinates are set to zero. Due to the non-zero steady-state value of θ, the corresponding weight is also set to zero. Hereafter, the value of $T_p = T_c$ is incrementally decreased until a noticeable change in the controller performance. T_p is fixed at the smallest value before this change occurs. T_c is then further decreased until a noticeable change in performance.

The last variable to be tuned is b_{ud}, which influences the disturbance rejection performance of the controller. A large value of b_{ud} results in a fast disturbance rejection performance, however, it also produces a large overshoot. Starting at an initial guess of $b_{ud} = 1$, the value is iteratively tuned for a consistent performance with a range of different payloads and wind disturbances.

The final designed controller configuration parameters are shown in Table 5.2. Note that the weights in $\boldsymbol{w}^y$ correspond to the variables in, $[V_N \quad \theta \quad \dot{\theta}]$, the weight in $\boldsymbol{w}^u$ corresponds to the variable $A_{N_{sp}}$, and the weight in $\boldsymbol{w}^{\Delta u}$ corresponds to the variable $\Delta A_{N_{sp}}(k) = A_{N_{sp}}(k) - A_{N_{sp}}(k-1)$.

Table 5.2: MPC configuration parameters.

Parameter	Value
N_p	166
N_c	116
T_s	0.03 s
$\boldsymbol{w}^y$	[2 0 10]
$\boldsymbol{w}^u$	[0.1]
$\boldsymbol{w}^{\Delta u}$	[10]
b_{ud}	0.1

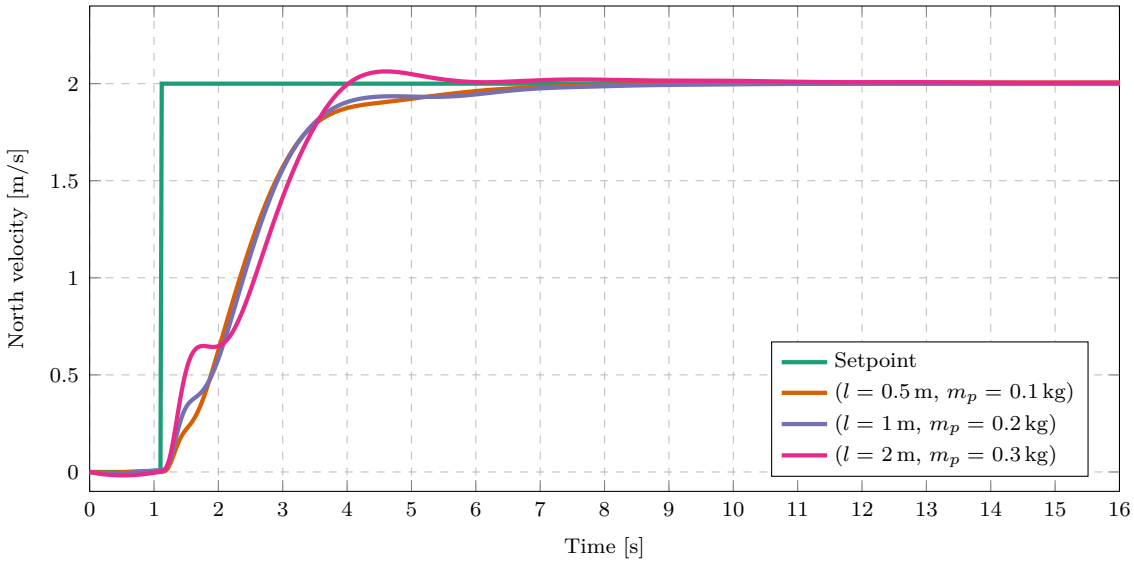


Figure 5.10: MPC velocity step responses with different payloads.

Figure 5.10 shows a plot of the simulated MPC velocity response with different payload parameters. It is clear that the controller damps the payload angles well and produces a smooth trajectory with different payloads. For each different payload, a DMDc model is first estimated. Thereafter the MPC is simulated with the same controller configuration defined in Table 5.2.

5.5. Implementation and results

After the system identification phase, active swing damping control can be applied to the multirotor and payload system. The control architectures are summarised in Table 5.1

by pairing the system identification techniques with the appropriate controllers. In this work, the *MPC architecture* refers to the entire control implementation, which includes the data-driven system identification method and the MPC. Likewise, the *LQR architecture* includes the white-box modelling, the parameter estimation methods, and the resulting LQR determined from the system identification model. These control architectures will be tested in simulation and their results will be shown and compared in this section.

5.5.1. Simple suspended payload

The modelling assumptions of the white-box model discussed in Chapter 3 defines a point-mass suspended with a rigid cable which is attached to the CoM of the multirotor. This is a simplistic suspended payload model but represents the dynamics of many practical payloads well. In this section, the simulated payload model complies will all these assumptions. This payload model is also used for simulations with an LQR controller in other studies [5, 6]. The simulation model used in this section was verified with practical data in Section 3.6.

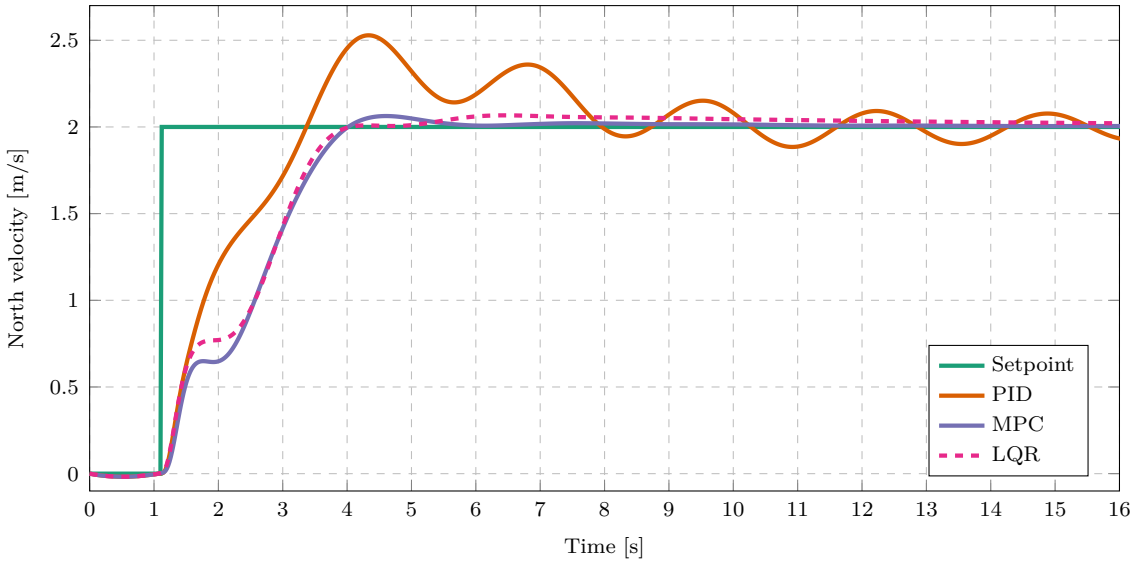


Figure 5.11: Velocity step response comparison of different controllers ($l = 2\,\text{m}$, $m_p = 0.3\,\text{kg}$).

From simulation results, it appears that both the MPC and LQR effectively damp the payload oscillations while controlling the velocity of the multirotor. Figure 5.11 shows the velocity step responses of the MPC, LQR and PID controllers for a multirotor with a suspended payload. From Figure 5.11 it is clear that both the LQR and MPC controllers actively damp the velocity oscillation caused by the swinging payload. The PID controller does not consider the payload angle, hence the oscillations are not damped well.

For the MPC and LQR, the respective models were first generated in the training phase of the simulation. Thereafter, the MPC and LQR were manually and iteratively tuned to

produce a step response with a similar response time and overshoot. The PID response shown uses the same controller gains used in the training phase.

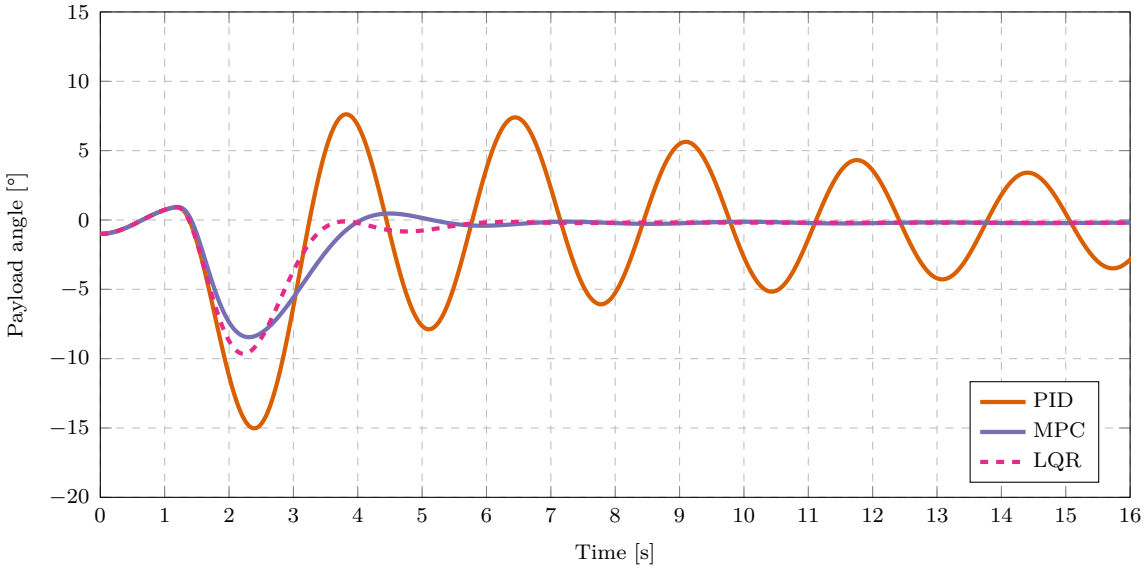


Figure 5.12: Payload angle comparison of different controllers ($l = 2\,\mathrm{m}$, $m_p = 0.3\,\mathrm{kg}$).

Figure 5.12 shows the payload angle data of the velocity step response. Both the MPC and LQR damp the payload angle well and the oscillations cease after only two or three cycles. In this case, the MPC response results in a slightly smaller initial swing angle, however, this is dependant on the specific tuning of each controller.

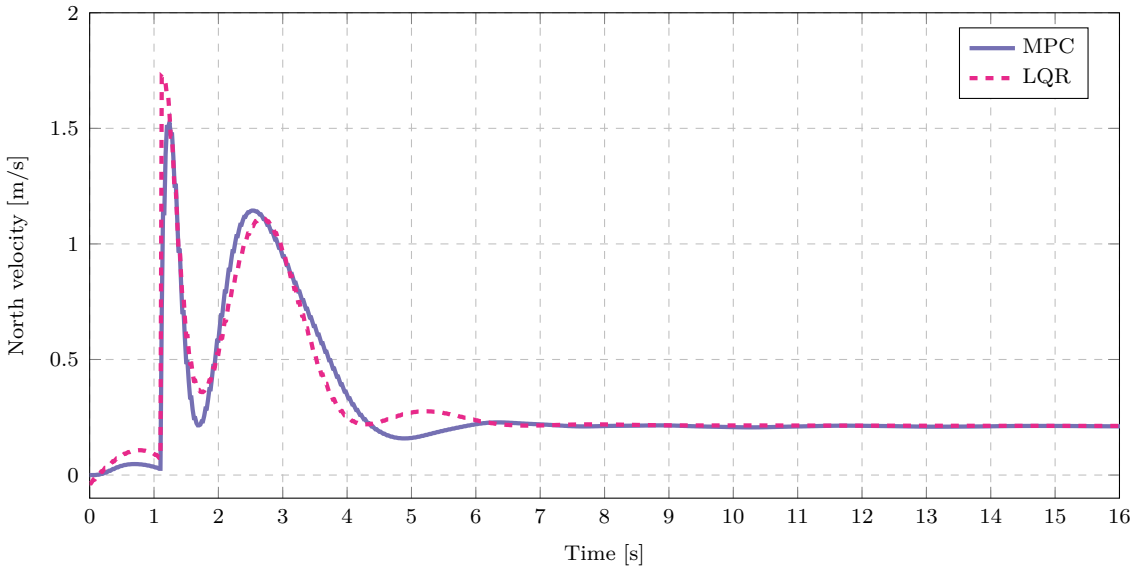


Figure 5.13: Acceleration setpoint commanded by different controllers for a velocity step input ($l = 2\,\mathrm{m}$, $m_p = 0.3\,\mathrm{kg}$).

Figure 5.13 shows the acceleration setpoint commanded by the two controllers for this step response. This is probably due to the inherent similarity between the controller

implementations as discussed in Section 5.4. The similarity in the acceleration setpoint responses also show that the energy expended in a velocity steps are roughly equal for these two controller implementations. However, this is also highly dependant on the weightings used in the optimisation problem of both controllers. Both controllers also produce a non-zero steady-state setpoint as expected, which is required to counter aerodynamic drag.

5.5.2. Different payload parameters

The system identification and control implementations are required to perform well with different unknown payload parameters. Therefore, numerous flights with a range of different payload were simulated, the respective models were trained and the controllers were implemented. Figure 5.14 and Figure 5.15 show velocity step responses with LQR and MPC implementations with two payloads flights. Both the parameter estimation with LQR implementation, and the DMDc with MPC implementation, handle flights with different cable lengths and payload masses well. In each flight, LQR and MPC damp the payload oscillations and control the multirotor velocity well.

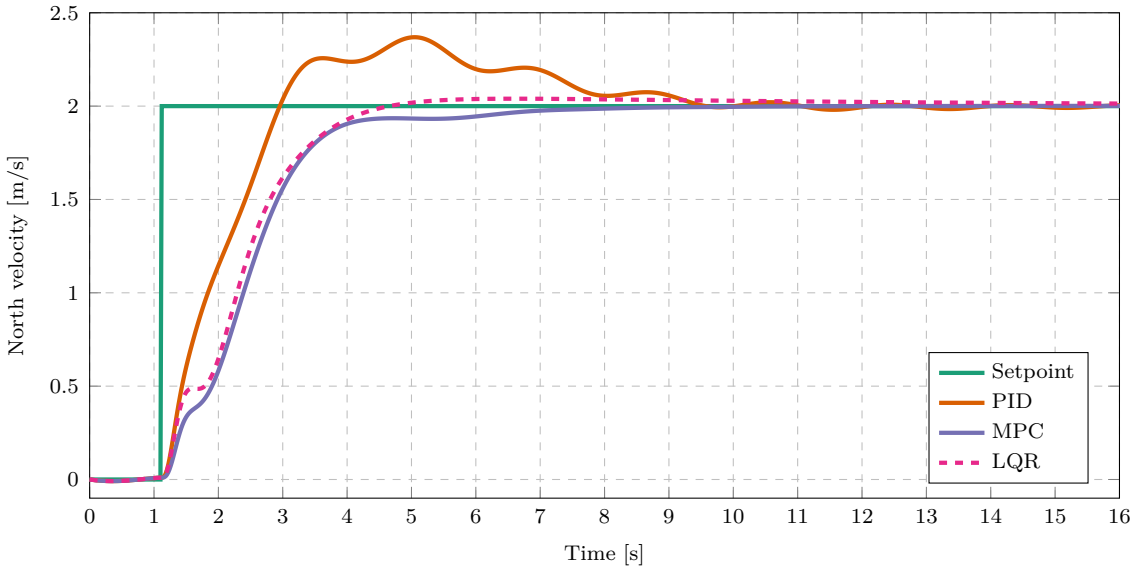


Figure 5.14: Velocity step response comparison of different controllers ($l = 1\,\mathrm{m}$, $m_p = 0.2\,\mathrm{kg}$).

The controllers were not specifically tuned for each simulation. Instead, the same controller parameters were used for these simulations as for the simulations in Section 5.5.1. This shows that each control architecture is adaptable to different payload parameters without manual intervention.

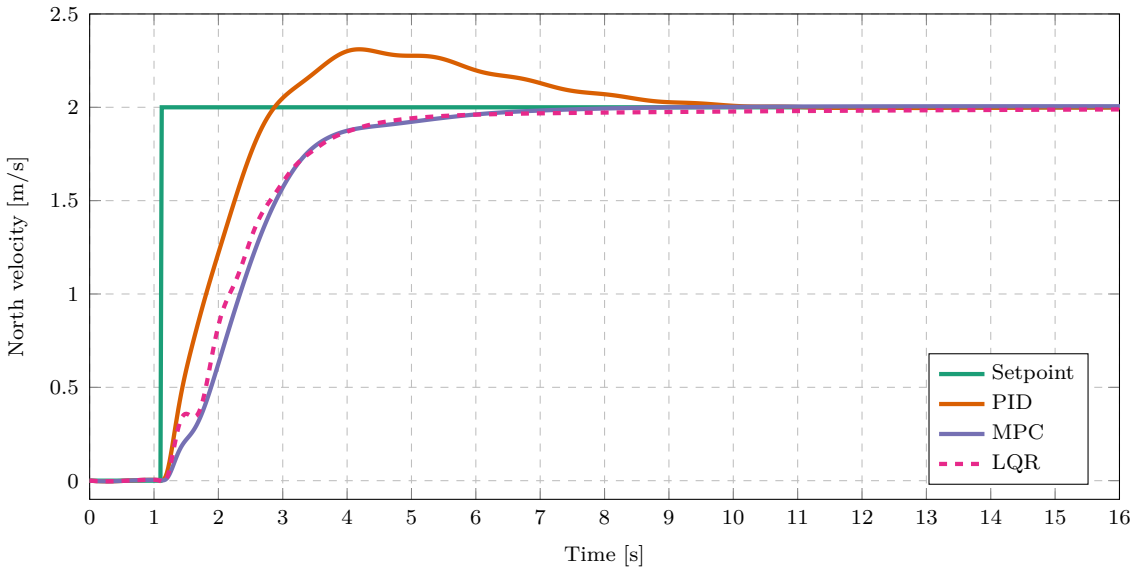


Figure 5.15: Velocity step response comparison of different controllers ($l = 0.5\,\text{m}$, $m_p = 0.1\,\text{kg}$).

5.5.3. Wind disturbance

For zero steady-state error with a practical system, a controller needs to apply some form of disturbance rejection. Practical systems experience unmeasured disturbances and other deviations which are not accounted for by the plant model. For example, a mean force applied by wind could prevent zero steady-state tracking error of the multirotor velocity without disturbance rejection. As discussed in Section 5.3, an integral state variable was added to the LQR plant model for integral action of the multirotor velocity tracking. As discussed in Section 5.4, an unmeasured input was added to the MPC plant model with a disturbance estimator to apply integral action to the multirotor velocity.

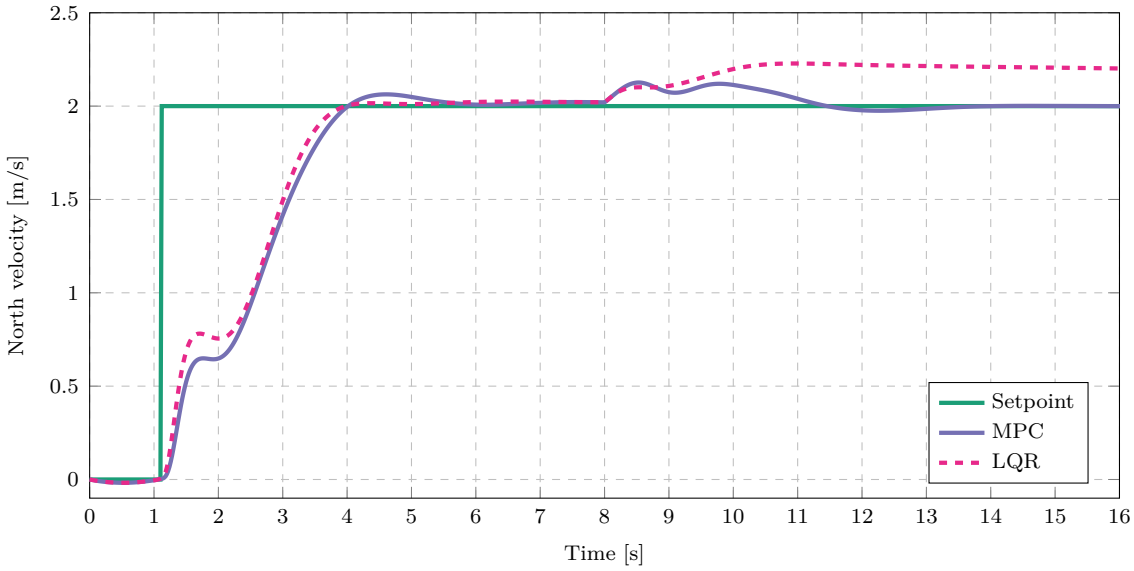


Figure 5.16: Effect of an unmeasured step input disturbance. ($l = 2\,\text{m}$, $m_p = 0.3\,\text{kg}$).

