# 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

# Contents

# List of figures

# List of tables

# Symbols

| | |
|---|---|
| $x$ | Do I include all symbols ??. |

# Abbreviations

| | |
|---|---|
| 2D | Two-Dimensional |
| ADC | Analog to Digital Converter |
| AIC | Akaike's Information Criteria |
| BIC | Bayesian Information Criteria |
| CoM | Centre of Mass |
| DMD | Dynamic Mode Decomposition |
| DMDc | Dynamic Mode Decomposition with Control |
| EKF | Extended Kalman Filter |
| FFT | Fast Fourier Transform |
| GPS | Global Positioning System |
| HAVOK | Hankel Alternative View Of Koopman |
| HAVOKc | Hankel Alternative View Of Koopman with Control |
| IMU | Inertial Measurement Unit |
| LQR | Linear Quadratic Regulator |
| MAPE | Mean Absolute Percentage Error |
| MASE | Mean Absolute Scaled Error |
| MPC | Model Predictive Control |
| MRAE | Mean Relative Absolute Error |
| MSE | Mean Squared Error |
| NMAE | Normalised Mean Absolute Error |
| PE | Percentage Error |
| PID | Proportional Integral Derivative |
| POD | Proper Orthogonal Decomposition |
| RLS | Recursive Least Squares |
| ROS | Robot Operating System |
| SITL | Software-in-the-Loop |
| SVD | Singular Value Decomposition |

# Chapter 1

# Introduction

## 1.1. Background

## 1.2. Project definition and objectives

## 1.3. Thesis outline

# Chapter 2

# Literature study

## 2.1. Unknown suspended payloads

Figure 2.1 shows an example of a practical use case of a multirotor with a suspended payload. Items are placed in a water proof bag and transported by the multirotor to a place of need. Note that the multirotor does not know the dynamics of the payload before flight, because the items may change depending on the need in the specific situation. The shape and mass of the loaded items will have an effect on swing of the payload, and the control system should be able to fly well despite the altered flight dynamics.

**Figure 2.1:** A practical suspended payload used for search and rescue missions [1]

## 2.2. System Identification

## 2.3. Control systems

## 2.4. System Design

Blok diagram komponente

# Chapter 3

# Modelling

This chapter discusses the mathematical modelling of a multirotor with a suspended payload which is based on a practical multirotor UAV named Honeybee. The model is first derived as a Two-Dimensional (2D) model. The system identification and control system techniques in later chapters will then be explained based on the 2D model to avoid unnecessary complexity. Finally, it will be described how this model and the techniques in later chapters are extended to the 3D case. This 3D mathematical model will be used in a nonlinear simulation of a multirotor and suspended payload. 1

## 3.1. Coordinate frames

## 3.2. States

## 3.3. Forces and moments

## 3.4. Lagrangian mechanics

1. Discuss weak link between altitude dynamics and swing angle

## 3.5. Linearised model

## 3.6. Discretised model

## 3.7. Model verification

## 3.8. Dynamic payloads

In this work, a dynamic payload refers to a payload with dynamics that differ significantly from a rigid mass. When considering suspended payloads, a dynamic payload induces dynamics which differ from that of a simple pendulum.

- Most control in literature model payload as rigid Mass. Give references.
- Some add spring stiffness of cable (give references e.g. QuadLoad ElasticCable Prasanth Kotaru)
- but not practical because can design which cable you used. Cannot design which payload needs to be transported
- water looks like double payload

# 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 [2] and [3] for system identification for swing damping control of a multirotor with an unknown suspended payload. The system was modelled as two rigid bodies connected by a link and the following assumptions were made regarding the suspended payload:

- The payload is a point mass.
- The link is massless.
- The link is rigid.
- The link is attached to the Centre of Mass (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 Linear Quadratic Regulator (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 [2] and [3], 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 [2] and [3] 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 identifcation 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. Add more advantages ??

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 expensive and may not be practically realisable on the available hardware on a multirotor.

DMDc and 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 [4] 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 Model Predictive Control (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

Only the North velocity controller will be considered in this section. Because of the symmetry of the multirotor, this controller can then duplicated for East velocity control. The resulting input vector of the system identification plant is given by,

$$\boldsymbol{u} = \left[ A_{N,sp} \right] . \tag{4.1}$$

For swing damping control, the controllers require state feedback from both the multirotor and payload, hence the state vector of the considered plant is,

$$\boldsymbol{x} = \left[ V_N \quad \theta \quad \dot{\theta} \right]^T . \tag{4.2}$$

Note that position is not included in the state vector because it does not affect the velocity controller. Also note that the inner loop controllers handle the attitude dynamics of the multirotor. During simulations of pure longitudinal velocity setpoints, it was observed that the multirotor experiences negligible altitude changes due the swinging payload. The plant seen by the system identification process therefore mimics the common pendulum-on-a-cart model. A schematic of this 2D plant is shown in Figure 4.1. In the following sections, simulations of the full multirotor and payload system will be performed and different methods will be applied to identify different models of this plant.

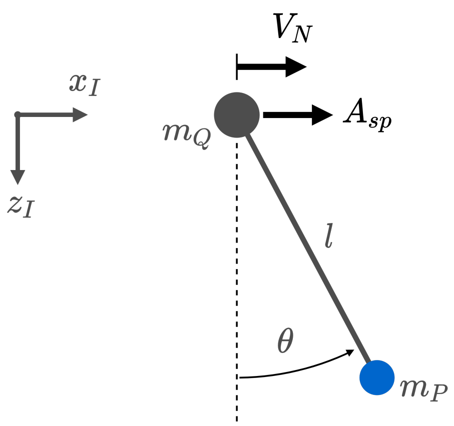



**Figure 4.1:** Schematic of a floating pendulum model considered for a North velocity controller

## 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 a linear, continuous-time, state-space form. This model was derived and linearised a priori in Section 3.5. 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 method was used by [2] and [3] to design a LQR controller and implement swing damping control of a multirotor with an unknown suspended payload.

### 4.3.1. Payload mass estimation

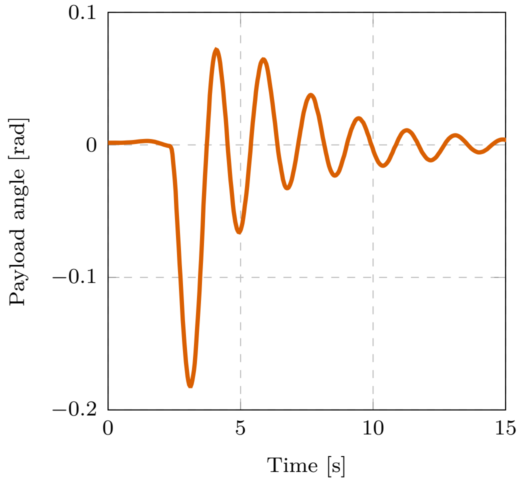
Recursive Least Squares (RLS) is used by [2] and [3] to estimate the payload mass. It is assumed that the multirotor mass is known before a flight, therefore the payload mass can be estimated from the additional trust required during hover. In both [2] and [3] it is demonstrated that RLS is very accurate 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.

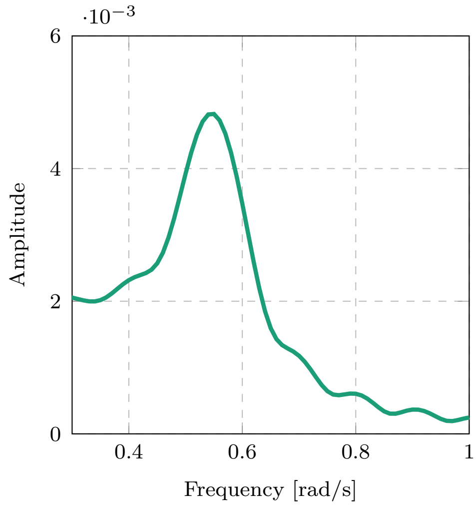
### 4.3.2. Cable length estimation

The cable length is estimated from the measurement of the natural frequency of the swinging payload. As described by [5], the natural frequency is given by:

$$\omega_n = \sqrt{\frac{g}{l} \cdot \frac{m_q + m_p}{m_q}}. \tag{4.3}$$




**(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 ($l = 1\,\text{m}$, $m_p = 0.3\,\text{kg}$.).

The cable length can clearly be calculated from Equation 4.3 if the other parameters are known. The natural frequency is measured by performing a Fast Fourier Transform (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, 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 $\omega_n$ has been measured, $l$ can be determined from Equation 4.3. 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 [2] and [3], an error of this magnitude is acceptably small and still results in effective control with a LQR. It was also shown by [2] and [3] 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 [6]. 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 [7]. 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 observables. Enriching a DMD model with time-delay-embedding is a known technique and is also seen in other DMD adaptations [8, 9].

DMD produces a linear, discrete state-space model of system dynamics. Discrete measurements, $\boldsymbol{x}_k$, of the continuous time observable, $\boldsymbol{x}(t)$, 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 [10]. Hence, we define a state delay vector as:

$$\boldsymbol{d}_k = \begin{bmatrix} \boldsymbol{x}_{k-1} & \boldsymbol{x}_{k-2} & \cdots & \boldsymbol{x}_{k-q+1} \end{bmatrix}^T, \tag{4.4}$$

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:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{A}_d\boldsymbol{d}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{4.5}$$

where $\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$ is the system matrix, $\boldsymbol{A}_1 \in \mathbb{R}^{(q-1)\cdot n_x \ \times \ (q-1)\cdot n_x}$ is the state delay system matrix and $\boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}$ 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 Proportional Integral Derivative (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 [11]. However this implementation is not considered because a companion computer with significant memory size can be used.

The training data is collected into the following matrices:

$$
\begin{aligned}
\boldsymbol{X'} &= \begin{bmatrix} \boldsymbol{x}_{q+1} & \boldsymbol{x}_{q+2} & \boldsymbol{x}_{q+3} & \cdots & \boldsymbol{x}_{w+q} \end{bmatrix}, \\
\boldsymbol{X} &= \begin{bmatrix} \boldsymbol{x}_{q} & \boldsymbol{x}_{q+1} & \boldsymbol{x}_{q+2} & \cdots & \boldsymbol{x}_{w+q-1} \end{bmatrix}, \\
\boldsymbol{X}_d &= \begin{bmatrix} \boldsymbol{x}_{q-1} & \boldsymbol{x}_{q+0} & \boldsymbol{x}_{q+1} & \cdots & \boldsymbol{x}_{w+q-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_{2} & \boldsymbol{x}_{3} & \boldsymbol{x}_{4} & \cdots & \boldsymbol{x}_{w+1} \\ \boldsymbol{x}_{1} & \boldsymbol{x}_{2} & \boldsymbol{x}_{3} & \cdots & \boldsymbol{x}_{w} \end{bmatrix}, \\
\boldsymbol{\Upsilon} &= \begin{bmatrix} \boldsymbol{u}_{q} & \boldsymbol{u}_{q+1} & \boldsymbol{u}_{q+2} & \cdots & \boldsymbol{u}_{w+q-1} \end{bmatrix},
\end{aligned}
\tag{4.6}
$$