Figure 5.16 shows the responses of the controllers from Section 5.5.1 with a constant wind disturbance starting at 8 s. At Time = 8 s, a wind speed of 2 m/s is applied to the simulation model as an unmeasured step input. This mostly affects the multirotor velocity because the wind causes a greater drag force on the multirotor, hence a larger acceleration setpoint is required to maintain a constant velocity. For both system identification approaches, the models were trained without wind.

It appears that the MPC shows better disturbance rejection than the LQR when using the controller parameters which were tuned for good performance in Section 5.5.1. This is primarily because the weighting of the integral variable in the LQR optimisation was minimised to reduce overshoot. The integral weighting can be increased to improve integral action at the expense of increasing overshoot in the velocity response.

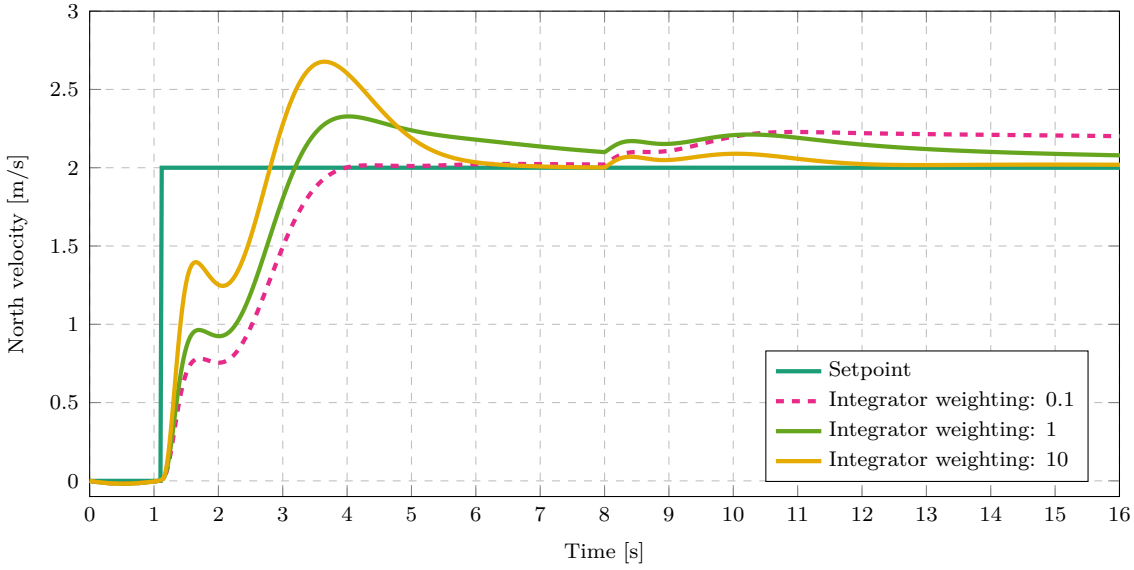


Figure 5.17: Different LQR responses for different integrator gains ($l = 2\,\text{m}$, $m_p = 0.3\,\text{kg}$).

Figure 5.17 shows the LQR responses with different integral state weightings. The other state variable weights are kept constant for each response. It is clear that the settling time and disturbance rejection of the LQR improves for larger integral state weighting. However, the overshoot increases significantly because of the integrator build-up at the start of the response.

In contrast to the LQR, the MPC shows good disturbance rejection while maintaining a low overshoot. This is because the disturbance estimator applies integral action which depends on the deviation of the actual dynamics from the plant model. whereas the LQR applies integral action proportional to the integral of the tracking error. Therefore the MPC implementation produces less integrator build up which results in a lower overshoot.

5.5.4. Dynamic payload

As discussed in Section 4.6.9, some payloads have dynamics that differ significantly from a suspended rigid mass. In this work, these payloads are referred to as dynamic payloads. An example of such a payload is an elongated payload, which can be represented by a double pendulum model.

In Section 4.6.9, the proposed system identification techniques were tested on simulated flight data with such a payload. For the white-box system identification approach, it was shown that the white-box model captures the dynamics of the dominant frequency of the oscillating payload but ignores the higher frequency dynamics. For the black-box approach, a prediction model was generated from a set of training data. This model accurately predicted the multirotor and payload dynamics, including the low and high-frequency dynamics, of a set of testing data.

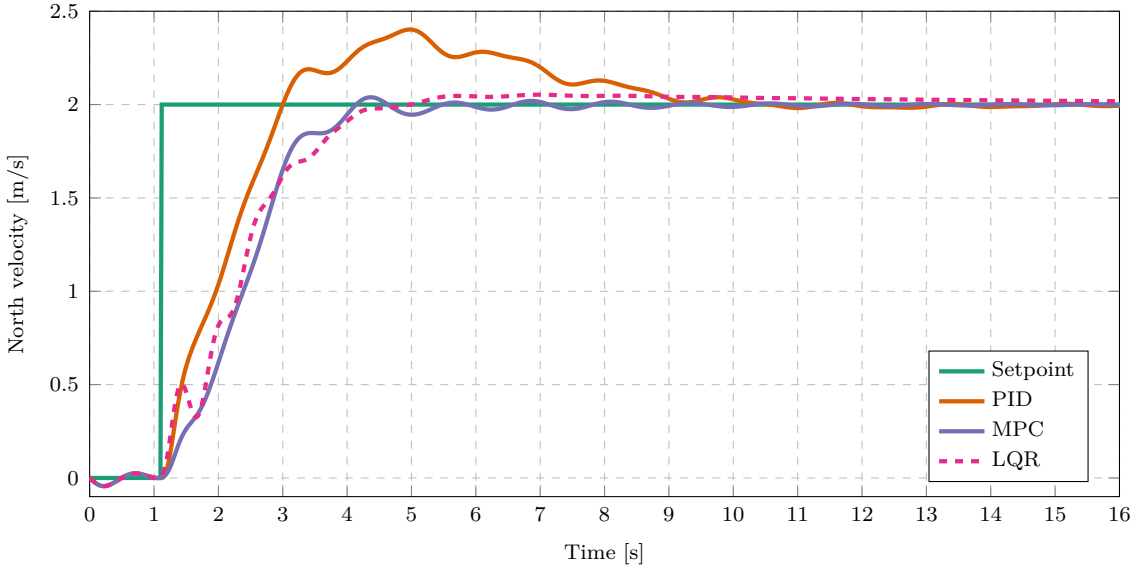


Figure 5.18: Velocity step response comparison of different controllers ($l = 2\,\mathrm{m}$, $m_p = 0.3\,\mathrm{kg}$).

For the simulations in this section, accurate system identification models were generated as described in Section 4.6.9 and used in the MPC and LQR controllers. Figure 5.18 shows the resulting controller responses with a dynamic payload. It appears that the velocity responses of both the LQR and the MPC are less smooth than with a simple suspended payload and show small velocity oscillations.

Figure 5.19 shows the cable angle for a velocity step response. It appears that the LQR and MPC damp the payload oscillations with a similar response time. However, the superimposed, high-frequency oscillations are smaller in the LQR response than in the

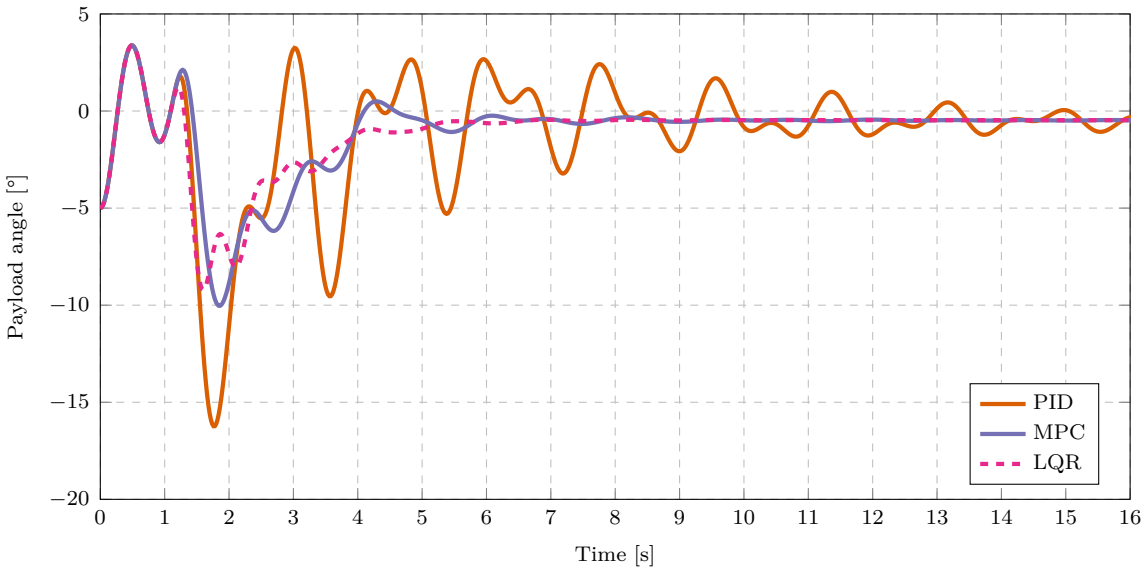


Figure 5.19: Payload angle comparison of different controllers ($l = 2\,\text{m}$, $m_p = 0.3\,\text{kg}$).

MPC response. The LQR naively damps the payload oscillations because its controller gain is determined from the same dynamical model as for a simple suspended payload. Because the angle of the payload relative to the cable is not measured or considered in the LQR plant model, it does not directly damp these high-frequency oscillations. The MPC should account for the superimposed frequency and provide a smoother response than the LQR, since the black-box model includes the double pendulum dynamics in its prediction model. Therefore it is expected that the MPC optimiser determines a smooth trajectory that damps both the low and high-frequency oscillations.

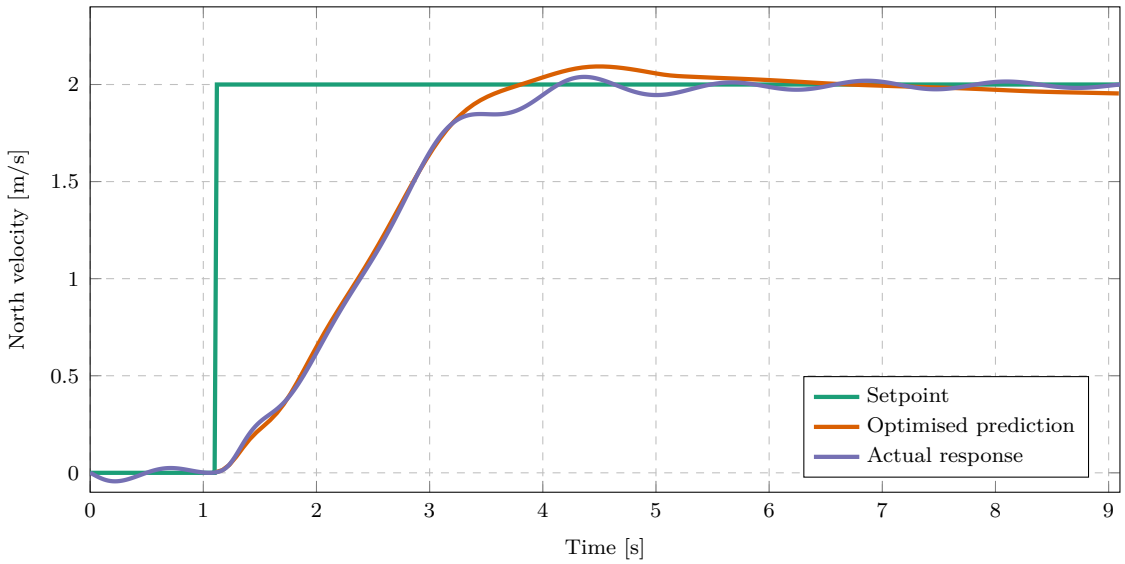


Figure 5.20: Optimised prediction and actual velocity response of the MCP with a dynamic payload.

However, the MPC does not provide a smooth velocity or payload angle response. Even though the MPC generates a smooth optimised trajectory with the plant model, the actual response of the simulated system differs from this prediction. Figure 5.20 shows the predicted velocity of the MPC optimiser, given the velocity setpoint and initial condition at Time = 1.1 s. The actual simulated response, resulting from replanning at every time-step with the MPC, is also shown in Figure 5.20. For the first part of the velocity response, the the actual response matches the optimised trajectory well. However, after Time = 3.2 s, the predicted dynamics is noticeably different from the actual response to the optimised input sequence.

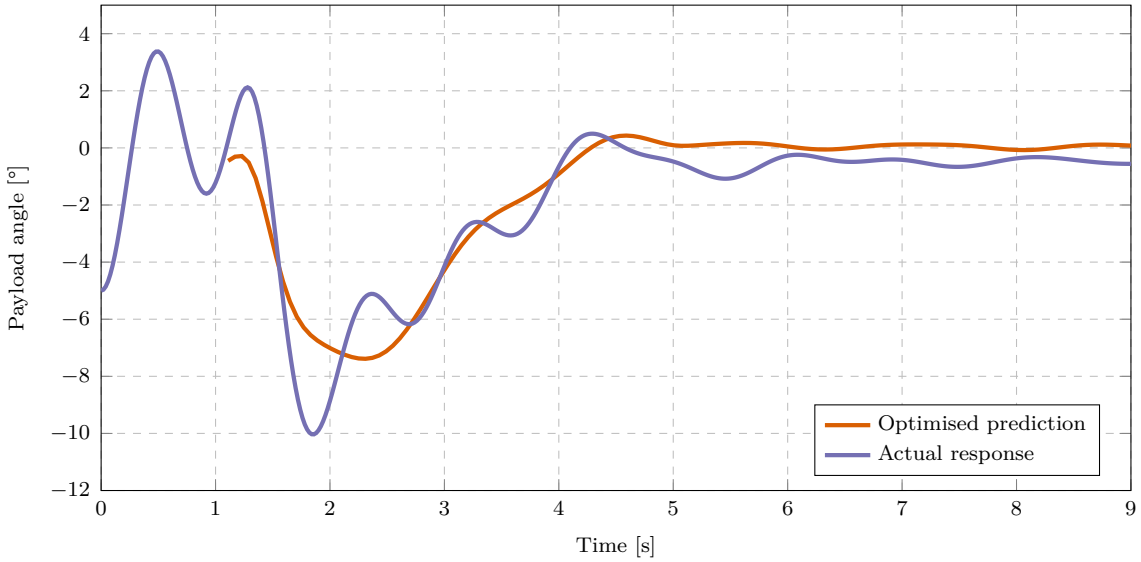


Figure 5.21: Optimised prediction and actual payload angle response of the MCP with a dynamic payload.

Figure 5.21 shows the predicted and actual payload angle response for this simulation, starting at the same time-step. The MPC optimiser also determined a smooth trajectory for the payload angle, but the actual response differs significantly from this trajectory. Even though the black-box model predictions accurately matched the testing data, Figure 5.20 and Figure 5.21 show that the model is not an accurate approximation of the simulated system for all values of the state and input vectors.

The estimated model provides accurate predictions in the domain of state and input vectors considered in the training data. However, the MPC generates trajectories that are beyond this domain of training data. The multirotor with a simple suspended payload represents a mildly non-linear system, hence the linear approximation was effective for control with an MPC. However, a double pendulum system reveals highly non-linear dynamics with multiple fixed points. This system also includes an unmeasured state variable which adds complexity to dynamics.

From the results in Figure 5.20 and Figure 5.21 it appears that the data-driven linear model results in acceptable control with an MPC. However, the actual dynamics do not follow the optimised trajectory of the MPC and the MPC does not conclusively outperform the LQR implementation.

5.5.5. Change in unconsidered system parameters

The control architectures have been shown to be adaptable to variations in the payload parameters. However, changing other system parameters may affect the performance of the different controllers. As mentioned in Chapter 4, a disadvantage of the white-box system identification approach used by the LQR, is that parameter estimation techniques need to be manually designed for each unknown parameter. In the specific implementation, the LQR model assumes that the multirotor mass is known. Hence, changing the mass of the multirotor is detrimental to the accuracy of plant model and therefore affects the LQR performance. In contrast, the data-driven system identification method for the MPC plant model does not rely on such modelling assumptions.

Simulations were performed with an altered multirotor mass to demonstrate how the control architectures handle changes in other system parameters. Figure 5.22 shows the velocity step responses of the PID, MPC and LQR implementations with an altered multirotor mass. For these simulations, the original multirotor mass, $m_Q = 0.796\,\text{kg}$, was decreased by $0.250\,\text{kg}$, resulting in a new multirotor mass of, $m_Q = 0.546\,\text{kg}$. The same system identification processes were naively followed as in the previous sections, without prior knowledge of the change in m_Q. The same tuned controller parameters were also used.

In Figure 5.22 it appears that the LQR results in lower payload oscillations than the PID controller but induces higher frequency oscillations. This results in jitter in the velocity response with the LQR. The LQR control performance has degraded because the dynamics of the LQR plant model differs significantly from the actual dynamics. In contrast, the MPC still results in a smooth velocity profile and damps the payload oscillations effectively, as in previous simulations. This is expected since the system identification model used by the MPC included the effect of the changed mass by estimating the entire model without considering individual parameters.

It should be noted that another mass estimator can be implemented to estimate m_Q in a flight stage before the payload is added. However, this involves manually redesigning the system identification procedure for each new unknown system parameter such as m_Q. In these simulations, it was shown that changing non-estimated system parameters in the white-box approach can be detrimental to the control performance. Unlike the white-box

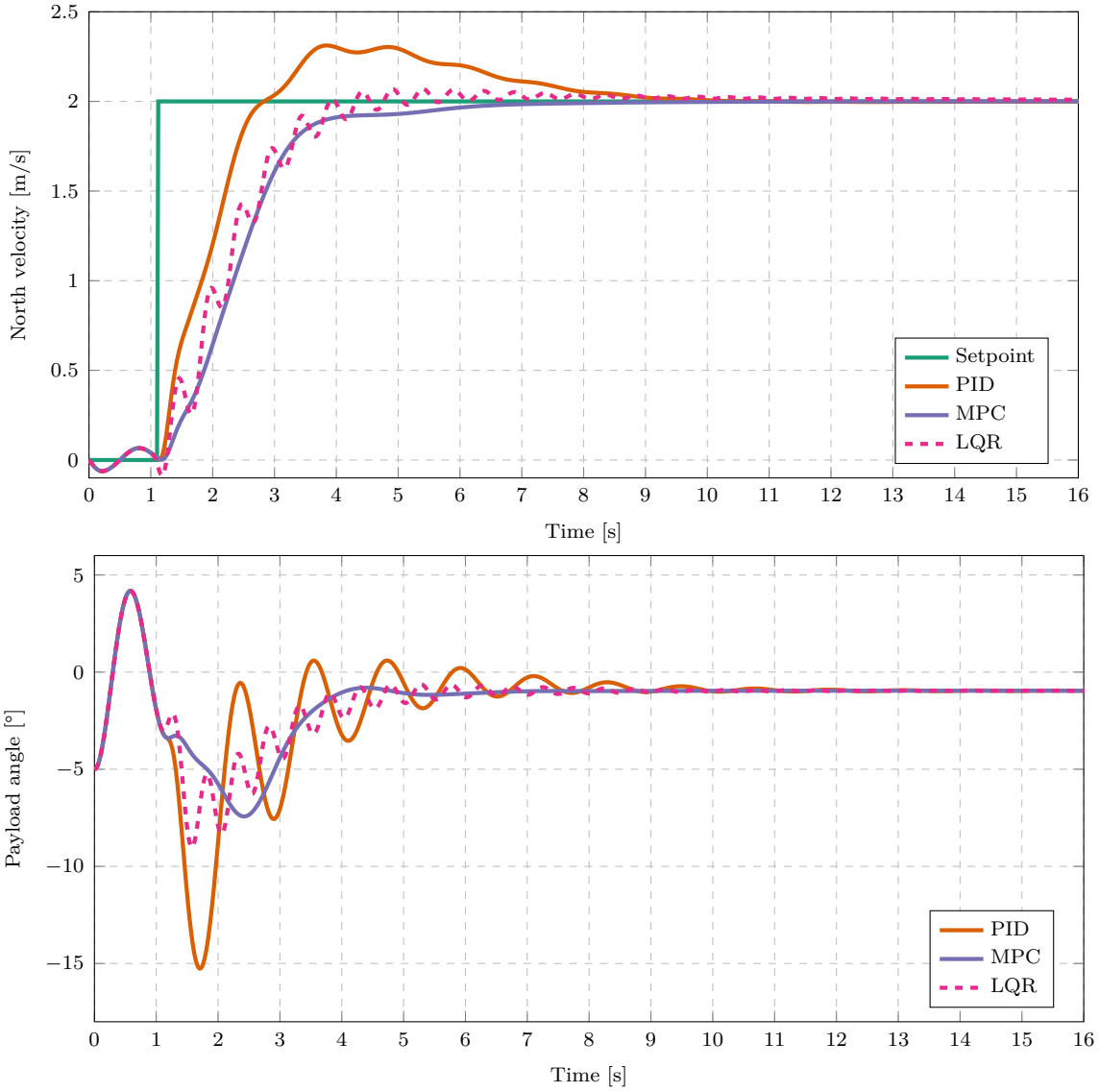


Figure 5.22: Velocity step responses with the multirotor mass decreased by 0.25 kg ($l = 0.5\,\mathrm{m}$, $m_p = 0.3\,\mathrm{kg}$)

approach, the black-box approach handles changes in different system parameters well without prior knowledge of these parameters.