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 4.5 can be combined with the matrices in Equation 4.6 to produce:

$$
\boldsymbol{X'} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{A}_d\boldsymbol{X}_d + \boldsymbol{B}\boldsymbol{\Upsilon}. \tag{4.7}
$$

Note that the primary objective of DMDc is to determine the best fit model matrices, $\boldsymbol{A}$, $\boldsymbol{A}_d$ and $\boldsymbol{B}$, given the data in $\boldsymbol{X'}$, $\boldsymbol{X}$, $\boldsymbol{X}_d$, and $\boldsymbol{\Upsilon}$ [7]. In order to group the unknowns into a single matrix, Equation 4.5 is manipulated into the form,

$$
\boldsymbol{X'} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{A}_d & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}_d \\ \boldsymbol{\Upsilon} \end{bmatrix} = \boldsymbol{G}\boldsymbol{\Omega}, \tag{4.8}
$$

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: $\boldsymbol{\Omega} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. Often, only the first $p$ columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are required for a good approximation of the dynamics [12]. 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:

$$
\boldsymbol{\Omega} \approx \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^T, \tag{4.9}
$$

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 4.9 with the over-constrained equality in Equation 4.8, the least-squared solution, $\boldsymbol{G}$, can be found with:

$$\boldsymbol{G} \approx \boldsymbol{X'}\tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{U}}. \tag{4.10}$$

By reversing Equation 4.8, $\boldsymbol{G}$ can now be separated into: $\boldsymbol{G} = \begin{bmatrix}\boldsymbol{A} & \boldsymbol{A}_d & \boldsymbol{B}\end{bmatrix}$. 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 [12, 13]. The Koopman operator is a method of representing finite-dimensional nonlinear dynamics in terms of an infinite-dimensional linear operator [12]. 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 [12].

The original formulation of HAVOK is only defined for uncontrolled dynamical systems [12]. In this work, we have adapted the standard HAVOK algorithm to account for the effect of control. This implementation well be refered to as HAVOKc HAVOKc 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:

$$\boldsymbol{a}_k = \begin{bmatrix}\boldsymbol{x}_{k-(q-1)} & \cdots & \boldsymbol{x}_{k-1} & \boldsymbol{x}_k\end{bmatrix}^T, \tag{4.11}$$

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,

$$\boldsymbol{a}_{k+1} = \boldsymbol{A}_H\boldsymbol{a}_k + \boldsymbol{B}_H\boldsymbol{u}_k, \tag{4.12}$$

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. Here, the subscript of $\boldsymbol{A}_H$ and $\boldsymbol{B}_H$ is used to differentiate these matrices from $\boldsymbol{A}$ and $\boldsymbol{B}$

used in DMDc.

The original HAVOK algorithm, developed by [14], 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:

$$\mathbf{\Pi} = \begin{bmatrix} \boldsymbol{a}_q & \boldsymbol{a}_{q+1} & \boldsymbol{a}_{q+2} & \cdots & \boldsymbol{a}_{w+q-1} \\ \boldsymbol{u}_q & \boldsymbol{u}_{q+1} & \boldsymbol{u}_{q+2} & \cdots & \boldsymbol{u}_{w+q-1} \end{bmatrix}, \tag{4.13}$$

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

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

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 [15]. 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 [14]. Consider the following discrete, state-space formulation:

$$\boldsymbol{v}_{k+1} = \mathbf{\Lambda}\boldsymbol{v}_k. \tag{4.15}$$

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 4.15, $\tilde{\boldsymbol{V}}^T$ is divided into two matrices:

$$\begin{aligned} \boldsymbol{V}_1 &= \begin{bmatrix} \boldsymbol{v}_q & \boldsymbol{v}_{q+1} & \dots & \boldsymbol{v}_{w-1} \end{bmatrix}, \\ \boldsymbol{V}_2 &= \begin{bmatrix} \boldsymbol{v}_{q+1} & \boldsymbol{v}_{q+2} & \dots & \boldsymbol{v}_w \end{bmatrix}, \end{aligned} \tag{4.16}$$

where $\boldsymbol{V}_2$ is $\boldsymbol{V}_1$ advanced a single step forward in time. The matrices from Equation 4.16 are now combined with Equation 4.15 and the best fit $\mathbf{\Lambda}$ is determined with the Moore-Penrose pseudoinverse:

$$\boldsymbol{V}_2 = \mathbf{\Lambda}\boldsymbol{V}_1 \qquad \Rightarrow \qquad \mathbf{\Lambda} \approx \boldsymbol{V}_1\boldsymbol{V}_1^\dagger \tag{4.17}$$

It can be shown from Equation 4.14 that Equation 4.15 is transformed from the eigen-time-delay coordinate system to the original coordinate system as the following:

$$\begin{bmatrix} \boldsymbol{a}_{k+1} \\ \boldsymbol{u}_{k+1} \end{bmatrix} = (\tilde{\boldsymbol{U}}\tilde{\mathbf{\Sigma}})\mathbf{\Lambda}(\tilde{\boldsymbol{U}}\tilde{\mathbf{\Sigma}})^\dagger \begin{bmatrix} \boldsymbol{a}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{4.18}$$

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

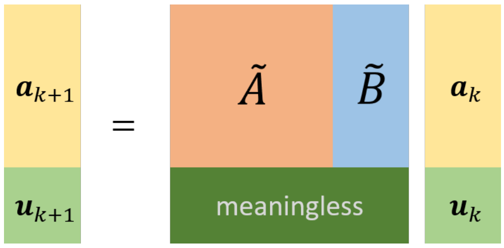



**Figure 4.3:** Illustration of the extraction of $\boldsymbol{A}_H$ and $\boldsymbol{B}_H$ from Equation 4.18

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 4.12 to render the HAVOKc model.

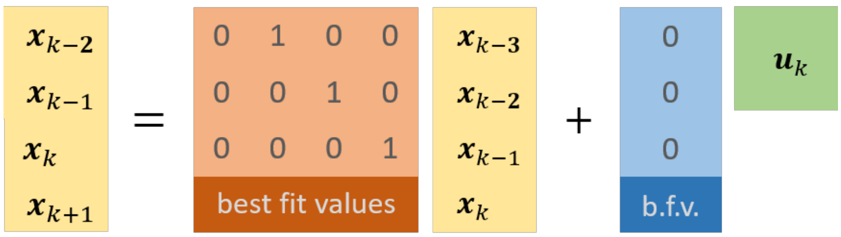



**Figure 4.4:** Illustration of forcing the known values in HAVOKc matrices

## 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 [16] using the Gazebo simulator [17] is used to generate data for system identification. Testing these techniques with simulation data 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 Extended Kalman Filter (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 |

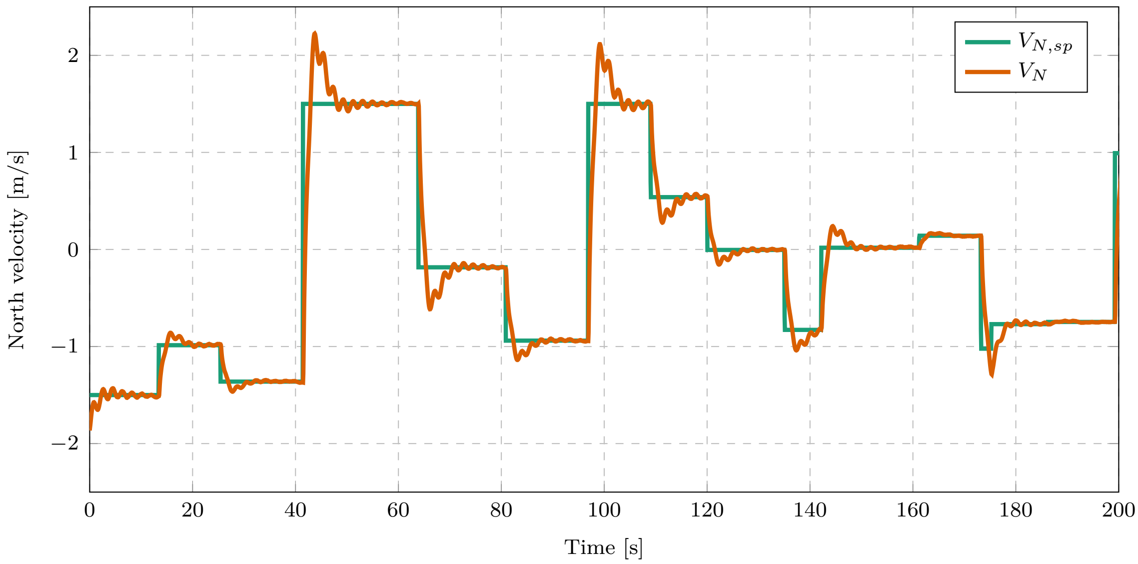



**Figure 4.5:** Example of training data with random velocity step inputs ($m_p = 0.2\,\mathrm{kg}$, $l = 1\,\mathrm{m}$)

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 Global Positioning System (GPS) noise rather than the swinging payload, since they do not show high freqeuncy osccilations 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 a MPC based on k-step-ahead prediction errors [18]. 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 [18]. 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:

$$\boldsymbol{MAE} = mean\left(\ |\hat{\boldsymbol{x}}_{\boldsymbol{k}} - \boldsymbol{x}_{\boldsymbol{k}}|\ \right), \tag{4.19}$$

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 [19]. 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 becasue 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 a MPC [20], 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 [19]. 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 for 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:

$$NMAE = \frac{MAE}{x_{max} - x_{min}} \tag{4.20}$$

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 [21]. 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 helps to avoid overfitting, but also ensures that the MPC optimisation problem is not too computational expensive 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 a 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,

$$5 < q < 30, \qquad 5 < p < 50 \tag{4.21}$$

.