5.6. Summary

From simulations without wind disturbances, the MPC and LQR architectures deliver similar control performances for a range of different payload parameters. Both controllers result in a similar response time and velocity overshoot, and the payload angle is damped well by both controllers. Therefore, in the absence of wind, the control performance does not conclusively differentiate between the LQR and MPC architectures for a simple suspended payload. As expected, the PID controller does not provide acceptable control of the multirotor with a suspended payload and does not actively damp the payload oscillations.

Both the LQR and the MPC effectively rejected the unmeasured input disturbance caused by wind, resulting in zero steady-state tracking error for the multirotor velocity. However, the LQR implementations which are tuned for effective disturbance rejection result in large velocity overshoots due to the integrator build-up. The MPC implementation applies integral action with a disturbance estimator and achieves zero steady-state error without increasing the velocity overshoot.

For simulations with a dynamic payload, both the LQR and MPC effectively applied swing damping control. However, the trajectories were not as smooth as with the simple suspended payload. Even though the simulated dynamics differed significantly from the simple suspended payload model used by the LQR, the LQR still managed to damp the payload oscillations effectively. The MPC also managed to damp the payload oscillations, but did not conclusively outperform the LQR. Even though the estimated model used by the MPC showed good prediction accuracy with the given testing data set, the actual system response did not follow the predicted trajectory of the MPC well. It appears that the optimised trajectory of the MPC is beyond the domain where the linear model provides an accurate approximation of the non-linear dynamics. Therefore, an improved data-driven system identification model, which provides an accurate approximation of the dynamics for a larger domain, is required for improved MPC control of such a dynamic payload.

Both the LQR and the MPC architectures handle different payload parameters well. Even though the parameter estimation techniques (used with the LQR) did not consider the mass of the multirotor (m_Q), the LQR architecture still provided acceptable swing damping control with small changes in m_Q. However, it was shown that the LQR architecture

produces an undesirable control performance for large changes in m_Q. Therefore, the parameter estimation procedure will need to be redesigned to account for such changes in other system parameters. However, the data-driven approach does not rely on modelling assumptions, hence the MPC still provides good control performance for different values of m_Q.

The advantage of the LQR architecture is that it is computationally simple in comparison to an MPC. However, the LQR architecture is designed for a specific system configuration and only accounts for changes in specific system parameters. In contrast, the MPC architecture provides a good general solution for different system configurations without considering individual parameters and without a priori modelling.

Chapter 6

Experimental design

As discussed in the literature study in Chapter 2, experimental data is a valuable part of any work involving multirotors control. This chapter will provide an overview of the hardware, software, Hardware-in-the-Loop (HITL) simulations, and practical methodology used in this work.

6.1. Hardware components

The main hardware components in this work include a multirotor vehicle, a payload angle sensor, and an OBC. These components are coupled together into the final multirotor system which will be used for practical flights.

6.1.1. Multirotor

Figure 6.1: Honeybee multirotor equipped with a OBC and payload angle sensor

The multirotor used in this work is a custom-built, lightweight multirotor named *Honeybee*. This vehicle was developed in the Electronic System Laboratory (ESL) at Stellenbosch

University [10]. Figure 6.1 shows a photo of Honeybee equipped with an OBC and payload angle sensor.

The physical parameters of this multirotor are summarised in Table 6.1. Note that the mass and inertial parameters include the OBC and payload angle sensor. The thrust profile of each motor is given by the third-order polynomial mapping the input Pulse-Width Modulation (PWM) signal, x, to the thrust output, T_m [10]:

$$T_m(x) = -3.508 \cdot 10^{-9}x^3 + 1.627 \cdot 10^{-5}x^2 - 0.0172x + 4.528 \tag{6.1}$$

Table 6.1: Physical parameters of Honeybee.

Description	Parameter	Value
Mass	m_Q	0.952 kg
Motor distance	d	0.11 m
Virtual yaw moment arm	R_N	$7.997 \cdot 10^{-3}$ m
Motor time constant	τ	15 ms
Mass moment of inertia about $\bar{\boldsymbol{x}}_\mathcal{B}$	I_{xx}	$2.00 \cdot 10^{-3}$ kg·m^2
Mass moment of inertia about $\bar{\boldsymbol{y}}_\mathcal{B}$	I_{yy}	$1.32 \cdot 10^{-3}$ kg·m^2
Mass moment of inertia about $\bar{\boldsymbol{z}}_\mathcal{B}$	I_{zz}	$3.35 \cdot 10^{-3}$ kg·m^2
Aerodynamic drag coefficient in $\bar{\boldsymbol{x}}_\mathcal{B}$	C_{Q_X}	0.096 m^2
Aerodynamic drag coefficient in $\bar{\boldsymbol{y}}_\mathcal{B}$	C_{Q_Y}	0.096 m^2
Aerodynamic drag coefficient in $\bar{\boldsymbol{z}}_\mathcal{B}$	C_{Q_Z}	0.256 m^2

The Flight Controller (FC) implemented on Honeybee is a *Pixhawk 4 mini* shown in Figure 6.2. This board includes internal IMU, magnetometer, and barometer sensors and is connected to an external GPS sensor and an additional magnetometer. Furthermore, an Radio Control (RC) receiver is used to communicate with a radio transmitter for manual pilot control. A telemetry radio module is used for communication with a ground control station. The OBC and external payload angle sensors are also connected to the FC.

Figure 6.2: Photo of a Pixhawk 4 mini FC [7]

6.1.2. Payload angle sensor

A sensor is required to measure the payload state as Euler angles about the $\bar{\boldsymbol{x}}_{\mathcal{B}}$ and $\bar{\boldsymbol{y}}_{\mathcal{B}}$ axes. Figure 6.3 shows a customised sensor attached to the Honeybee airframe for this purpose.

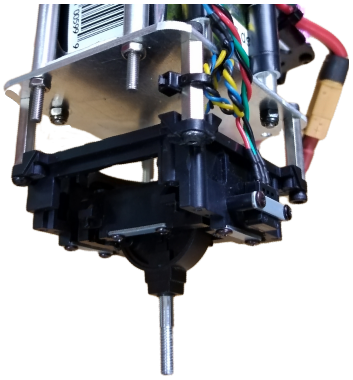


Figure 6.3: Payload angle sensor with linear potentiometers

This sensor was constructed from a two-axis joy-stick and two linear potentiometers. Each potentiometer is implemented as a voltage divider and attached to an ADC channel on the FC. Experimental data was used to map the ADC reading to an angle measurement with a best-fit straight line function. A cable can therefore be attached to this device to transport a suspended payload and measure the payload swing angles during flight.

6.1.3. On-Board Computer

An OBC, also called a companion computer, is used to run intensive computational processes that cannot be handled by the FC. A NVIDIA® Jetson Nano™, which is shown in Figure 6.4, is used as the OBC for Honeybee. This has 4GB memory and a quad-core processor which runs at 1.43 GHz. The OBC is connected to a serial port on the FC for communication.

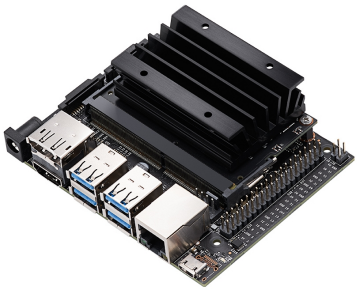


Figure 6.4: NVIDIA® Jetson Nano™ [9] used as a OBC

6.2. Software Toolchain

The software toolchain used with Honeybee includes PX4-Autopilot, QGroundControl, ROS, and Gazebo simulator. This toolchain and was also implemented and described well by Erasmus and Jordaan [5], Slabber and Jordaan [6], and Grobler and Jordaan [10]. A brief overview of the software toolchain is provided here.

6.2.1. PX4-Autopilot

PX4-Autopilot is an open-source flight stack that focuses on autonomous UAVs [94] and is used in research and industrial applications. SITL and HITL simulations are supported by PX4, which is helpful for research and development.

6.2.2. QGroundControl

QGroundControl (QGC) is the recommended ground station software for PX4 systems [7]. A ground station computer running QGC can be used to monitor and control a PX4 vehicle. QGC communicates with PX4 via the MAVLink protocol over a telemetry connection during practical flights. During SITL simulations, QGC connects to PX4 over a local User Datagram Protocol (UDP) connection. QGC is also an open-source product.

6.2.3. Gazebo simulator

Gazebo is an open-source graphical-based physics simulator used for robotics. This is the recommended simulator in the PX4 development toolchain and is capable of both SITL and HITL simulations [7]. The PX4 flight stack includes multirotor models developed for Gazebo which include realistic sensor plugins. These plugins apply sensor noise, drift and bias which replicates the actual sensors used on Pixhawk boards. The physical parameters of these models were changed to match Honeybee and a suspended payload was added to the model as shown in Figure 6.5.

6.2.4. Robot Operating System

ROS is a communication framework with a set of tools used for robotics and control applications [120]. ROS is also open-source and is supported by PX4. In this framework, executables are called ROS nodes and these nodes interact with each with a publish-subscribe architecture. A ROS node can publish messages to a topic, and a different node can subscribe to that topic to read those messages.

MAVROS is an open-source ROS package that provides a bridge between ROS and PX4 through the MAVLink protocol. A MAVROS node receives MAVLink messages from PX4

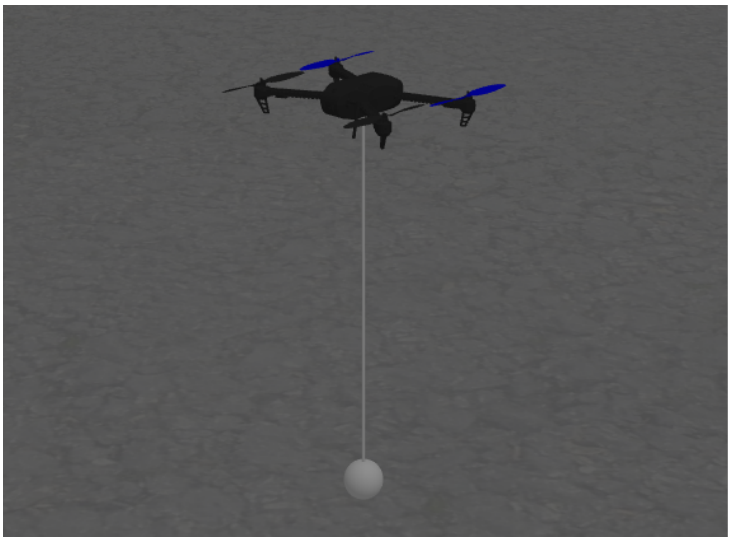


Figure 6.5: Model of Honeybee in the Gazebo simulator

and converts this to published ROS topics for other ROS nodes to access. The MAVROS node also subscribes to other topics to receive ROS messages and send this data to PX4.

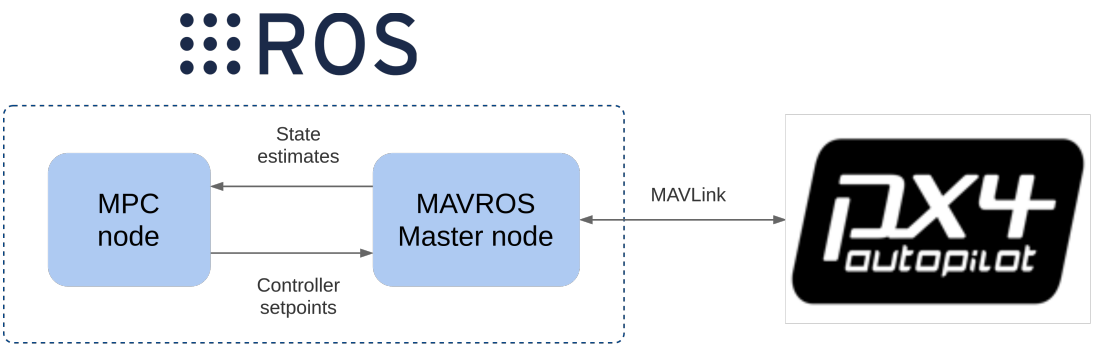


Figure 6.6: Communication between ROS, flight stack, simulator, and ground station [10]

Simulink™ was used to convert the MPC controller developed in Section 5.4 to C++ code and generate a standalone ROS node. Figure 6.6 illustrates how the MPC node is used as an offboard controller with PX4 and MAVROS. A MAVROS Master node receives data, including state estimates, from PX4 through MAVLink communication. This data is published by MAVROS to various ROS topics. State estimate data is received by the MPC node by subscribing to the appropriate MAVROS topic. After the MPC node calculates the next controller decision, it publishes the controller setpoint data to a MAVROS topic. The MAVROS node then sends it to PX4 via MAVLink.

6.3. Hardware-in-the-Loop simulations

In HITL simulations, the simulator mimics the sensor outputs, but the PX4 firmware and the accompanying software runs on the designated hardware. Figure 6.7 illustrates how

the different software and hardware components interlink for HITL simulations.

QGC runs on the desktop computer and communicates with the Gazebo simulator with MAVlink messages over a local UDP connection. The Gazebo simulator also runs on the desktop computer and simulates the Honeybee multirotor. The simulator mimics the multirotor sensor values and sends them to PX4 with MAVlink messages over a USB connection.

The PX4 firmware runs on the FC board. Based on the received sensors values, the PX4 controllers determine PWM actuator commands which are communicated back to Gazebo. The OBC runs the MPC and MAVROS nodes which send and receive MAVLink messages over a serial port connection to the FC.

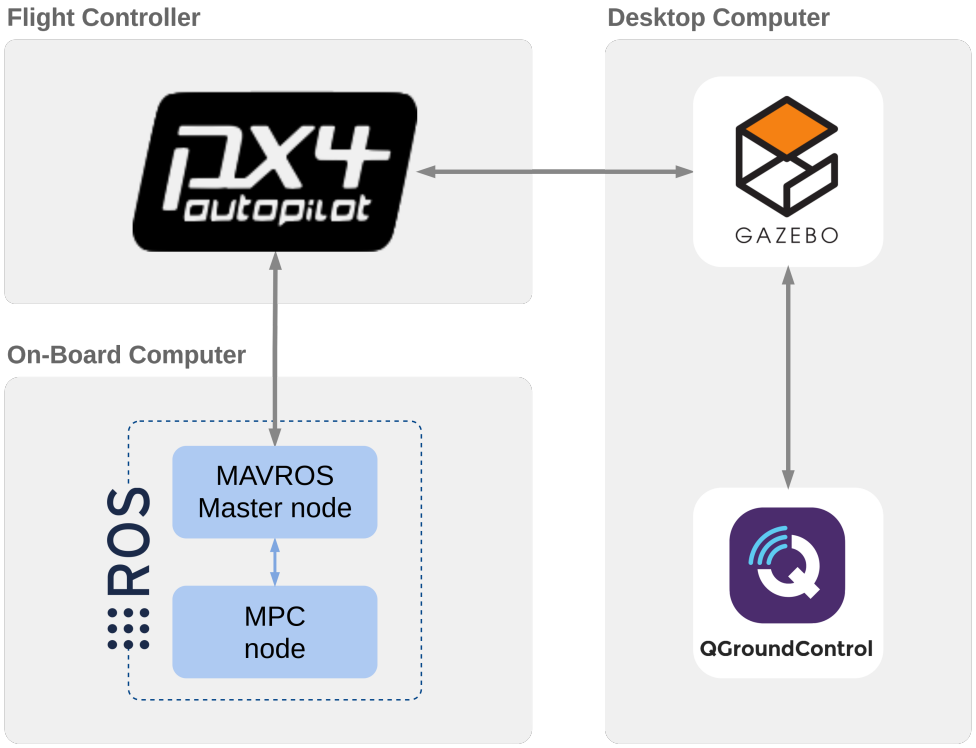


Figure 6.7: Different software and hardware components of a HITL simulation

6.4. Practical flights

The major differences between simulated and practical flights involve wind disturbances and the attachment of the payload. In simulations, the payload cable is attached to the exact CoM of the multirotor. However, for practical flights the cable is attached slightly below the CoM of Honeybee due to mechanical constraints. Practical flights are also influenced by wind gusts which are difficult to model accurately in simulations. The

measurement noise experienced by a practical multirotor may also differ from the noise models used in simulations.

Figure 6.8: Practical flight with Honeybee and a suspended payload

Figure 6.8 shows Honeybee with a suspended payload during a practical flight. Numerous flights were performed with different payload masses, cable lengths and wind conditions. Different flights were also performed with a dynamic payload. The system identification methods were then applied to the flight data logged by PX4. The results of these flight experiments will be discussed in Chapter 7.

The same general methodology used for simulations will be used for practical flights:

1. Arm the multirotor for data logging to start.
2. Takeoff and hover with the multirotor.
3. Command velocity step setpoints.
4. Land the multirotor.
5. Disarm the multirotor for data logging to stop.
6. Download the data log from the multirotor.
7. Split the data into separate training and testing periods.

8. Build a model from the training data.

9. Evaluate model predictions with the testing data.

6.5. Summary

This chapter provided an overview of the hardware and software used in this work. The HITL simulation and practical flight setups were also discussed. This provides a background for the experimental tests, results, and discussion in the next chapter.

Chapter 7

Practical implementation and results

In Chapter 4, it was shown within simulations that both the white-box and black-box system identification models could accurately represent the dynamics of a multirotor with a suspended payload. In Chapter 5, it was also shown within simulations that the proposed controllers effectively achieve swing-damping control. However, practical flights may differ significantly from simulations, which would affect the performance of these implementations. Wind is a common unmeasured disturbance that influences the flight dynamics of a multirotor, but this disturbance was not considered in simulations. The practical dynamics and sensor noise may also differ from the simulation model, which further motivates the need for practical data.

In this chapter, the system identification techniques will be applied to practical flight data using the same methodology described in Chapter 4. The effect of different wind conditions and payloads on the performance of these techniques will be investigated. The performance of these techniques on a practical dynamic payload system will also be shown. Finally, HITL simulations will be performed to determine whether the available hardware can handle the computational complexity of the MPC implementation.

7.1. Parameter estimation with practical data

In Section 4.3, a cable length parameter estimation technique was performed with simulation data and the resulting white-box models provided reasonably accurate representations of the simulated multirotor-payload dynamics. In this section, the cable length estimation technique will be applied to practical flight data with different payload masses and cable lengths. The effect of wind on the parameter estimation technique will also be investigated. Finally, the cable length estimator will be applied to data from a dynamic payload.

7.1.1. Simple payload cable length estimation

As discussed in Section 4.3.3, an FFT of the payload angle data is used to estimate the natural frequency of the suspended payload, which is used to estimate the cable length. Percentage Error (PE) is used as the error metric to quantify the estimation accuracy.

The PE of the cable length estimation is calculated as,

$$PE = \frac{l_{estimated} - l_{actual}}{l_{actual}} \times 100\%, \tag{7.1}$$

where l_{actual} is the actual cable length and $l_{estimated}$ is the estimated cable length. The PE can be interpreted as the percentage of the actual length by which the actual length differs from the estimated length.