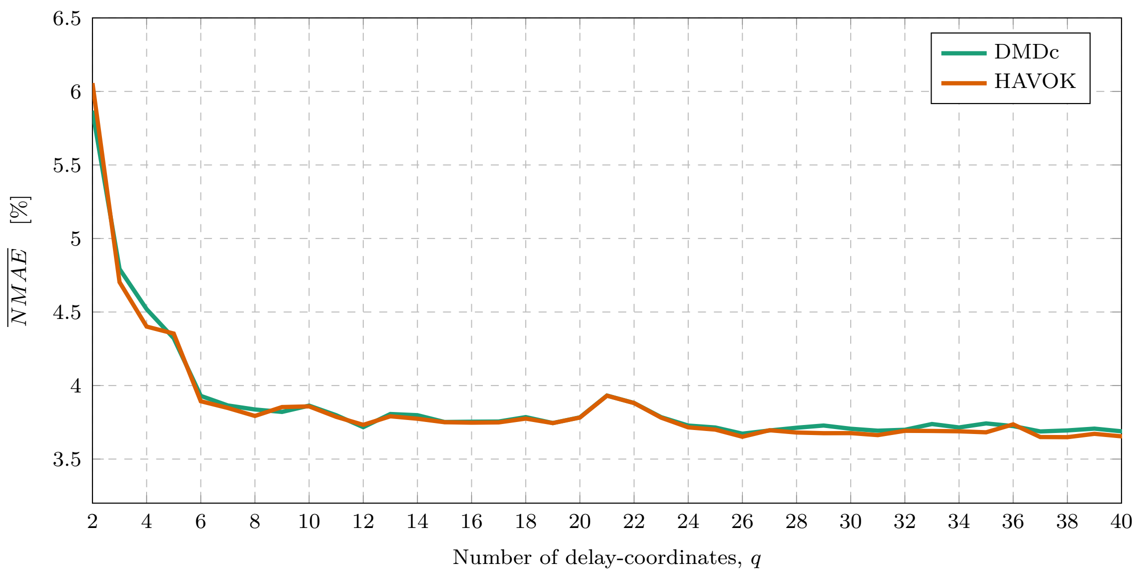



**Figure 4.6:** DMDc and HAVOKc predictions error for different lengths of noisy training data ($m_p = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$, $T_{train} = 60\,\text{s}$.)

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 [21]. 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 [22]. 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 the corresponding Proper Orthogonal Decomposition (POD) mode to the plant dynamics [23]. 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.

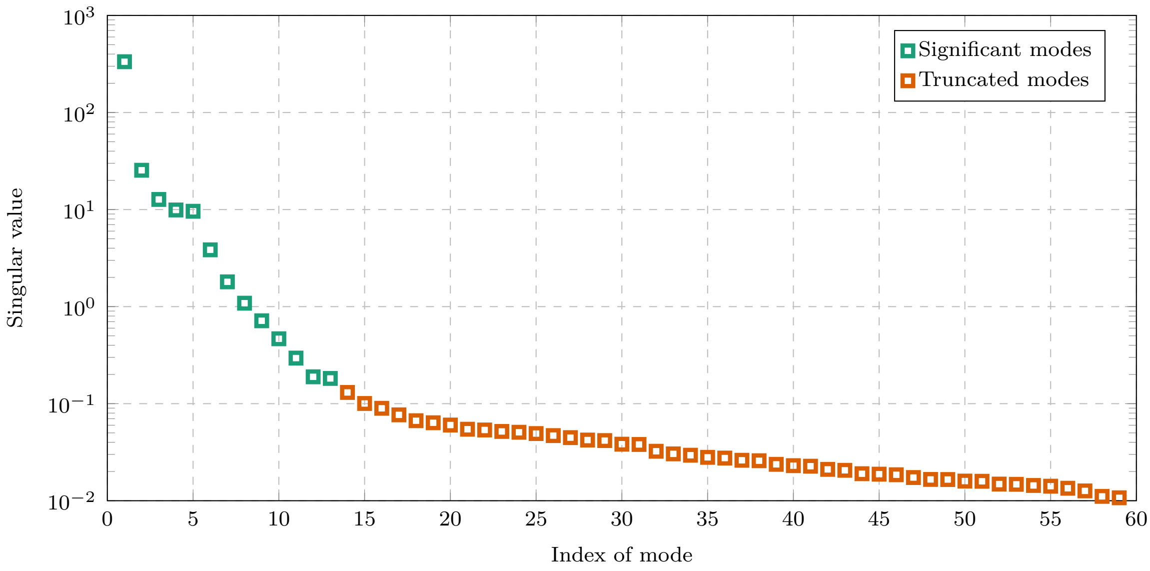



**Figure 4.7:** Significant and truncated singular values of a HAVOKc model produced from noisy data ($m_p = 0.2\,\mathrm{kg}$, $l = 0.5\,\mathrm{m}$, $T_s = 0.03\,\mathrm{s}$, $T_{train} = 60\,\mathrm{s}$.)

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 [23].

### 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 $T_s = 0.02\,\mathrm{s}$. Due to the computational complexity of an MPC, the optimiser will struggle to run at 50 Hz on a companion computer 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.