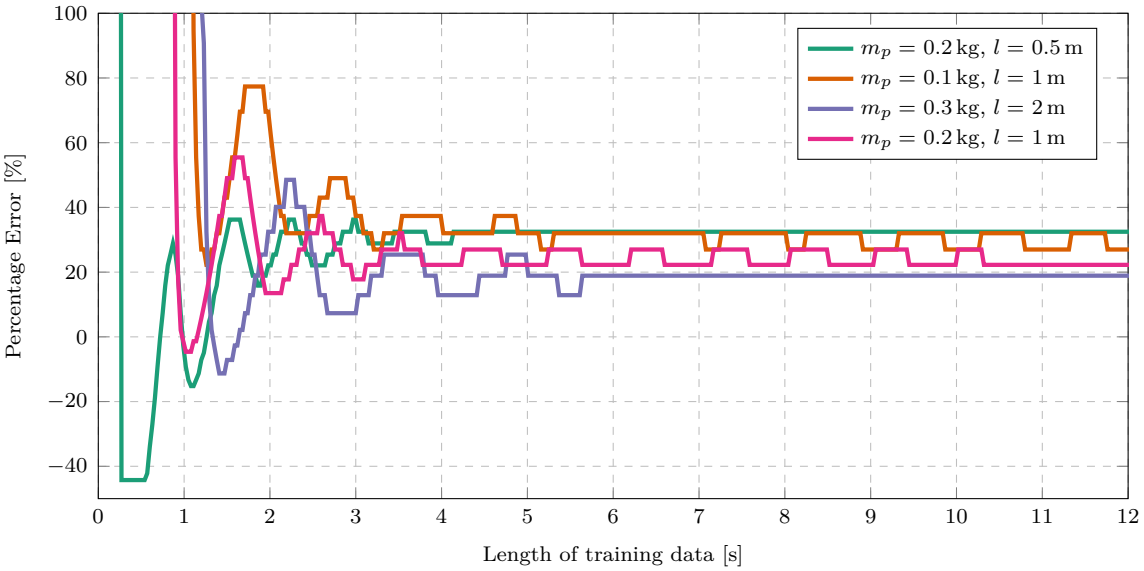


Figure 7.1: Plot of the error in cable length estimation as a function of length of training data (wind speed $\approx 0.5\,\text{m/s}$).

Figure 7.1 shows the PE of the cable length estimation for practical flight data with different payload masses and cable lengths using practical fligth data. Note that for each payload configuration, the estimation converges to a constant error after a sufficient length of training data. For these payload configurations, the converged PE ranges from 18.9 % to 32.4 %. These errors may be due to the large difference between the theoretical and the damped natural frequency. It appears that the PID controllers damp the payload oscillations significantly, which affects the oscillation frequency of the payload. Hence, an inaccurate length is estimated from the frequency peak identified in the FFT. Unlike in the simulations, the actual cable attachment may be below the CoM of the vehicle. This increases the effective suspended length and may also contribute to the error in parameters estimation.

Figure 7.2 compares the actual payload angle to the predicted angle of the white-box model for a velocity step in a practical flight. The cable length estimated from this flight is 2.64 m resulting in a PE of 32.0 %. The prediction matches the general shape of the practical data well. However, the transient response of the practical data, which can be

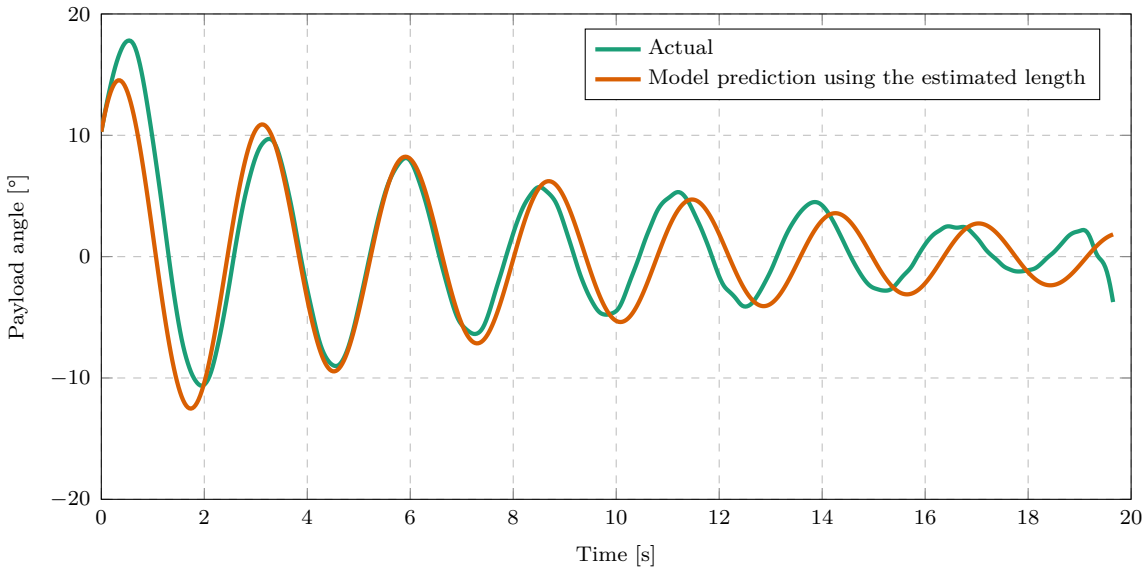


Figure 7.2: White-box model prediction for a North velocity step input ($l = 2\,\text{m}$, $m_p = 0.3\,\text{kg}$.).

seen in the first two oscillation peaks, is noticeably different from the model prediction. This is probably due to the inner loop controllers dynamics which affects the transient response of the practical data but are not accounted for in the white-box.

Furthermore, the practical swing angle peaks are attenuated by non-linear damping, which differs from the linear damping of the white-box model. This is a minor modelling error expected from a linear approximation of a non-linear system.

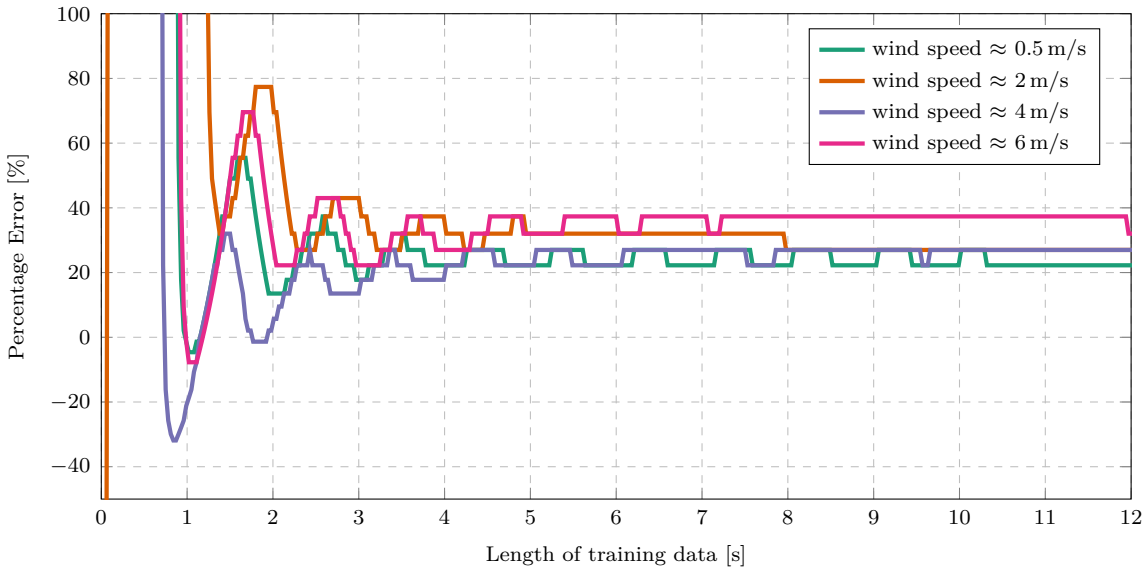


Figure 7.3: Cable length estimation error as a function of length of training data with wind disturbances ($m_p = 0.2\,\text{kg}$, $l = 1\,\text{m}$).

Figure 7.3 shows the PE of cable length estimation for flights with different wind conditions. These flights were all performed with the same payload. Wind conditions are referenced here by the wind speed recorded by the website, www.yr.no, for the hour of the day of the flight. It appears that the wind speed does affect the parameter estimation result since the estimation error differs significantly for different wind speeds. This is probably due to the variable damping effect of the controllers at different wind speeds. From the considered flights, it appears that the largest PE occurs at the highest wind speed, and the lowest PE at the lowest wind speed. However, only a few different wind speeds were tested and a trend cannot be identified conclusively from this small sample.

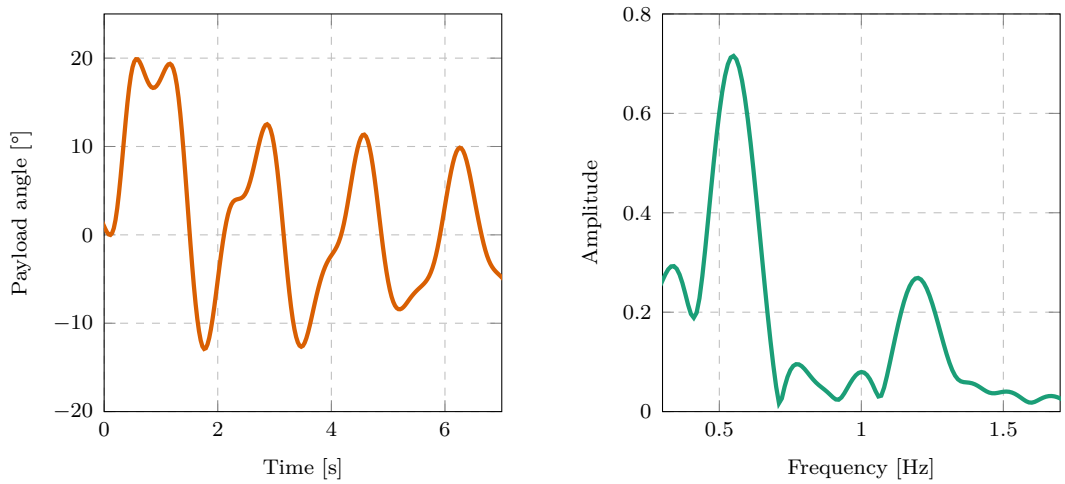
Note in Figure 7.3 that the estimation PE converges for each considered flight, even with wind speeds up to 6 m/s. Therefore a dominant oscillation frequency emerges from each flight, even when the multirotor is heavily affected by wind.

7.1.2. Dynamic payload cable length estimation

As discussed in Chapter 4, the dynamical equations of the white-box model are fixed in the a priori modelling phase. The model is then populated with values from parameter estimation techniques. However, when the dynamics of the observed system differ significantly from the pre-determined model, the parameter estimation algorithms still determine naive, best-fit values for the pre-determined model.

Figure 7.4: Practical flight with a suspended elongated payload attached to Honeybee

One of the a priori modelling assumptions mentioned in Section 4.3, is that the suspended payload is a point-mass. This reduced the considered suspended payload system to a simple pendulum in the white-box model. Figure 7.4 shows a photo of an elongated payload suspended from Honeybee during a practical flight This is a practical example



(a) Measured payload angle data. **(b)** FFT amplitude spectrum.

Figure 7.5: White-box model prediction for a North velocity step input for a dynamic payload ($m_1 = 0.2\,\text{kg}$, $l_1 = 0.5\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.6\,\text{m}$)

of a dynamic payload that deviates significantly from the point-mass assumption. The mass distribution causes a rotation of the payload relative to the suspended cable, which significantly affects the flight dynamics.

Figure 7.5a shows a snapshot of the payload angle data from a practical flight with a dynamic payload. Two superimposed frequencies are visible in the payload oscillations due to the double pendulum action of the elongated pendulum. The two peaks corresponding to these two frequencies can easily be identified from the FFT amplitude spectrum in Figure 7.5b. The cable length estimation method uses the frequency of the dominant peak and calculates the effective length corresponding to that frequency. This results in a simple pendulum model that best matches the dynamic payload oscillations.

Figure 7.6 shows the estimated cable length as a function of the length of training data for a practical dynamic payload. Note that the estimated length converges after a sufficient length of training data, showing that a dominant oscillation frequency can be identified. For this flight, the estimated cable length is 1.03 m.

Figure 7.7 shows two model predictions resulting from slightly different starting points in flight data. The two prediction runs in Figure 7.7 differ significantly from each other even though the starting points of the predictions are offset by only 0.06 s. Since the oscillations of the dynamic payload are irregular compared to the sinusoidal dynamics of the white-box model, the prediction accuracy is very sensitive to the initial condition.

Figure 7.8 shows the white-box model predictions for the dynamic payload data with slightly different starting positions in the data. Note how much the predictions differ even

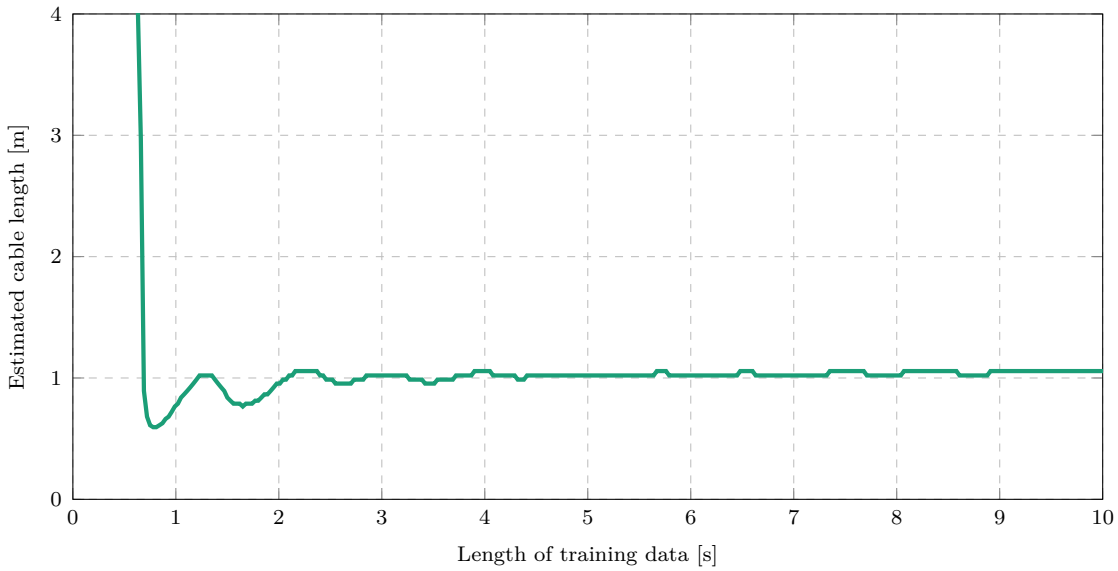


Figure 7.6: Estimated cable length as a function of length of training data for a dynamic payload ($m_1 = 0.2\,\text{kg}$, $l_1 = 0.5\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.6\,\text{m}$) .

though the starting points are so close together. This shows how sensitive the white-box model is to the initial condition of the prediction. This is because the white-box model consists of ordinary differential equations depend on the initial angular rate of the payload. Even though the oscillations of the dynamic payload have an approximate sinusoidal shape, the time derivative of the data differs significantly from the sinusoidal white-box dynamics. This is clear from numerous infliction points in the payload angle data shown in Figure 7.7.

However, as the size of the swing angles attenuate the relative oscillations of the elongated payload also decrease. Therefore the effect of the superimposed higher frequency oscillations become less prominent and the system dynamics approximate a simple pendulum more closely. For example, in Figure 7.7 it can be seen that the oscillations after 12 s are much less irregular than before. Therefore the simple pendulum model provides a decent representation of a practical dynamic payload for small swing angles. Overall, the white-box model represents the general shape of the practical data, but does not capture the transient response of the system and is very sensitive to initial conditions.

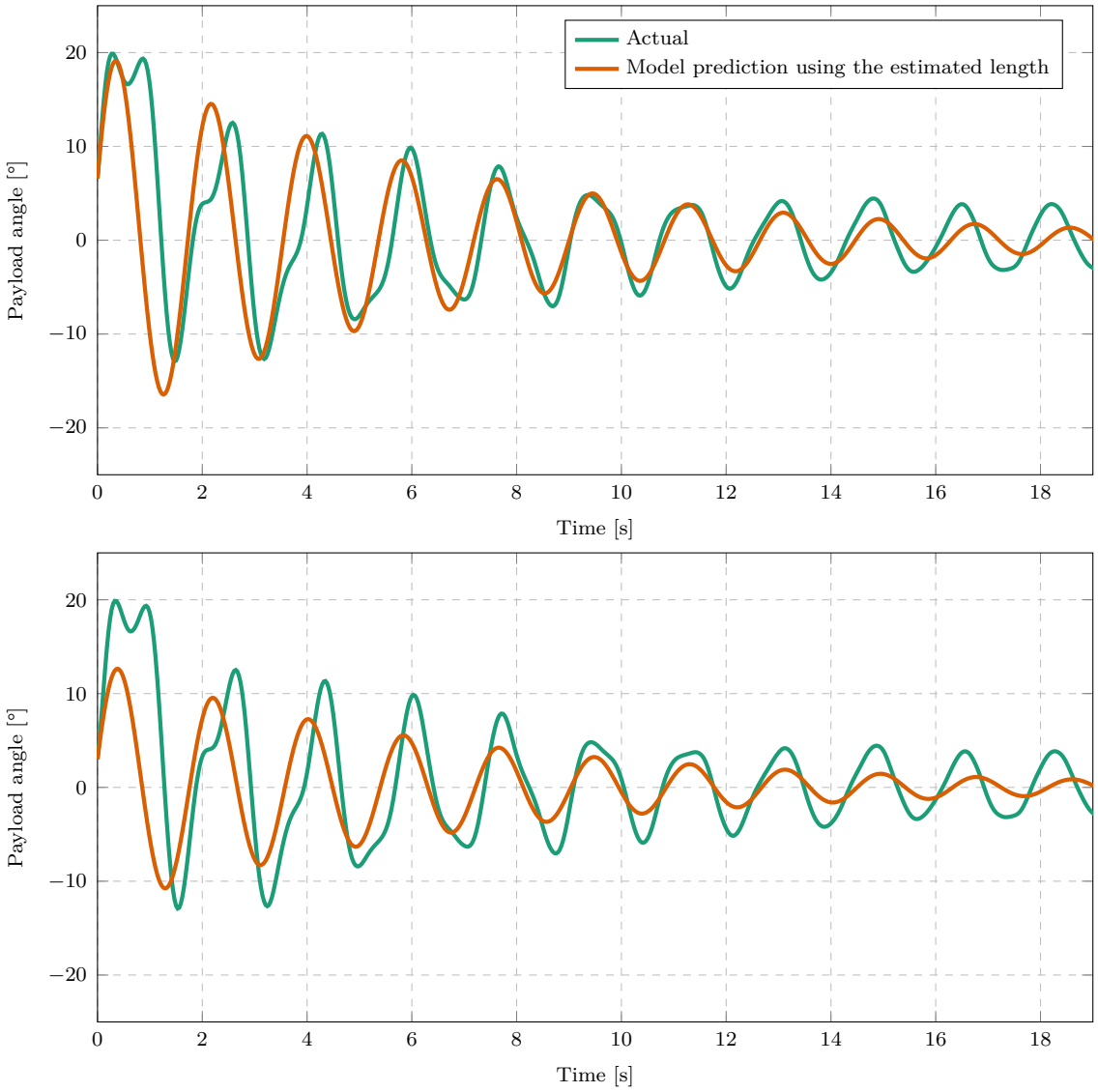


Figure 7.7: Data from a velocity step response with a dynamic payload ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 0.5\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.6\,\mathrm{m}$).