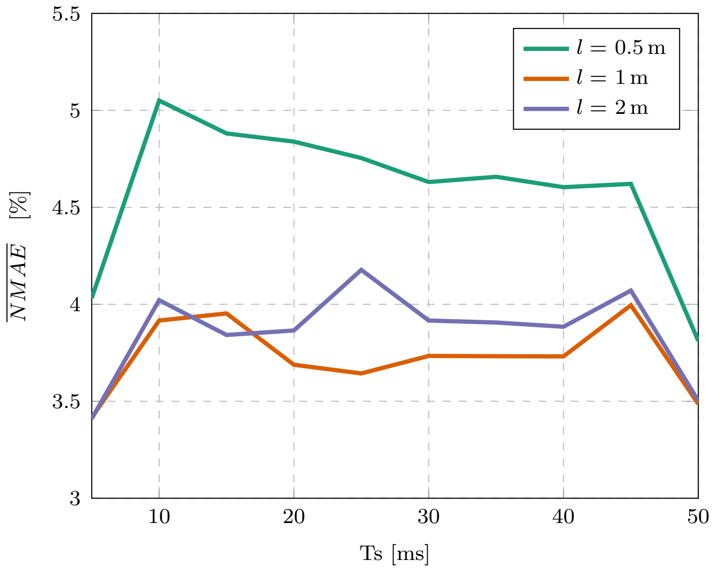



**Figure 4.8:** DMDc prediction error using different cable lengths with a range of different sample times of noisy training data ($m_p = 0.2\,\mathrm{kg}$)

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 oscilations 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 $\theta$. 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 $\theta$. An example of this is the backward Euler form,

$$\dot{\theta}_k = (\frac{1}{T_s}) \cdot \theta_k - (\frac{1}{T_s}) \cdot \theta_{k-1}. \tag{4.22}$$

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 $\theta$. This is because $\dot{\theta}$ is directly related to the dynamics, compared to $\theta$ 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.

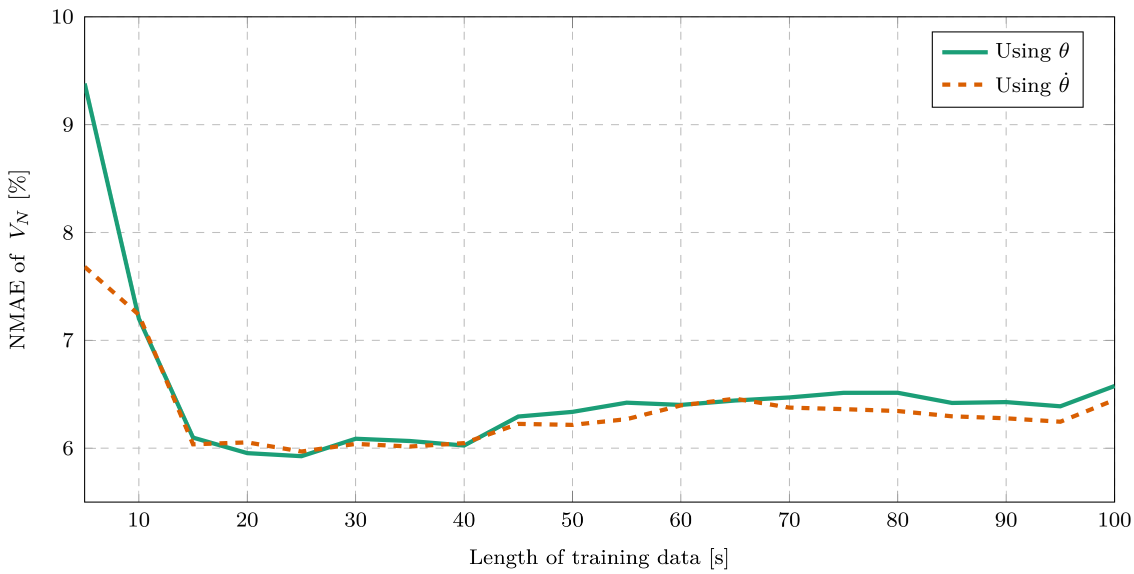



**Figure 4.9:** Prediction NMAE for HAVOKc models using either angle or angular rate measurements ($m_p = 0.2\,\mathrm{kg}$, $l = 1\,\mathrm{m}$, $T_s = 0.03\,\mathrm{s}$).

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

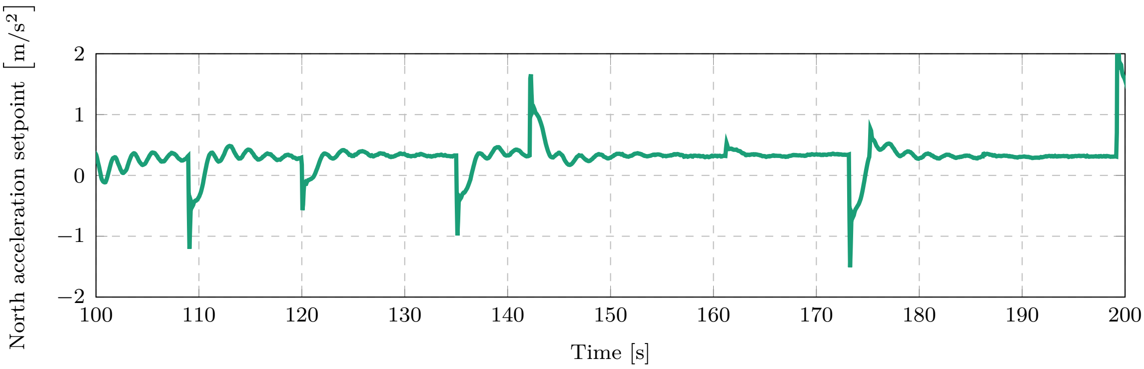



**Figure 4.10:** Accelleration setpoint training data from random velocity step inputs ($m_p = 0.2\,\mathrm{kg}$, $l = 1\,\mathrm{m}$)