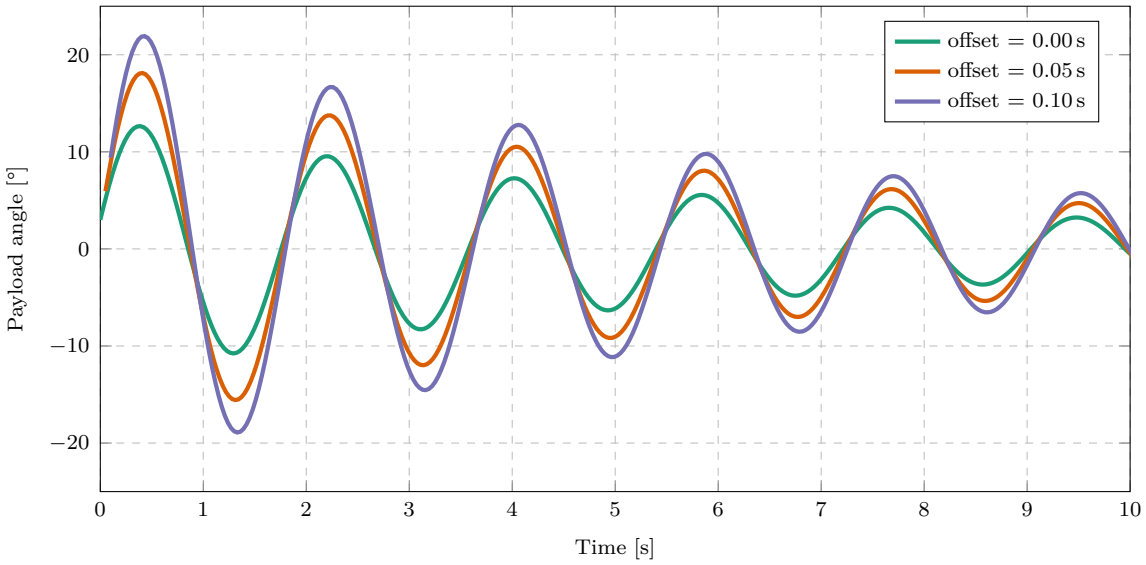


Figure 7.8: White-box predictions from different initial conditions for a dynamic payload ($m_1 = 0.2\,\text{kg}$, $l_1 = 0.5\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.6\,\text{m}$) .

7.2. Data-driven system identification with practical data

In Chapter 4 it was shown that the considered data-driven methods build accurate models of the system dynamics from simulation data. It was also shown in Chapter 3 that the simulation environment is a realistic representation of the practical system. However, there are still differences between simulations and practical flights. Therefore the data-driven algorithms will be tested with real flight data to evaluate their suitability for practical implementations.

7.2.1. Wind disturbance

The wind conditions during practical flights have a large influence on the quality of the flight data gathered. Wind adds an unmeasured disturbance to the considered system which is detrimental to system identification. This disturbance consists of a randomly fluctuating force applied to the vehicle, cable and payload. It is very difficult to model these forces accurately and to determine accurate drag coefficients of the practical system for realistic simulations.

The mean force applied to the multirotor by the wind affects the mean offset in acceleration setpoint data because the velocity controller integrators compensate for the disturbance. The mean offset is subtracted from the acceleration setpoint data, which results in a signal with a zero mean which is used for system identification. This accounts for the mean force applied by the wind. However, the wind speed also fluctuates from the mean randomly. This results in random process noise in the plant which cannot easily be removed from the measured data.

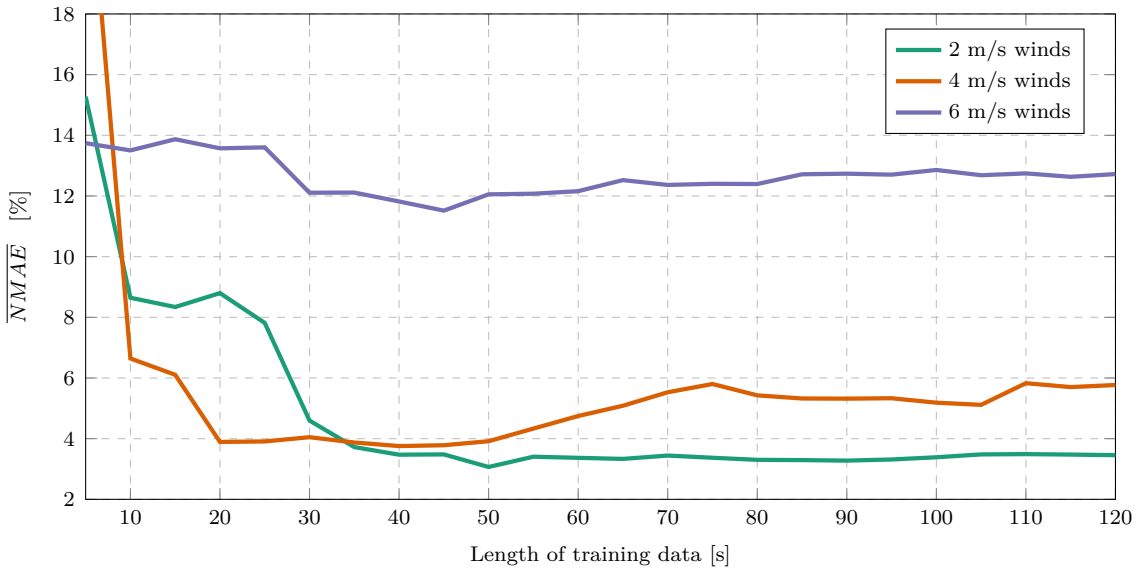


Figure 7.9: DMDc prediction errors as a function of length of training data for practical data with different wind conditions ($m_p = 0.2\,\text{kg}$, $l = 1\,\text{m}$, $T_s = 0.03\,\text{s}$).

Figure 7.9 shows prediction error as a function of training data length for different wind conditions. This plot shows that the minimum prediction error decreases with decreasing wind speeds. This is expected since lower wind speeds correspond to less process noise which is beneficial for system identification. Note that the prediction error corresponding to 6 m/s winds does not vary much with the length of training data. The prediction error at this wind speed is quite large and a model generated from such data will probably not be useful for control.

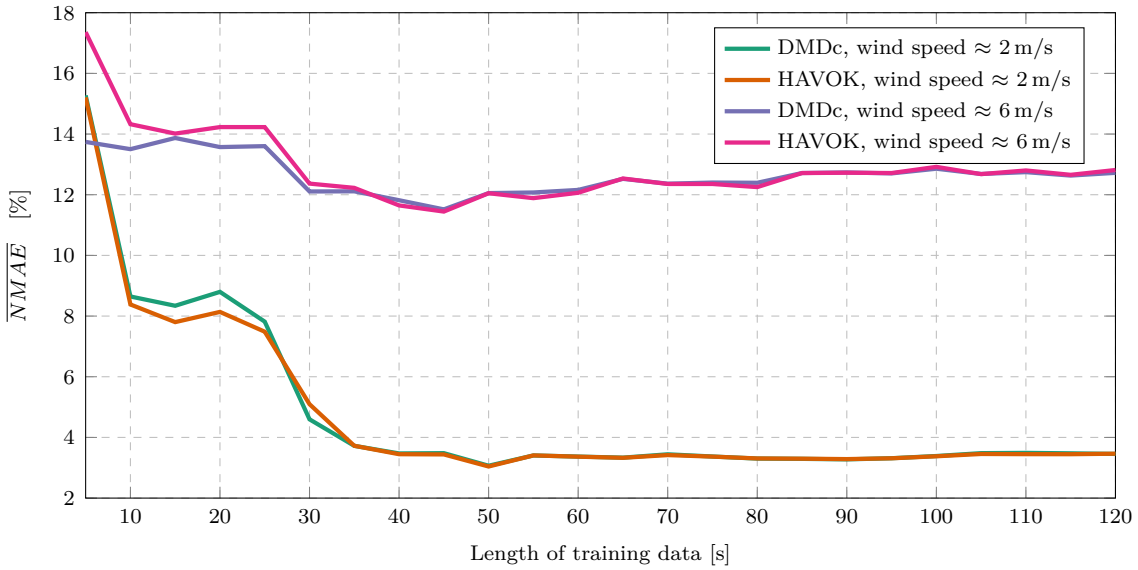


Figure 7.10: DMDc and HAVOKc prediction errors for different lengths of practical training data ($m_p = 0.2\,\text{kg}$, $l = 1\,\text{m}$, $T_s = 0.03\,\text{s}$).

Figure 7.10 compares the prediction errors of DMDc and HAVOKc models. This shows that the two techniques produce similar prediction errors for different wind conditions. The difference in prediction error is small and will probably not affect the performance of the controllers using these models. This affirms the observation in Chapter 4 that the minor difference in the algorithm implementations has a negligible effect on prediction accuracy. The DMDc implementation is therefore preferred over HAVOKc for practical data due to lower computational complexity.

7.2.2. Hyperparameters

As discussed in Section 4.6.3, the prediction error generally improves for a higher number of delay-coordinates because the number of parameters in the model increases. However, the prediction error reaches a Pareto optimum, after which the error does not significantly decrease with an increasing number of terms. Figure 7.11 shows the prediction error as a function of the number of delay-coordinates for practical flight data. Even though the Pareto elbow is not as smooth and clear as shown in Chapter 4, the elbow can still be identified.

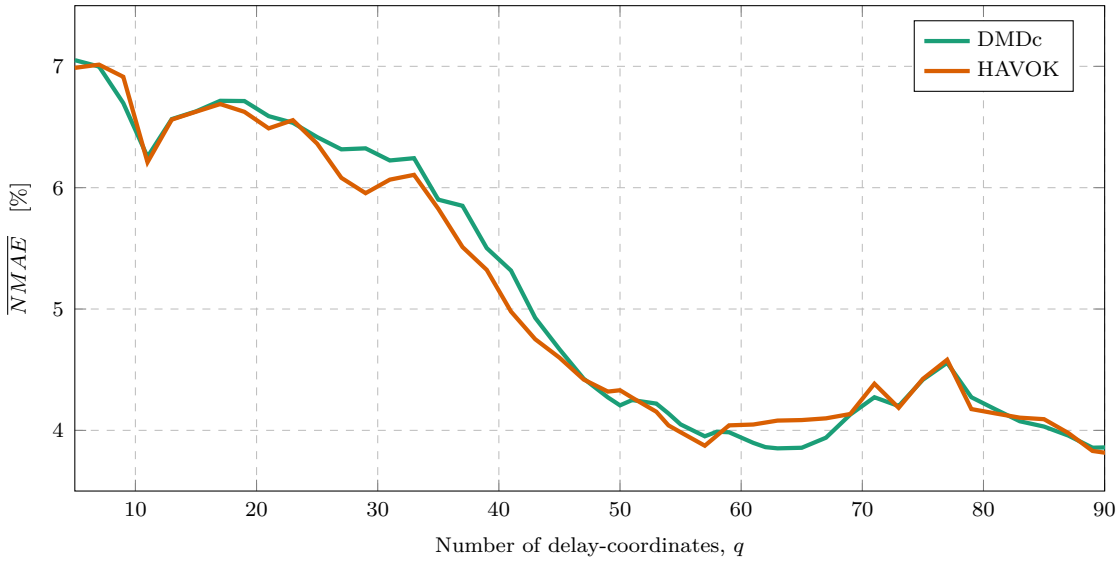


Figure 7.11: DMDc and HAVOKc prediction errors for different number of delays included in the model ($m_p = 0.2\,\text{kg}$, $l = 1\,\text{m}$, $T_s = 0.03\,\text{s}$, wind speed $\approx 2\,\text{m/s}$).

7.2.3. System parameters

It was shown with multiple simulations in Section 4.6.7 that the system identification methods work for a range of different payload parameters. Figure 7.12 shows the prediction error for different payloads with practical data. This shows that the proposed methods also work in practice with different payload configurations. The 'double-descent' trend

(discussed in Section 4.6.8) is also seen in the practical data results where the prediction error increases slightly after a specific length of training data.

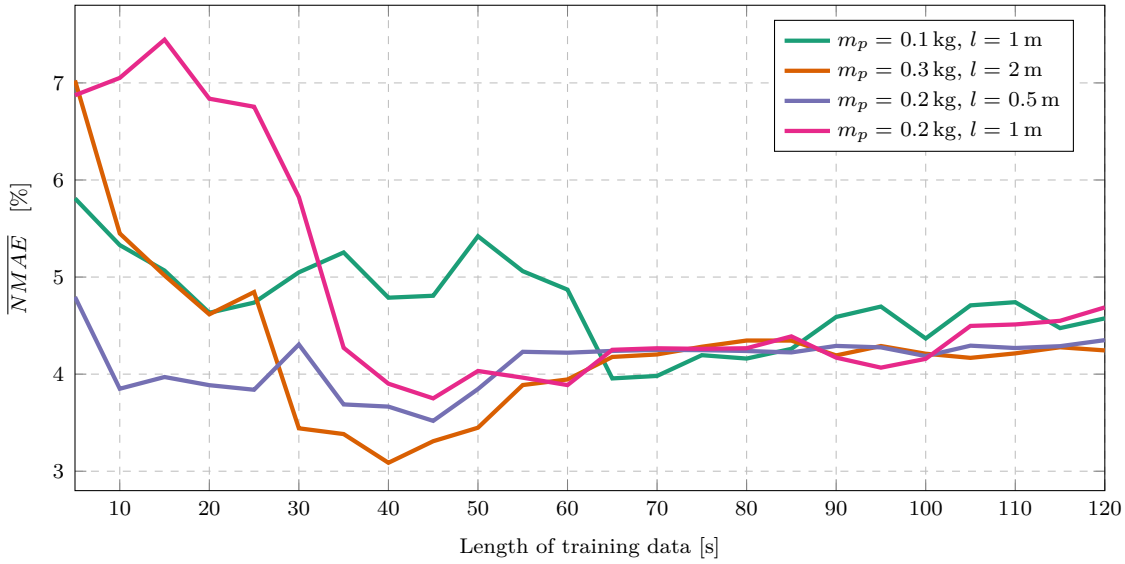


Figure 7.12: DMDc prediction error as a function of training data length for different payload parameters.

Recall that the models producing these predictions do not use a priori information about the plant. Only input and output measurements are used in the model generation. In contrast to the white-box technique, the effect of system parameters such as multirotor mass, payload mass, cable length, and damping coefficients are inherently included in the estimated model. Therefore these parameters can all be varied and the system identification algorithm should still be able to determine a prediction model of the system.

7.2.4. State predictions

Figure 7.13 shows the measured and predicted payload angle data of a suspended payload for a velocity step in a practical flight. Note that the model is generated from training data and is tested with a separate set of previously unseen, testing data. Figure 7.13 plots the state prediction against testing data and it is clear that the prediction data fits the measured data very well.

Recall from Section 4.4 that DMDc produces a discrete, state-space model in the form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_{dmd}\boldsymbol{x}_k + \boldsymbol{A}_d\boldsymbol{d}_k + \boldsymbol{B}_{dmd}\boldsymbol{u}_k. \tag{7.2}$$

The state prediction starts at the initial condition, $\boldsymbol{x}_0$ and $\boldsymbol{d}_0$, and predicts the state vector for each successive time-step, $\boldsymbol{x}_{k+1}$, from the state vector, $\boldsymbol{x}_k$, delay vector, $\boldsymbol{d}_k$ and input vector, $\boldsymbol{u}_k$ at the previous time-step. Each of these time-step predictions results in

a small error that accumulates with each successive time-step. Therefore the prediction error increases as the prediction horizon increases, as shown in Figure 7.13.

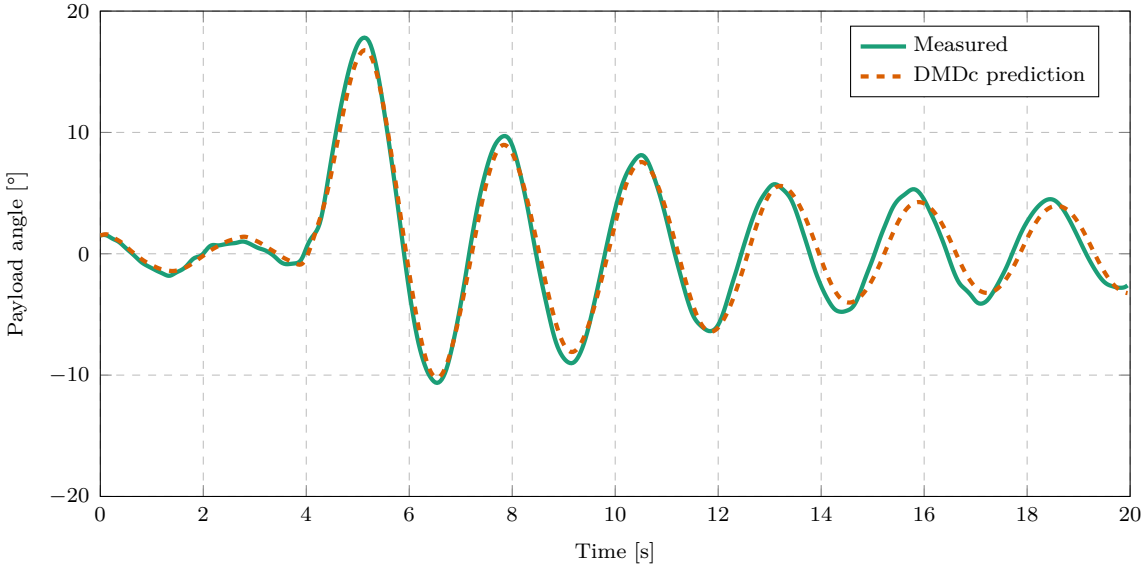


Figure 7.13: Model predictions of practical flight data with a suspended payload for a North velocity step input ($l = 2\,\text{m}$, $m_p = 0.3\,\text{kg}$).

Figure 7.14 shows the measured and predicted North velocity of the same flight. The oscillations in the velocity response due to the swinging payload are visible in this plot. The model predicts the frequency and size of these oscillations reasonably well. Note that predicting the payload angle is significantly easier than predicting the velocity response. The payload angle prediction inherently oscillates around a zero mean. However, the velocity response has a non-zero mean and depends on numerical integration of the acceleration setpoint data. A slight error in the correction of the setpoint offset (discussed in Section 4.6.6 and Section 7.2.1) may result in a large error in the velocity prediction due to a build-up of integration error. However, despite this challenge, the model accurately predicts the velocity step size of the practical data.

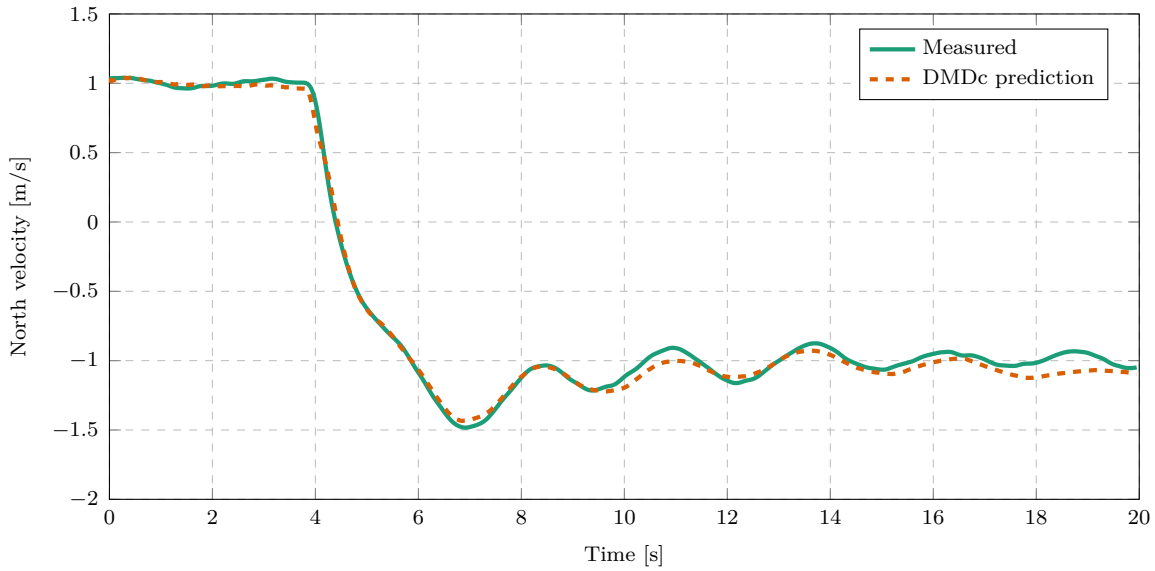


Figure 7.14: Model predictions of practical flight data with a suspended payload for a North velocity step input ($l = 2\,\text{m}$, $m_p = 0.3\,\text{kg}$, wind speed $\approx 0.5\,\text{m/s}$).

7.2.5. Extended dimensions

Because the data-driven methods are only dependant on the input and output data used, the prediction model can easily be extended to include more dimensions by adding more state measurement variables. In this section, a prediction model is generated and discussed which includes both the North and East axes dynamics. Such a model could be used in a single MPC velocity controller to damp the payload oscillations in both axes simultaneously.