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 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

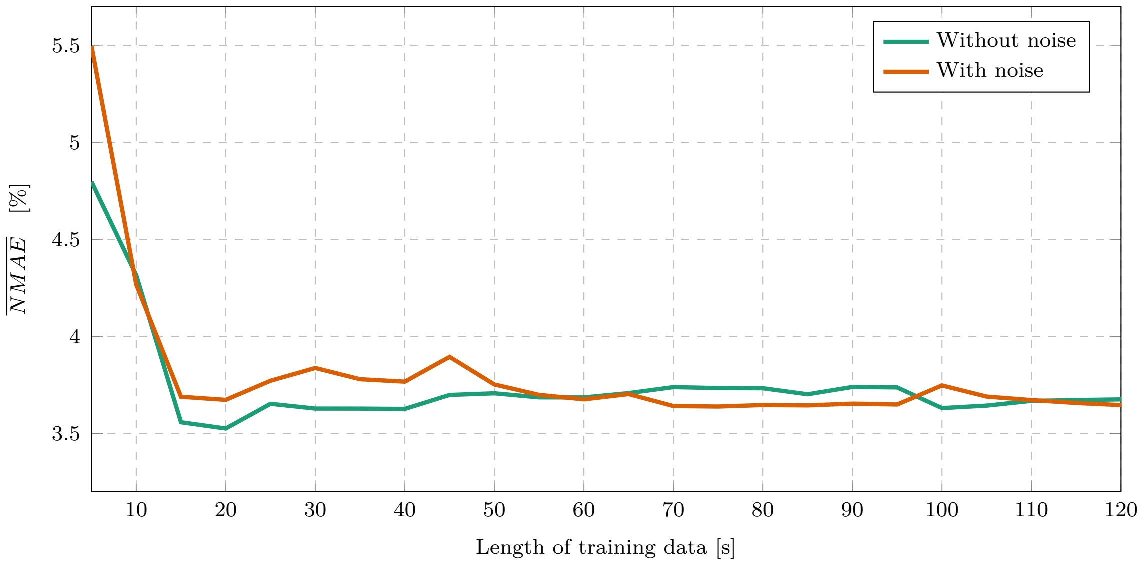



**Figure 4.11:** HAVOKc prediction errors for different lengths of training data with and without noise ($m_p = 0.2\,\mathrm{kg}$, $l = 0.5\,\mathrm{m}$, $T_s = 0.03\,\mathrm{s}$).

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.

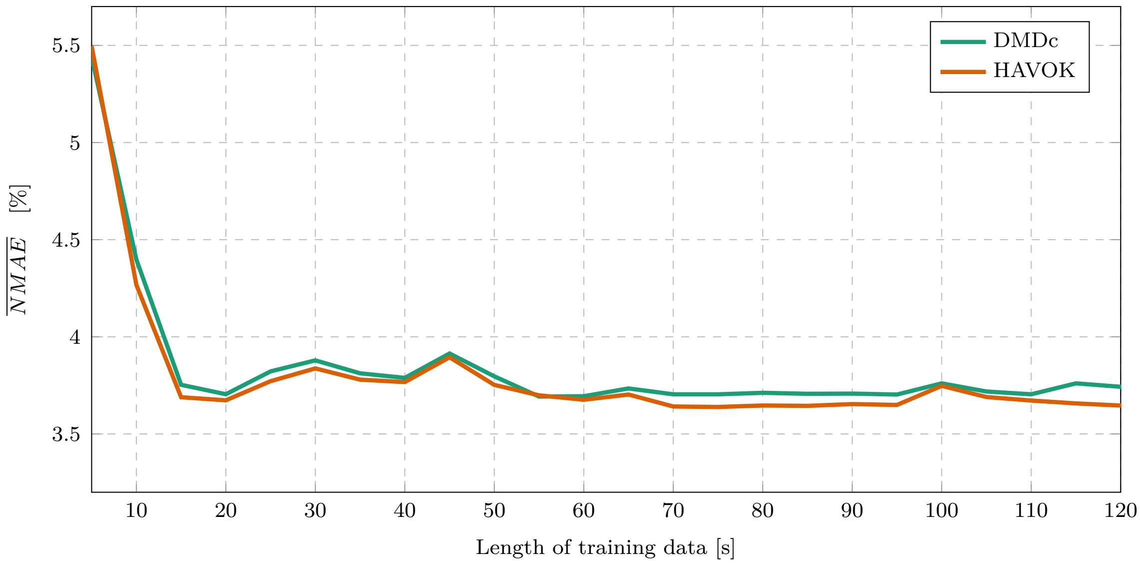
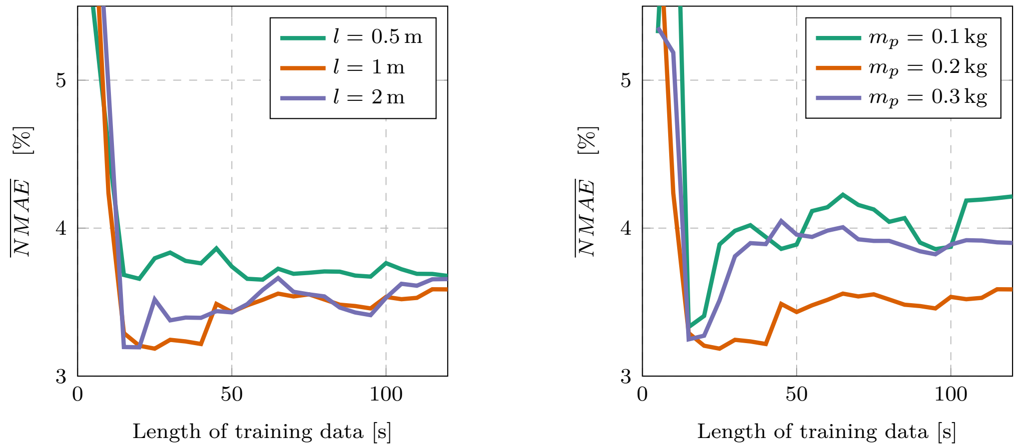