For this model, the state vector is,

$$\boldsymbol{x} = \begin{bmatrix} V_N & V_E & \theta_N & \theta_E \end{bmatrix}^T, \tag{7.3}$$

and the corresponding input vector is,

$$\boldsymbol{u} = \begin{bmatrix} A_{N_{sp}} & A_{E_{sp}} \end{bmatrix}. \tag{7.4}$$

To generate training data, random steps are commanded in the North and East axes simultaneously to excite the dynamics in both axes. Figure 7.15 shows an example of the practical training data used for this extended dimension model. Clear oscillations in the payload angle and velocity response of both axes are visible. Figure 7.16 shows the state variable predictions of a DMDc model build from the data in Figure 7.15. It is clear that this model provides an accurate prediction of each state variable considered. This shows that the data-driven methods can be effectively extended to include both the North and

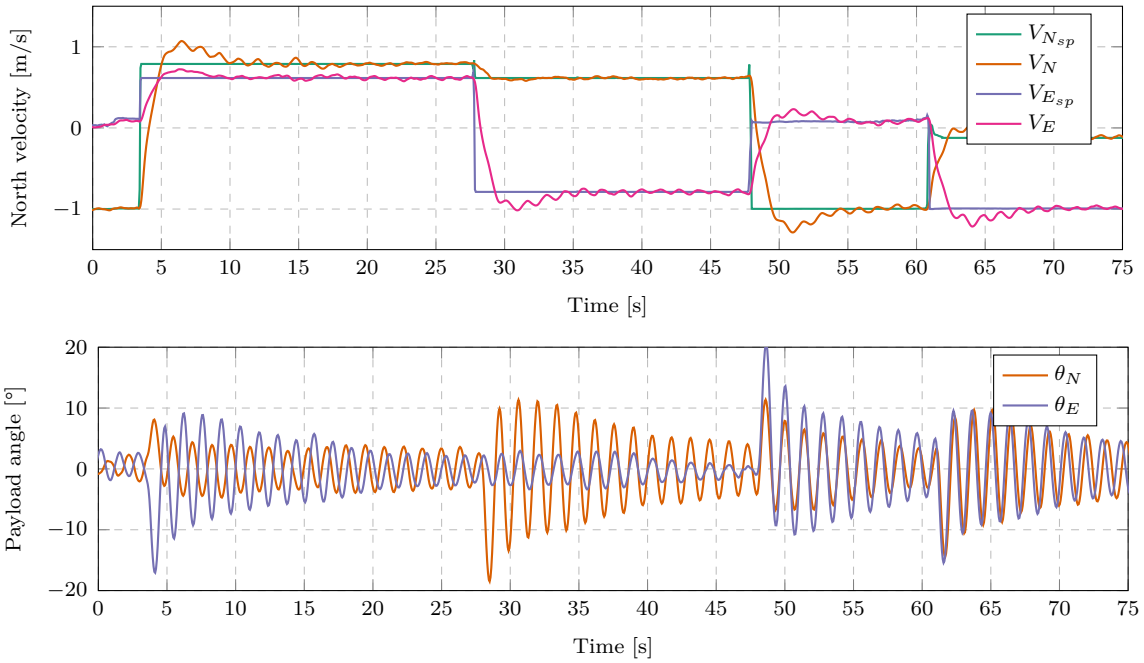


Figure 7.15: Snapshot of training data with random velocity step inputs for the North and East axes ($m_p = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$)

East axes dynamics. This is a great advantage of the proposed data-driven approach. The model is easily adapted for different use cases without redesigning estimation techniques or remodelling the plant manually.

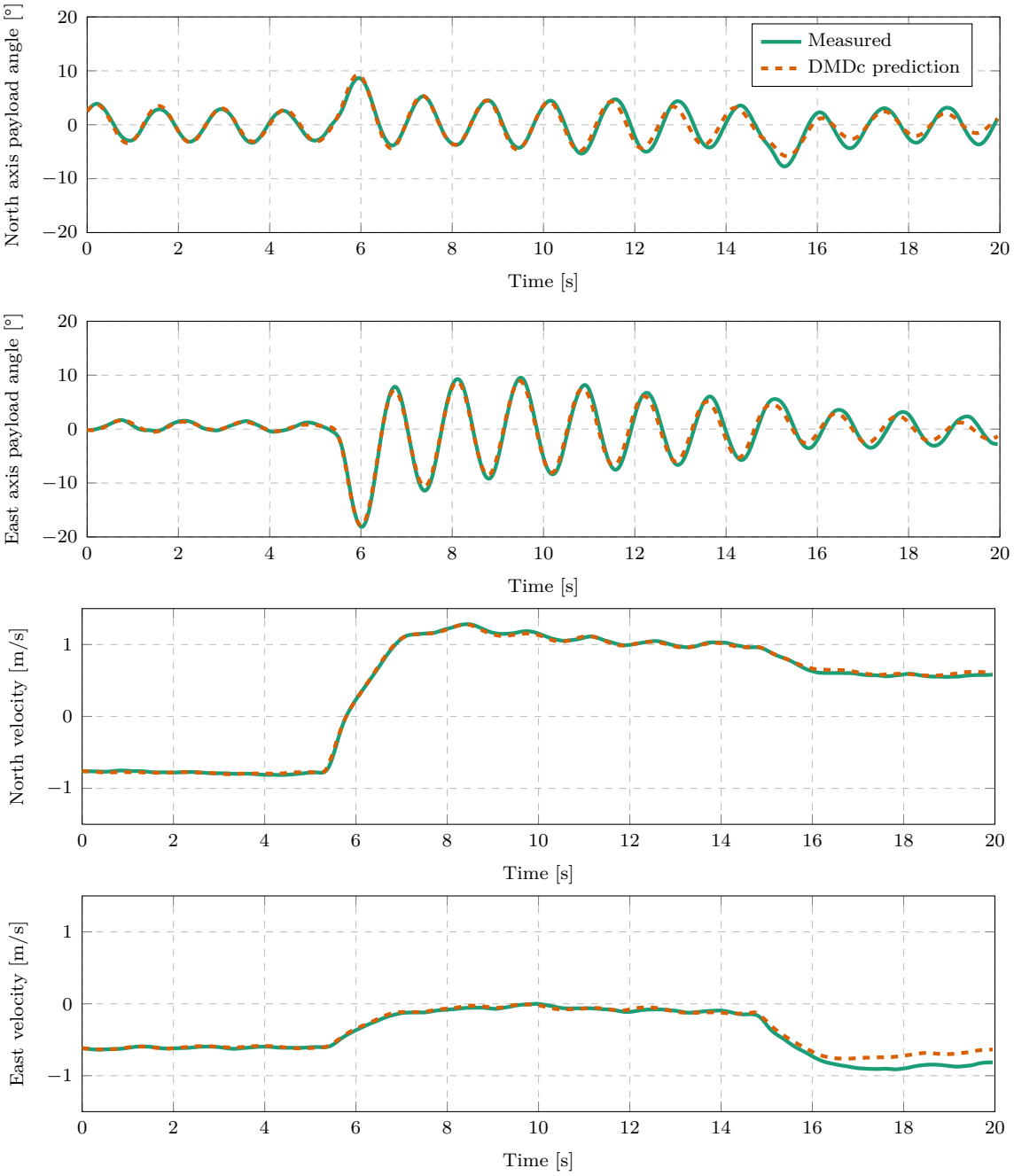


Figure 7.16: Data-driven predictions of practical data for a model with both North and East axis dynamics

7.2.6. Dynamic payload

As mentioned in Section 7.1.2, one of the disadvantages of the white-box system identification approach is that it relies heavily on a priori modelling assumptions. When a payload is attached to the multirotor that deviates from the white-box modelling assumptions, the performance of the system identification method decreases. However, the data-driven methods can handle this deviation because it does not rely on these assumptions.

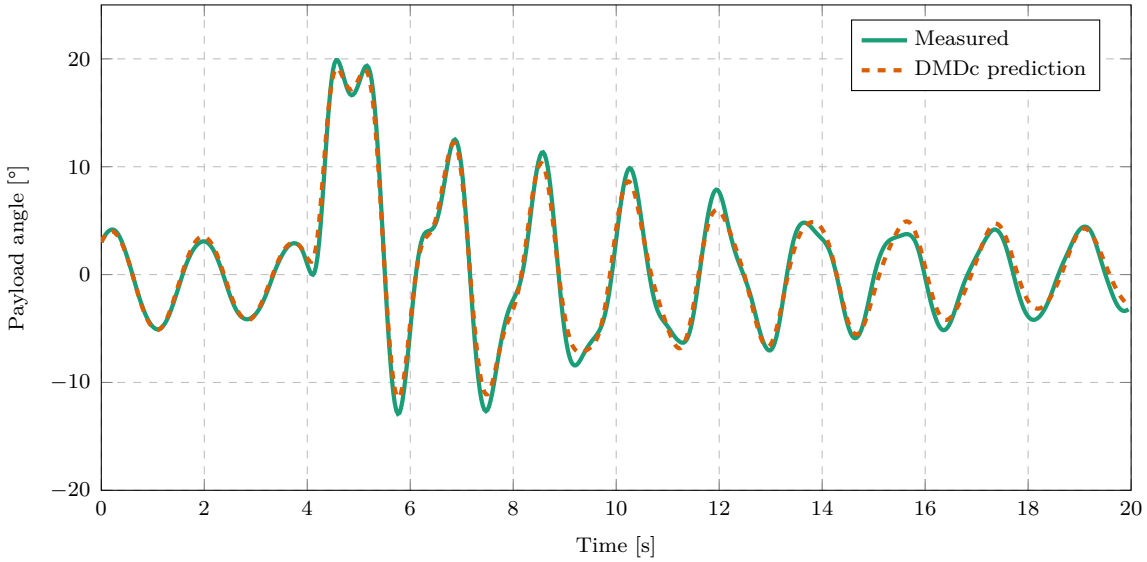


Figure 7.17: Practical flight data and model predictions with an elongated payload for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 0.5\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.6\,\text{m}$).

Figure 7.17 shows the measured and predicted payload angle of the practical dynamic payload. The irregular oscillations due to the double pendulum action of an elongated payload are visible in the angle data. It is clear that the data-driven model represents the actual payload dynamics for unseen testing data well. It appears that the prediction differs slightly from the measurement data at the peaks, however, this does not appear to be a significant error. Overall, it is clear that the DMDc model captures the multi-frequency oscillations of the dynamic payload well.

Figure 7.18 shows the measured and predicted velocity of the same flight. The superimposed frequencies are not as visible in the velocity oscillations as they were in the payload angle data. However, the oscillations in the velocity response still appear irregular compared to the simple payload data shown in Figure 7.14. In Figure 7.18, the size of the velocity prediction deviates from the measurement data at the velocity overshoot. However, this error does not appear significant enough to affect the corresponding MPC controller. The shape of the velocity oscillations also appears to be captured well in the prediction.

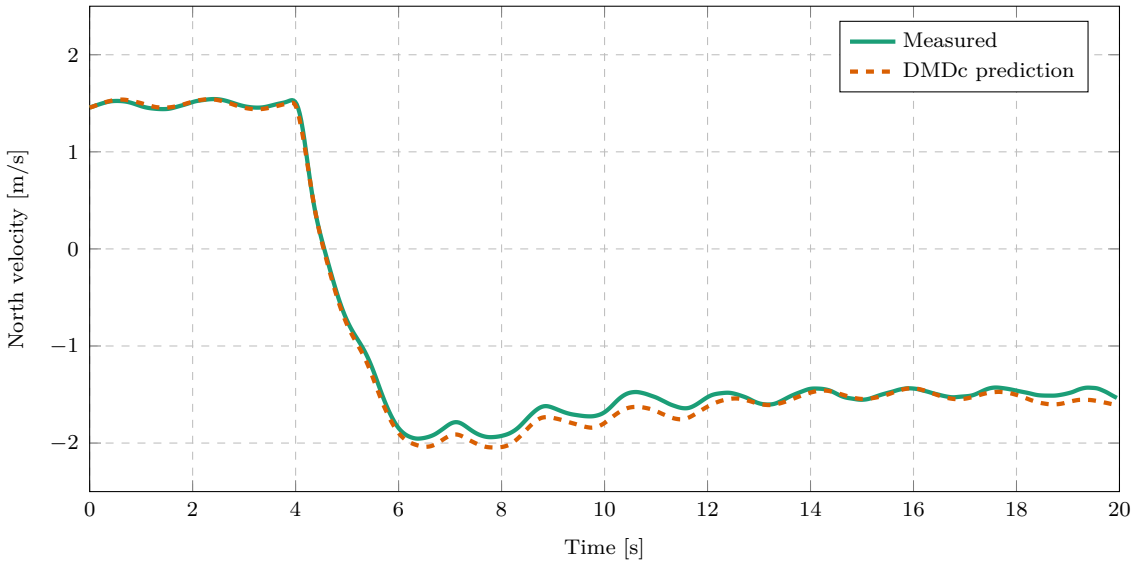


Figure 7.18: Practical flight data and model predictions with an elongated payload for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 0.5\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.6\,\text{m}$).

Recall that the prediction is propagated from an initial condition using the given input data only. The model does not use state measurements to readjust after the initial condition is taken. Therefore an accumulation in prediction error is expected as the prediction horizon increases. Because the model prediction matches the shape of the practical testing data so closely, it is expected that this model can be used for a practical MPC implementation on practical data.

7.3. Hardware-in-the-Loop simulations

In Section 7.2 it was shown that DMDc can generate accurate state prediction models from practical flight data. It was also shown in Section 5.4 that the DMDc models can be used in an MPC for effective swing-damping control. A HITL simulation can now be used to test the complete control architecture with the final software running on the actual hardware.

As described in Section 6.3, an OBC runs a ROS node which implements the MPC algorithm. The MPC is based on a DMDc plant model generated from training data from a HITL simulation. The OBC receives state feedback and sends control signals via a Universal Asynchronous Receiver Transmitter (UART) connection to the FC. The FC runs the actual PX4 flight stack firmware and executes the control signals received from the MPC node. Sensor values are generated and sent to PX4 by the Gazebo simulator, which runs on a desktop computer connected to the FC via Universal Serial Bus (USB). In this way, we can safely determine whether the practical hardware is suitable for the computational complexity of the control algorithms.

In this section, the results of different HITL simulations will be shown and discussed. The effect of the MPC sample time on the Central Processing Unit (CPU) consumption will also be investigated. Finally, it will be shown that an MPC node, which is based on a data-driven system identification model, effectively implements swing-damping control in a HITL simulation.

7.3.1. Effect of hyperparameters on computational requirements

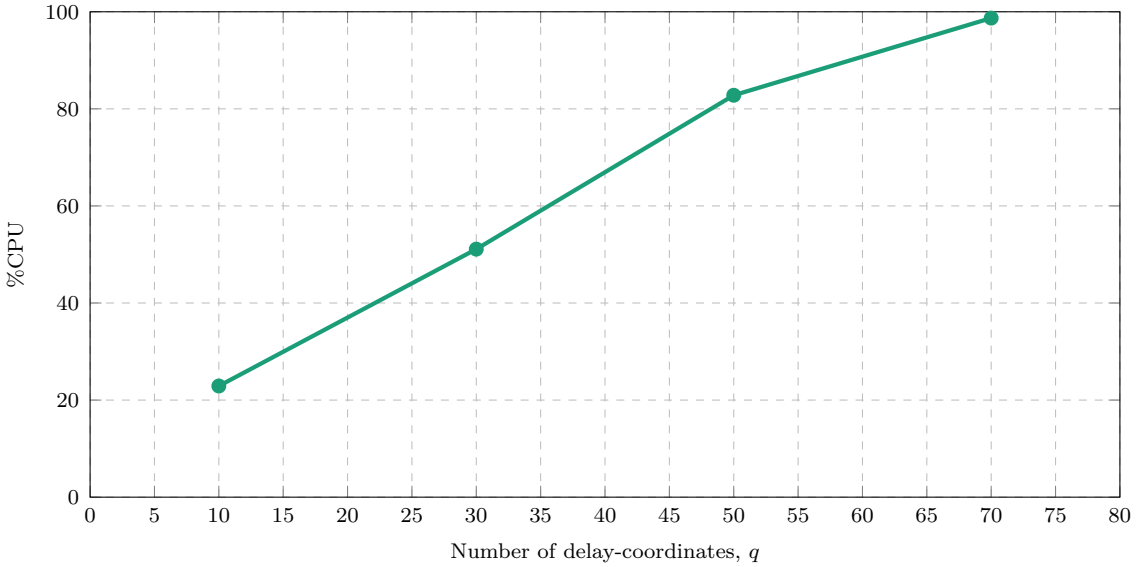


Figure 7.19: Maximum %CPU used by the MPC node for different values of q ($Ts = 0.03\,\mathrm{s}$).

As discussed in Section 5.4, the computational complexity of the MPC optimisation problem increases for larger state-space matrices, and the hyperparameter, q, determines the size of these matrices. Figure 7.19 shows the %CPU consumption of MPC nodes based on models with different q values. The %CPU value represents the average percentage of CPU time used by the MPC node running on the OBC. From Figure 7.19 it is clear that %CPU increases with increasing values of q. This shows that the computational complexity increases for larger values of q.

In Section 4.6.3, it was shown that $q = 50$ is near the Pareto optimum for the practical data and the prediction accuracy does not increase significantly for $q > 50$. As shown in Figure 7.19, an MPC node with $q = 50$ and $T_s = 0.03\,\mathrm{s}$ run at %CPU = 82.8% on the OBC. The status of the QP solver was also monitored, which showed that the optimisation problem was consistently solved within the given optimisation time for $q = 50$. The MPC node achieves stable, swing-damping control with this model. However, for $q = 70$ the MPC node uses %CPU = 98.7% and the optimisation problem can not be solved

fast enough for stable multirotor control. This results in an unstable controller and the multirotor-payload system crashes consistently with this MPC node.

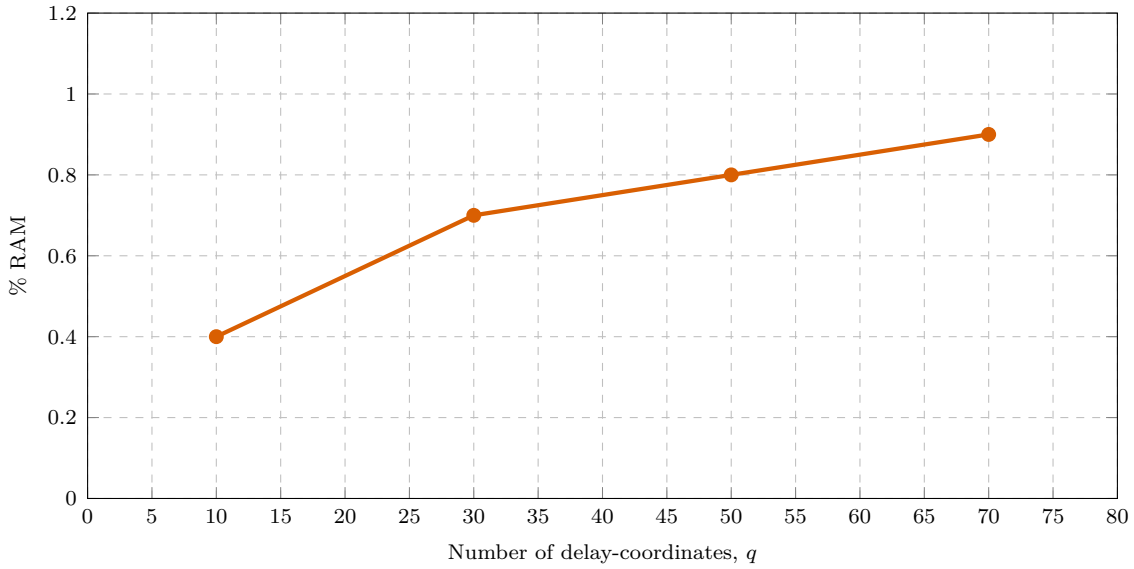


Figure 7.20: Maximum % RAM used by the MPC node for different values of q.

Figure 7.20 shows that the %RAM also increases for larger values of q. The %RAM value represents the maximum percentage of Random Access Memory (RAM) space used by the MPC node while running on the OBC. It is clear that the OBC has sufficient memory to handle the MPC, since the MPC node uses less than 1% RAM, even for large models with $q = 70$.

7.3.2. Effect of sample time on computational requirements

The sample time of the controller also affects the %CPU consumption of the MPC node. Figure 7.21 shows the measured %CPU of MPC nodes with different sample times using a plant model with $q = 50$. It is clear that %CPU decreases as T_s increases. The default velocity controller in PX4 runs at 50 Hz, which corresponds to $T_s = 20$ ms. However, an MPC node with $T_s = 20$ ms struggles to run fast enough on the given OBC. This is clear from the high %CPU consumption (98.6%) of the MPC node running at $T_s = 20$ ms. This node results in unstable control because the QP problem cannot be solved within the given optimisation time.

An MPC node with $T_s = 40$ ms runs with a low %CPU, however it also results in unstable control. This is because the controller frequency is too low to provide adequate velocity control of the multirotor-payload dynamics. However, an MPC node with $T_s = 30$ ms provides stable control of the multirotor-payload system. The QP problem is solved within

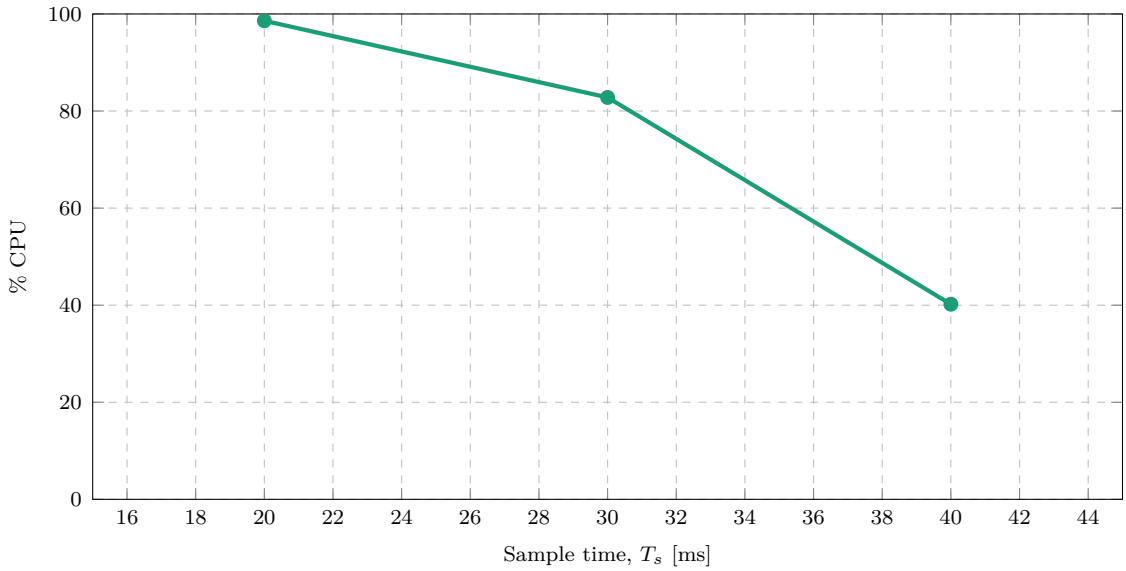


Figure 7.21: Maximum % CPU used by the MPC node for different sample times ($q = 50$).

the given optimisation time and, as shown in the section below, the resulting controller frequency of 33.33 Hz appears to be fast enough to control the multirotor-payload dynamics.

7.3.3. Velocity step response

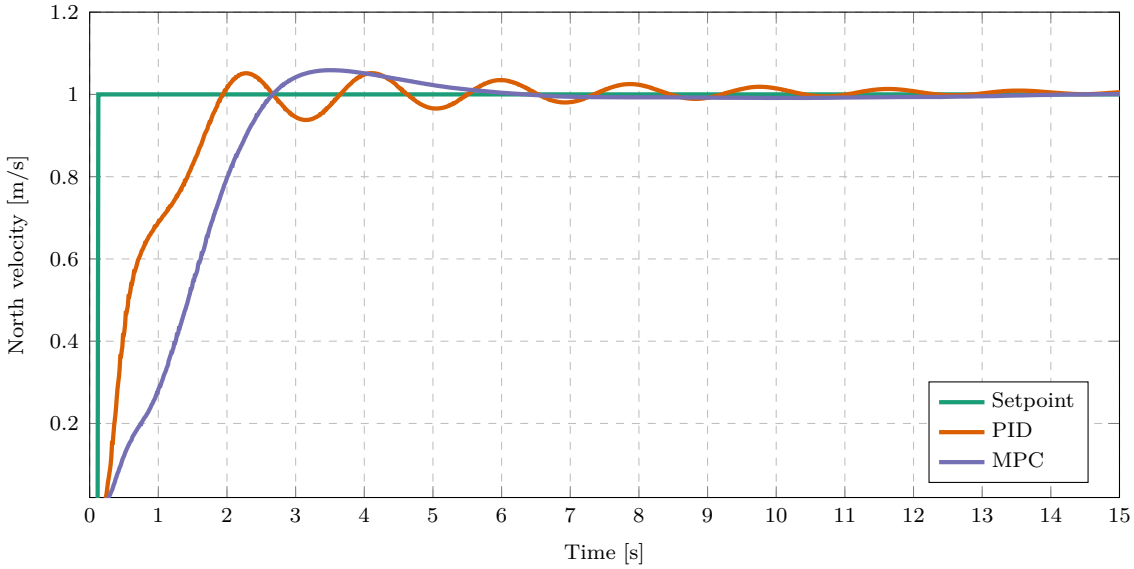


Figure 7.22: Velocity step responses of MPC and PID controllers for HITL simulations ($q = 50$, $T_s = 0.03\,\mathrm{s}$).

Figure 7.22 plots the velocity responses of the PID and MPC controllers in HITL simulations of the multirotor-payload system. The MPC controller is based on a system identification model with $q = 50$ and a sample time of $T_s = 30\,\mathrm{ms}$. It is clear from this plot that the MPC node running on the OBC provides stable control of the multirotor payload system and damps the velocity oscillations well.

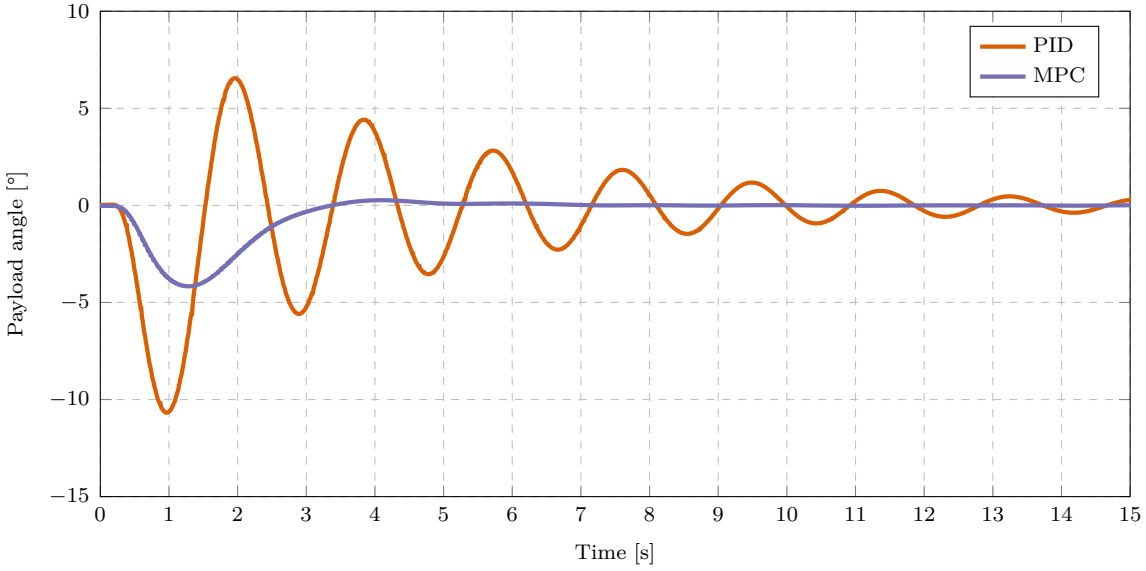


Figure 7.23: Payload angle responses of MPC and PID controllers for HITL simulations ($q = 50$, $T_s = 0.03\,\mathrm{s}$).

Figure 7.23 shows the payload angle data of the PID and MPC controllers during the velocity step response. From this plot, it is visible that the MPC controller damps the payload angle well. Overall, these plots show that the swing-damping control achieved in simulations in Section 5.4, is also achievable with the final controller software running on the actual hardware.

7.4. Summary

In this chapter, the system identification techniques were applied to practical flight data. The cable length estimation technique described in Chapter 4 was applied to data from flights with different payload masses and cable lengths. The results showed that the estimated length converged quickly for each payload configuration and that the estimation error was consistently quite large. The large error was attributed to cable attachment being below the vehicle CoM, and the damping effect of the controller, which were not considered in the parameter estimation algorithm.

It was also shown that the estimated length converged for flight data with different wind conditions and that the estimation error was the largest for the data with the most wind. Furthermore, the cable length estimator was applied to flight data with a double-pendulum payload. The estimated value converged to a length corresponding to the dominant frequency in the data, and the resulting white-box model prediction only captured the general shape of the data well. However, the white-box model prediction was shown to be very sensitive to initial conditions.

The data-driven system identification techniques were also successfully applied to practical flight data. As expected, the prediction error of the techniques decreased for flights with lower wind speeds. With a reasonable amount of wind, the techniques were able to generate accurate prediction models. However, at very high wind speeds, the resultant models were not usable.

The DMDc and HAVOKc state predictions performed equally well on data from various flights. DMDc was chosen as the preferred method due to its lower complexity. It was shown that DMDc could be applied to data from a wide range of different payload parameters and that it consistently generated accurate prediction models without prior knowledge of the payload configuration. The Pareto front and ‘double-descent’ trends discussed in Chapter 4 were also identified in these results.

DMDc was also extended to capture the North and East velocity dynamics simultaneously, and the resulting prediction model accurately reconstructed the system dynamics for test

data. Furthermore, DMDc was applied to flight data with a dynamic payload representing a double-pendulum system. DMDc accurately reconstructed the dynamics of this system for unseen test data. Unlike the white-box model which only represented the dominant oscillation frequency, the DMDc model also predicted the irregular oscillations well.

Finally, the full data-driven system identification with MPC control architecture was tested in HITL simulations. It was shown that the hardware successfully executes the MPC algorithm at the desired speed for the selected hyperparameters and sample time. The successful velocity step response proved that the various software systems work together seamlessly. It also showed that the control architecture can be implemented on the hardware of an actual multirotor to successfully apply swing-damping control of the multirotor-payload system.

Chapter 8

Conclusion

This thesis considered the design and practical implementation of a stabilising control architecture for a multirotor with an unknown suspended payload. A broad scope was considered which includes two different area of research, namely,

- Data-driven system identification of the unknown payload dynamics
- Optimal swing damping control of the multirotor-payload system

The content and outcome of this work will be discussed in this chapter.

8.1. Literature study

Existing solutions for stabilised multirotor control with a suspended payload were identified in the literature. The literature study showed that research seldomly includes experimental results or algorithm testing on practical hardware, even though this would provide valuable insight. It also showed that most studies do not account for uncertainty in the controlled system. A thorough study of the literature showed that some solutions account for parameter uncertainty, but very few assume no knowledge of the payload dynamics.

Furthermore, the few studies that achieved stabilised control despite unknown dynamics, counteracted the payload effect as an unknown disturbance instead of actively controlling the payload state. This places the focus on robustness rather than smooth control of the complete multirotor-payload system.

An LQR controller was identified as a popular baseline controller in the literature and was selected as the baseline swing damping controller for this work. The specific LQR implementation considered in this work is based on a previous study that only considers parameter uncertainty. This LQR controller is based on a linearised, predetermined model of the multirotor-payload system. The payload mass and cable length are unknown prior to a flight and are estimated with RLS and FFT estimators respectively.

8.2. System identification

The baseline parameter estimation technique was described and applied to data from SITL simulations with Gazebo. It was shown that the white-box model which uses the estimated parameters captured the general shape of the system state predictions well. The white-box model with parameter estimation technique was also applied to a dynamic payload simulation. An elongated payload was suspended from the multirotor and acted as a double pendulum, inducing irregular oscillations in the system. For this use case, the resultant white-box model predictions did not represent the general shape of the payload dynamics.

DMDc and HAVOKc were introduced as the data-driven system identification techniques proposed by this work. These linear regression techniques each produce a discrete, linear space-space model of the considered dynamics based on input and output data only. The conventional HAVOK is not designed to be applied to controlled systems. However, this algorithm was extended in this work to account for control inputs in a dynamical system and will be referred to as HAVOKc. The conventional DMDc algorithm was altered to include delay-coordinates in a similar way to HAVOK. Furthermore, the mathematical complexity of these techniques was described in detail.

These algorithms were applied to multiple SITL simulations for testing. Data was generated by tracking a sequence of random velocity step inputs with the standard PID controllers from PX4. This data was split into training and testing sets. The algorithms could then be trained on the set of training data and could be validated on the unseen testing set. The prediction accuracy of each model produced by these techniques was quantified with an NMAE error metric. This metric is based on multiple model prediction runs from different initial conditions over a specified time horizon.

A hyperparameters search showed a Pareto elbow as a function of the number of delay-coordinates, q, such that increasing q passed this elbow does not significantly increase the model accuracy, but does increase model complexity. Furthermore, the 'double-descent' phenomenon was identified when testing with various lengths of training data. It was consistently observed in different experiments that increasing the length of training data past a specific point decreases the prediction accuracy. This is unintuitive because longer lengths of training data are expected to increase model accuracy by reducing overfitting.

Both techniques were shown to be robust to measurement noise. It was also shown that the techniques consistently produced accurate models with a range of different system parameters. The techniques were also tested with the dynamic pendulum and the prediction

models accurately captured the irregular oscillations, despite having no prior knowledge of the payload. This showed a major improvement compared to the white-box model. For the range of different tests the prediction accuracies of DMDc and HAVOKc models were similar, hence DMDc is preferred due to lower computational complexity.

8.3. Swing damping controllers

Furthermore, the different controllers were discussed and tested. The cascaded PID controller was described and the gains of each control loop were tuned in simulation. This simulation environment was verified with practical data from the Honeybee multirotor and was shown to be an accurate representation of the actual system.

The baseline LQR controller was also described and the weights were tuned for the simulated multirotor-payload system. The LQR was designed based on the white-box model which uses estimated parameters for each different payload. The control architecture proposed in this work includes an MPC controller which uses a data-driven system identification model for predictive control. The MPC implementation from the Model Predictive Control Toolbox™ in Simulink™ was used and the algorithm was described in detail. This controller, using data-driven system identification models, was also successfully applied for swing damping control in simulation.

Numerous tests were performed to evaluate the combined system identification and control architectures. For each test, the system identification stages firstly produced the parameter estimates and identified model for the LQR and MPC approaches respectively. Thereafter, each controller was applied based on those results. For a payload with a simple pendulum model, the LQR and MPC both achieved stabilised control resulting in near swing free motion. The controllers showed similar swing damping performances, but the MPC produced a faster settling time. The control architectures consistently achieved swing damping control with different payload masses and cable lengths, despite parameter uncertainty for the LQR approach and no prior payload knowledge for the MPC approach. Both controllers also showed acceptable disturbance rejection during a constant unknown step input force to the multirotor.

The control approaches were also tested for the dynamic payload case. Despite the data-driven model showing a much better prediction accuracy than the white-box model, the LQR and MPC approached produced similar swing damping performances. It was shown that even though the data-driven model is accurate in the domain of the state and input vectors considered in training, the optimised trajectory of the MPC goes beyond that domain. Hence, the model approximation is inaccurate and the resulting MPC control is

suboptimal. Both controller responses were not as smooth as for the simple pendulum model. However, both controllers showed stabilised swing damping control of the system and reduced system oscillations quickly.

The controller architectures were also applied to a simulation where a different multirotor mass was used. The LQR controller induced undesirable, high-frequency oscillations in the system response due to the model inaccuracy. This is because the baseline approach only considers parameter uncertainty in the payload mass and cable length. However, the MPC approach produced the same swing-free motion shown in previous simulations and clearly outperformed the LQR approach. The proposed control architecture does not rely on prior knowledge of the system dynamics, therefore the architecture can adapt to unconsidered system changes without redesigning the implementation.

8.4. Practical implementation

For experimental work, the hardware and software toolchains were described. Numerous practical flights were performed with different payload masses, cable lengths, and with a dynamics payload. All conclusions made with simulation data for the system identification methods were verified with practical flight data. This further validated that the simulation environment is a good representative of the actual system. This also showed that DMDc produces accurate prediction models with practical levels of wind disturbances and measurement noise.

Finally, HITL simulations were performed to demonstrate the computationally intensive controller algorithm working with the actual hardware and software toolchain. This involved a complex system of different, interconnected software tools. The MPC algorithm ran as a standalone ROS node on the OBC and was generated from Simulink™ code. Tests showed that the OBC was acceptable for the processing requirements of the MPC and the algorithm could run at the desired speed. It was also shown that the final system produces smooth, swing damping control of the multirotor-payload system as seen in previous simulations.

Overall, it was shown that the full data-driven system identification with MPC control architecture produces swing damping control of the multirotor-payload system despite having no prior knowledge of the payload dynamics. It was demonstrated that this control architecture works for various configurations with different system parameters and even with a dynamic payload. Furthermore, it was demonstrated that an accurate data-driven prediction model could be determined from practical flight data with wind disturbances and measurement noise. It was also demonstrated that the available hardware fulfils the

computational requirements of the proposed algorithms. Finally, it is concluded that the proposed control architecture is practically feasible for stabilised control of a multirotor and suspended payload with unknown dynamics.

8.5. Recommended future work

The current work can be continued to further show the practical feasibility of this approach. It can also be extended to improve the control performance or to solve different control problems with the same approach. Recommendations for future work include:

- Perform practical flights with the MPC to demonstrate the practical performance of the controller.
- Perform practical flights with a rigidly attached container with sloshing fluid to demonstrate the adaptability of this approach to unknown dynamics.
- Test the current solution using obstacle avoidance trajectories and manoeuvres that require the payload to follow a specific trajectory instead of a simple non-swing reference.
- Add a continuous excitation functionality to continue training the system identification model during flight and possibly adapt to time-variant uncertainty.
- Quantify the domain represented by the state and input vectors considered in the training data. This could be used to adjust the input signal during training to cover a larger domain and ultimately train a model that better represents the system dynamics. This would hopefully improve the controller performance with unknown dynamical models.
- Compare the proposed control architecture to a non-linear MPC using a non-linear data-driven model like Sparse Identification of non-linear Dynamics (SINDy)

Bibliography

[1] Compare Commander, "Drones For Search & Rescue Missions," 2020. [Online]. Available: https://comparecommander.com/drones-for-search-rescue-missions/

[2] M. Ireland, "Investigations in Multi-Resolution Modelling of the Quadrotor Micro Air Vehicle," Ph.D. dissertation, University of Glasgow, 2014.

[3] Z. Wang, J. Qi, C. Wu, M. Wang, Y. Ping, and J. Xin, "Control of Quadrotor Slung Load System Based on Double ADRC," in *39th Chinese Control Conference (CCC)*. IEEE, 2020, pp. 6810–6815.

[4] P. K. Muthusamy, M. Garratt, H. R. Pota, and R. Muthusamy, "Realtime Adaptive Intelligent Control System for Quadcopter UAV with Payload Uncertainties," *Transactions on Industrial Electronics*, pp. 1–1, 2021.

[5] A. P. Erasmus and H. W. Jordaan, "Stabilization of a Rotary Wing Unmanned Aerial Vehicle with an Unknown Suspended Payload," Master's Thesis, Stellenbosch University, 2020.

[6] J. F. Slabber and H. W. Jordaan, "Vision-Based Control of an Unknown Suspended Payload with a Multirotor Unmanned Aerial Vehicle," Master's Thesis, Stellenbosch University, 2020.

[7] PX4, "PX4 Autopilot User Guide," 2021. [Online]. Available: https://docs.px4.io/master/en/

[8] A. Miller and M. Rybczak, "Methods of controller synthesis using linear matrix inequalities and model predictive control," *Scientific Journals of the Maritime University of Szczecin*, vol. 43, pp. 22–28, 2015.

[9] NVIDIA, "Jetson Store." [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/jetson-store/

[10] P. R. Grobler and H. W. Jordaan, "Automated Recharging and Vision-Based Improved Localisation for a Quadrotor UAV," Master's Thesis, Stellenbosch University, 2020.

[11] M. Nakamura, K. Takaya, H. Ohta, K. Shibayama, and V. Kroumov, "Quadrotor Modeling and Simulation for Industrial Application," in *23rd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2019, pp. 37–42.

[12] B. Vergouw, H. Nagel, G. Bondt, and B. Custers, "Drone Technology: Types, Payloads, Applications, Frequency Spectrum Issues and Future Developments," in *The Future of Drone Use*. T.M.C. Asser Press, 2016, pp. 21–45.

[13] J. Vanian, "First FAA-approved drone delivery mission a success - Fortune," 2015. [Online]. Available: https://fortune.com/2015/07/17/faa-drone-delivery-amazon/

[14] J. Hidalgo, "Flirtey starts taking preorders for its drone delivery technology," 2021. [Online]. Available: https://www.rgj.com/story/news/money/business/2021/03/04/flirtey-starts-taking-preorders-its-drone-delivery-technology/6913942002/

[15] Wing, "Canberra - Australia," 2021. [Online]. Available: https://wing.com/australia/canberra/

[16] L. M. González-deSantos, J. Martínez-Sánchez, H. González-Jorge, and P. Arias, "Active UAV payload based on horizontal propellers for contact inspections tasks," *Measurement*, vol. 165, p. 108106, 2020.

[17] B. Suthar and S. Jung, "Design and Feasibility Analysis of a Foldable Robot Arm for Drones Using a Twisted String Actuator: FRAD-TSA," *Robotics and Automation Letters*, vol. 6, no. 3, pp. 5769–5775, 2021.

[18] H. Wolf, "The Race For Drone Delivery Just Got A New Leader: 3 Key Differences That Will Define Who Wins," 2020. [Online]. Available: https://www.forbes.com/sites/worldeconomicforum/2020/09/05/the-race-for-drone-delivery-just-got-a-new-leader-3-key-differences-that-will-define-who-wins/?sh=19bae1114d7e

[19] Flirtey, "Flirtey is the premier independent drone delivery service." [Online]. Available: https://partechpartners.com/companies/flirtey/#:$\sim$:text=Flirteyisthepremierindependent,%2Cretailande-commerce.

[20] N. Tardella, "Earthbound Robots Today Need to Take Flight - IEEE Spectrum," 2016. [Online]. Available: https://spectrum.ieee.org/earthbound-robots-today-need-to-take-flight

[21] K. T. San, S. J. Mun, Y. H. Choe, and Y. S. Chang, "UAV Delivery Monitoring System," in *MATEC Web of Conferences*, vol. 151. EDP Sciences, 2018, p. 04011.

[22] B. Min, J.-H. Hong, and E. Matson, "Adaptive Robust Control (ARC) for an altitude control of a quadrotor type UAV carrying an unknown payloads," in *11th International Conference on Control, Automation and Systems*, 2011.

[23] B. J. Emran, J. Dias, L. Seneviratne, and G. Cai, "Robust adaptive control design for quadcopter payload add and drop applications," in *34th Chinese Control Conference (CCC)*. IEEE, 2015, pp. 3252–3257.

[24] M. E. Guerrero-Sánchez, D. A. Mercado-Ravell, R. Lozano, and C. D. García-Beltrán, "Swing-attenuation for a quadrotor transporting a cable-suspended payload," *ISA Transactions*, vol. 68, pp. 433–449, 2017.

[25] K. Klausen, T. I. Fossen, and T. A. Johansen, "Nonlinear Control with Swing Damping of a Multirotor UAV with Suspended Load," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 88, no. 2-4, pp. 379–394, 2017.

[26] S. Ichikawa, A. Castro, N. Johnson, H. Kojima, and W. Singhose, "Dynamics and command shaping control of quadcopters carrying suspended loads," *IFAC-PapersOnLine*, vol. 51, no. 14, pp. 84–88, 2018.

[27] E. L. de Angelis, "Swing angle estimation for multicopter slung load applications," *Aerospace Science and Technology*, vol. 89, pp. 264–274, 2019.

[28] P. Kotaru, G. Wu, and K. Sreenath, "Dynamics and control of a quadrotor with a payload suspended through an elastic cable," in *American Control Conference (ACC)*. IEEE, 2017, pp. 3906–3913.

[29] S. Tang and V. Kumar, "Mixed Integer Quadratic Program trajectory generation for a quadrotor with a cable-suspended payload," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2216–2222.

[30] F. A. Goodarzi and T. Lee, "Dynamics and control of quadrotor UAVs transporting a rigid body connected via flexible cables," in *American Control Conference (ACC)*. IEEE, 2015, pp. 4677–4682.

[31] F. A. Goodarzi, D. Lee, and T. Lee, "Geometric stabilization of a quadrotor UAV with a payload connected by flexible cable," in *American Control Conference (ACC)*. IEEE, 2014, pp. 4925–4930.

[32] P. Kotaru, G. Wu, and K. Sreenath, "Differential-flatness and control of quadrotor(s) with a payload suspended through flexible cable(s)," in *Indian Control Conference (ICC)*. IEEE, 2018, pp. 352–357.

[33] T. Lee, "Collision avoidance for quadrotor UAVs transporting a payload via Voronoi tessellation," in *American Control Conference (ACC)*. IEEE, 2015, pp. 1842–1848.

[34] D. Sanalitro, H. J. Savino, M. Tognon, J. Cortes, and A. Franchi, "Full-Pose Manipulation Control of a Cable-Suspended Load With Multiple UAVs Under Uncertainties," *Robotics and Automation Letters*, vol. 5, no. 2, pp. 2185–2191, 2020.

[35] K. Klausen, T. I. Fossen, and T. A. Johansen, "Suspended load motion control using multicopters," in *22nd Mediterranean Conference on Control and Automation, MED 2014*. IEEE, 2014, pp. 1371–1376.

[36] M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2958–2964.

[37] J. Zeng, P. Kotaru, and K. Sreenath, "Geometric Control and Differential Flatness of a Quadrotor UAV with Load Suspended from a Pulley," in *American Control Conference (ACC)*. IEEE, 2019, pp. 2420–2427.

[38] B. Xian, S. Wang, and S. Yang, "An Online Trajectory Planning Approach for a Quadrotor UAV With a Slung Payload," *Transactions on Industrial Electronics*, vol. 67, no. 8, pp. 6669–6678, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8825990/

[39] G. Starr, J. Wood, and R. Lumia, "Rapid Transport of Suspended Payloads," in *International Conference on Robotics and Automation*. IEEE, pp. 1394–1399.

[40] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach," in *International Conference on Robotics and Automation*. IEEE, 2012, pp. 2691–2697.

[41] J. Su, J.-H. Bak, and J. Hyun, "Optimal Trajectory Generation for Quadrotor with Suspended Load Under Swing Angle Constraint," in *15th International Conference on Control and Automation (ICCA)*. IEEE, 2019, pp. 549–554.

[42] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Fierro, "A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots," in *International Conference on Robotics and Automation*. IEEE, 2013, pp. 4896–4901.

[43] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Learning swing-free trajectories for UAVs with a suspended load," in *International Conference on Robotics and Automation*. IEEE, 2013, pp. 4902–4909.

[44] J. Vaughan, A. Yano, and W. Singhose, "Comparison of robust input shapers," *Journal of Sound and Vibration*, vol. 315, no. 4-5, pp. 797–815, 2008.

[45] O. M. Smith, "Posicast Control of Damped Oscillatory Systems," in *Proceedings of the IRE*, vol. 45, no. 9, 1957, pp. 1249–1255.

[46] G. Starr, "Swing-free transport of suspended objects with a robot manipulator," in *The 22nd Conference on Decision and Control*. IEEE, 1983, pp. 1484–1487.

[47] M. Bisgaard, A. La Cour-Harbo, and J. D. Bendtsen, "Input shaping for helicopter slung load swing reduction," in *AIAA Guidance, Navigation and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics Inc., 2008.

[48] J. Potter, W. Singhose, and M. Costelloy, "Reducing swing of model helicopter sling load using input shaping," in *9th International Conference on Control and Automation (ICCA)*. IEEE, 2011, pp. 348–353.

[49] P. Homolka, M. Hromcik, and T. Vyhlidal, "Input shaping solutions for drones with suspended load: First results," in *21st International Conference on Process Control (PC)*. IEEE, 2017, pp. 30–35.

[50] S. Sadr, S. A. A. Moosavian, and P. Zarafshan, "Damping control of a quadrotor with swinging load using input shaping method," in *Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. IEEE, 2014, pp. 227–232.

[51] S. Fielding and M. Nahon, "Input Shaped Trajectory Generation and Controller Design for a Quadrotor-Slung Load System," in *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 162–170.

[52] J. Trachte, F. Gonzalez, and A. McFadyen, "Nonlinear Model Predictive Control for a multi-rotor with heavy slung load," in *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 1105–1110.

[53] Y. Alothman and D. Gu, "Using Constrained NMPC to Control a Cable-Suspended Payload With Two Quadrotors," in *24th International Conference on Automation and Computing (ICAC)*. IEEE, 2018, pp. 1–6.

[54] Y. Alothman and D. Gu, "Incentive stackelberg dynamic game approach to transporting a cable-suspended load with two quadrotors," in *10th Computer Science and Electronic Engineering (CEEC)*. IEEE, 2018, pp. 270–275.

[55] Y. Alothman and D. Gu, "Quadrotor transporting cable-suspended load using iterative Linear Quadratic regulator (iLQR) optimal control," in *8th Computer Science and Electronic Engineering (CEEC)*. IEEE, 2016, pp. 168–173.

[56] S. Notter, A. Heckmann, A. Mcfadyen, and F. Gonzalez, "Modelling, Simulation and Flight Test of a Model Predictive Controlled Multirotor with Heavy Slung Load," *IFAC-PapersOnLine*, vol. 49, no. 17, pp. 182–187, 2016.

[57] A. Erasmus and H. Jordaan, "Linear Quadratic Gaussian Control of a Quadrotor with an Unknown Suspended Payload," in *International SAUPEC/RobMech/PRASA Conference*. IEEE, 2020, pp. 1–6.

[58] M. A. Santos, A. Ferramosca, and G. V. Raffo, "Tube-based MPC with Nonlinear Control for Load Transportation using a UAV," *IFAC-PapersOnLine*, vol. 51, no. 25, pp. 459–465, 2018.

[59] X. Liang, H. Lin, P. Zhang, S. Wu, N. Sun, and Y. Fang, "A Nonlinear Control Approach for Aerial Transportation Systems With Improved Antiswing and Positioning Performance," *Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 2104–2114, 2021.

[60] D. Allahverdy, A. Fakharian, and M. B. Menhaj, "Back-Stepping Integral Sliding Mode Control with Iterative Learning Control Algorithm for Quadrotor UAV Transporting Cable-Suspended Payload," in *29th Iranian Conference on Electrical Engineering (ICEE)*. IEEE, 2021, pp. 660–665.

[61] A. Faust, P. Ruymgaart, M. Salman, R. Fierro, and L. Tapia, "Continuous action reinforcement learning for control-affine systems with unknown dynamics," *IEEE/CAA Journal of Automatica Sinica*, vol. 1, no. 3, pp. 323–336, 2014.

[62] H. Hua, Y. Fang, X. Zhang, and C. Qian, "A New Nonlinear Control Strategy Embedded With Reinforcement Learning for a Multirotor Transporting a Suspended Payload," *ASME Transactions on Mechatronics*, pp. 1–1, 2021.

[63] C. C. Taylor, J. A. A. Engelbrecht, and M. J. Treurnicht, "Robust Control of a Quadrotor with a Suspended Payload," Ph.D. dissertation, Stellenbosch University, 2020.

[64] S. Dai, T. Lee, and D. S. Bernstein, "Adaptive control of a quadrotor UAV transporting a cable-suspended load with unknown mass," in *53rd Conference on Decision and Control*. IEEE, 2014, pp. 6149–6154.

[65] M. A. Santos and G. V. Raffo, "Path tracking Model Predictive Control of a Tilt-rotor UAV carrying a suspended load," in *19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1458–1463.

[66] R. Andrade, G. V. Raffo, and J. E. Normey-Rico, "Model predictive control of a tilt-rotor UAV for load transportation," in *European Control Conference (ECC)*. IEEE, 2016, pp. 2165–2170.

[67] M. Zurn, K. Morton, A. Heckmann, A. McFadyen, S. Notter, and F. Gonzalez, "MPC controlled multirotor with suspended slung Load: System architecture and visual load detection," in *Aerospace Conference*. IEEE, 2016, pp. 1–11.

[68] C. Y. Son, D. Jang, H. Seo, T. Kim, H. Lee, and H. J. Kim, "Real-time Optimal Planning and Model Predictive Control of a Multi-rotor with a Suspended Load," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5665–5671.

[69] C. Y. Son, H. Seo, T. Kim, and H. Jin Kim, "Model Predictive Control of a Multi-Rotor with a Suspended Load for Avoiding Obstacles," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–6.

[70] C. Y. Son, T. Kim, S. Kim, and H. J. Kim, "Model predictive control of a multi-rotor with a slung load for avoiding obstacles," in *14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2017, pp. 232–237.

[71] J. E. Trachte, L. F. Gonzalez Toro, and A. McFadyen, "Multi-rotor with suspended load: System Dynamics and Control Toolbox," in *Aerospace Conference*. IEEE, 2015, pp. 1–9.

[72] J. Zeng and K. Sreenath, "Geometric Control of a Quadrotor with a Load Suspended from an Offset," in *American Control Conference (ACC)*. IEEE, 2019, pp. 3044–3050.

[73] S. Yang and B. Xian, "Robust Control Design for the Quadrotor UAV with a Suspended Payload," in *8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2018, pp. 469–473.

[74] A. Martinez-Vasquez, R. Castro-Linares, and A. Rodriguez-Mata, "Sliding Mode Control of a Quadrotor with Suspended Payload: a Differential Flatness Approach," in *17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. IEEE, 2020, pp. 1–6.

[75] A. Mosco-Luciano, R. Castro-Linares, and H. Rodriguez-Cortes, "Trajectory Tracking Control for a Quadrotor with a Slung Load," in *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 322–328.

[76] G. Rigatos, K. Busawon, P. Wira, and M. Abbaszadeh, "Nonlinear Optimal Control of the UAV and Suspended Payload System," in *IECON - 44th Annual Conference of the Industrial Electronics Society*. IEEE, 2018, pp. 3776–3781.

[77] Y. Alothman, W. Jasim, and D. Gu, "Quad-rotor lifting-transporting cable-suspended payloads control," in *21st International Conference on Automation and Computing (ICAC)*. IEEE, 2015, pp. 1–6.

[78] J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, pp. 415–424, 2011.

[79] V. P. Tran, M. A. Mabrok, M. A. Garratt, and I. R. Petersen, "Hybrid adaptive negative imaginary- neural-fuzzy control with model identification for a quadrotor," *IFAC Journal of Systems and Control*, vol. 16, p. 100156, 2021.

[80] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear Quadrocopter Attitude Control," Eidgenössische Technische Hochschule Zürich, Departement Maschinenbau und Verfahrenstechnik, Tech. Rep., 2013.

[81] H. Schaub, "Kinematics: Describing the Motions of Spacecraft - Coursera," 2017. [Online]. Available: https://www.coursera.org/learn/spacecraft-dynamics-kinematics

[82] P. D. S. Moller, "Automated landing of a quadrotor Unmanned Aerial Vehicle on a translating platform," Ph.D. dissertation, Stellenbosch University, 2015.

[83] The MathWorks Inc., "Symbolic Math Toolbox," Natick, Massachusetts, USA, 2021. [Online]. Available: https://www.mathworks.com/help/symbolic/

[84] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," in *Proceedings of the National Academy of Sciences*, vol. 113, no. 15. National Academy of Sciences, 2016, pp. 3932–3937.

[85] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *Journal of Computational Dynamics*, vol. 1, no. 2, pp. 391–421, 2014.

[86] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[87] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[88] H. Arbabi, M. Korda, and I. Mezic, "A Data-Driven Koopman Model Predictive Control Framework for Nonlinear Partial Differential Equations," in *Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6409–6414.

[89] C. M. Chen and K. H. Wang, "State-space model conversion of a system with state delay," in *Proceedings of the National Science Council, Republic of China, Part A: Physical Science and Engineering*, vol. 23, no. 6, 1999, pp. 782–788.

[90] B. R. Noack, W. Stankiewicz, M. Morzyński, and P. J. Schmid, "Recursive dynamic mode decomposition of transient and post-transient wake flows," *Journal of Fluid Mechanics*, vol. 809, pp. 843–872, 2016.

[91] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. Nathan Kutz, "Chaos as an intermittently forced linear system," *Nature Communications*, vol. 8, no. 1, 2017.

[92] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," in *Proceedings of the National Academy of Sciences*, vol. 116, no. 45. National Academy of Sciences, 2019, pp. 22 445–22 451.

[93] M. Kamb, E. Kaiser, S. L. Brunton, and J. N. Kutz, "Time-Delay Observables for Koopman: Theory and Applications," *SIAM Journal on Applied Dynamical Systems*, vol. 19, no. 2, pp. 886–917, 2020.

[94] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *International Conference on Robotics and Automation*. IEEE, 2015, pp. 6235–6240.

[95] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149–2154, 2004.

[96] J. Zhao, Y. Zhu, and R. Patwardhan, "Identification of k-step-ahead prediction error model and MPC control," *Journal of Process Control*, vol. 24, no. 1, pp. 48–56, 2014.

[97] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[98] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," in *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2219, 2018, p. 20180335.

[99] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, "Model selection for dynamical systems via sparse regression and information criteria," in *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2204. Royal Society Publishing, 2017, p. 20170009.

[100] S. L. Brunton and J. N. Kutz, "Regression and Model Selection," in *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, vol. 1, pp. 117–153.

[101] S. L. Brunton and J. N. Kutz, "Balanced Models for Control," in *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019, pp. 321–344.

[102] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep Double Descent: Where Bigger Models and More Data Hurt," in *International Conference on Learning Representations*, 2020.

[103] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.

[104] K. Y. Us, A. Cevher, M. Sever, and A. Kirli, "On the Effect of Slung Load on Quadrotor Performance," *Procedia Computer Science*, vol. 158, pp. 346–354, 2019.

[105] L. Li, L. Sun, and J. Jin, "Survey of advances in control algorithms of quadrotor unmanned aerial vehicle," in *International Conference on Communication Technology Proceedings, ICCT*, 2016, pp. 107–111.

[106] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[107] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice - A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[108] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model Predictive Control for Micro Aerial Vehicles: A Survey," 2020.

[109] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[110] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit - An open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[111] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proceedings of the International Symposium on Computer-Aided Control System Design*, 2004, pp. 284–289.

[112] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," *European Control Conference, ECC 2013*, pp. 502–510, 2013.

[113] S. Lucia, A. Tătulea-Codrean, C. Schoppmeyer, and S. Engell, "Rapid development of modular and sustainable nonlinear model predictive control solutions," *Control Engineering Practice*, vol. 60, no. 60, pp. 51–62, 2017.

[114] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System," in *Robot Operating System (ROS) The Complete Reference, Volume 2*, A. Koubaa, Ed. Springer, 2017.

[115] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-Aware Model Predictive Control for Quadrotors," in *International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 5200–5207.

[116] The MathWorks Inc., "Model Predictive Control Toolbox," Natick, Massachusetts, USA, 2019. [Online]. Available: https://www.mathworks.com/help/mpc/ug/optimization-problem.html

[117] C. Schmid and L. Biegler, "Quadratic programming methods for reduced hessian SQP," *Computers & Chemical Engineering*, vol. 18, no. 9, pp. 817–832, 1994.

[118] D. D. Ruscio, "Model Predictive Control with Integral Action: A simple MPC algorithm," *Identification and Control*, vol. 34, no. 3, pp. 119–129, 2013.

[119] J. Sawma, F. Khatounian, E. Monmasson, R. Ghosn, and L. Idkhajine, "The effect of prediction horizons in MPC for first order linear systems," in *International Conference on Industrial Technology*. IEEE, 2018, pp. 316–321.

[120] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Open-Source Software Workshop*, 2009.

Appendix A

PID controller gains

This appendix documents the PID gains used for Honeybee.
The angular rate controller gains are given in Table A.1.

Controller	P gain	I gain	D gain
Roll rate	0.150	0.200	0.003
Pitch rate	0.150	0.200	0.003
Yaw rate	0.200	0.100	0.000

Table A.1: The angular rate controller gains.

The angle controller gains are given in Table A.2.

Controller	P gain
Roll angle	6.5
Pitch angle	6.5
Yaw angle	2.8

Table A.2: The angle controller gains.

The velocity controller gains are given in Table A.3.

Controller	P gain	I gain	D gain
North velocity	1.8	0.4	0.2
East velocity	1.8	0.4	0.2
Down velocity	4.0	2.0	0.0

Table A.3: The velocity controller gains.

The position controller gains are given in Table A.4.

Controller	P gain
North position	0.95
East position	0.95
Down position	1.0

Table A.4: The position controller gains.