**Figure 4.12:** DMDc and HAVOKc prediction errors for different lengths of noisy training data ($m_p = 0.2\,\mathrm{kg}$, $l = 0.5\,\mathrm{m}$, $T_s = 0.03\,\mathrm{s}$).

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 simlar, with HAVOKc producing slightly lower prediction errors than DMDc. However, this difference in error may be so small that it has a negligible effect on control.




**(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

### 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, $N_{train}$ [24]. In this critical region at the transition between overparameterized and under-parameterized models, the prediction error initially decreases, then increases to a peak whereafter it decreases again [24].

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

$$D = (q \cdot n_x)^2 + (q \cdot n_x)(n_u) = 3660. \tag{4.23}$$

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

$$T_{train} = N_{train} \cdot T_s = 54.9\,\mathrm{s}. \tag{4.24}$$

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 $l = 0.5\,\mathrm{m}$ 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 [3] and [2] 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.

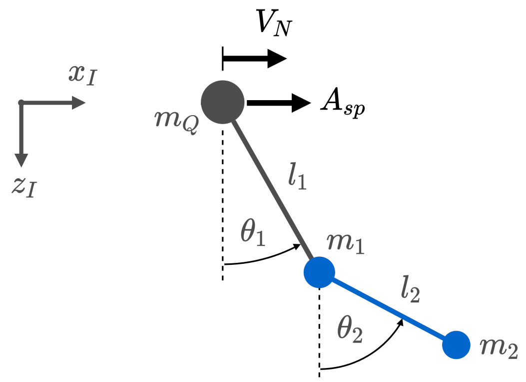



**Figure 4.14:** Double pendulum model representing an elongated suspended payload

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 4.3.2. 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 4.16 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.

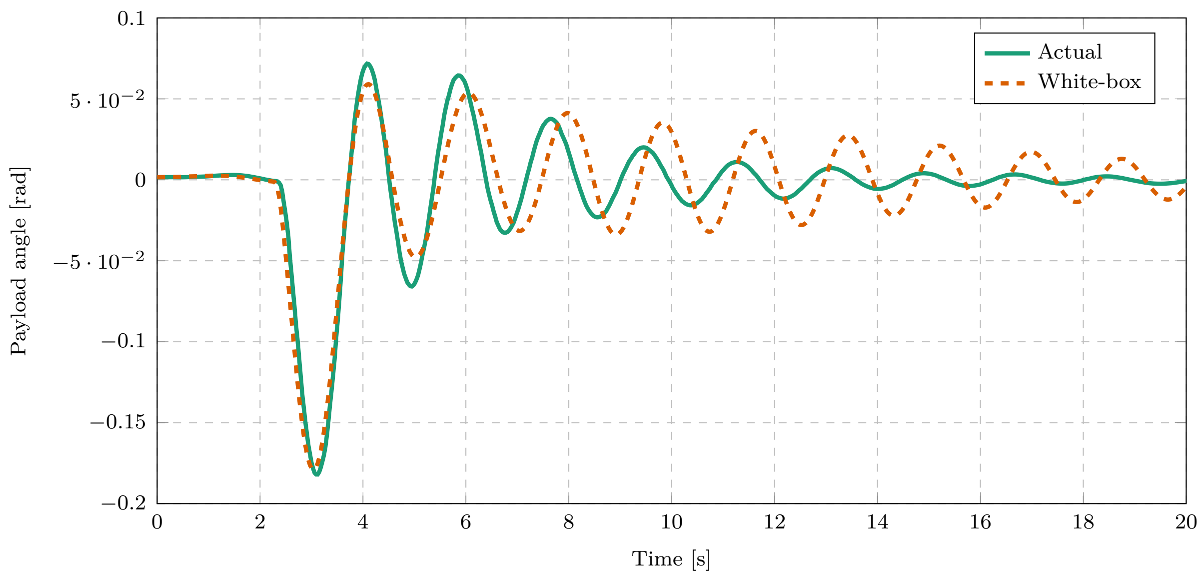



**Figure 4.15:** White-box model predictions of a single pendulum for a North velocity step input ($l = 1\,\mathrm{m}$, $m_p = 0.3\,\mathrm{kg}$.)

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 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.

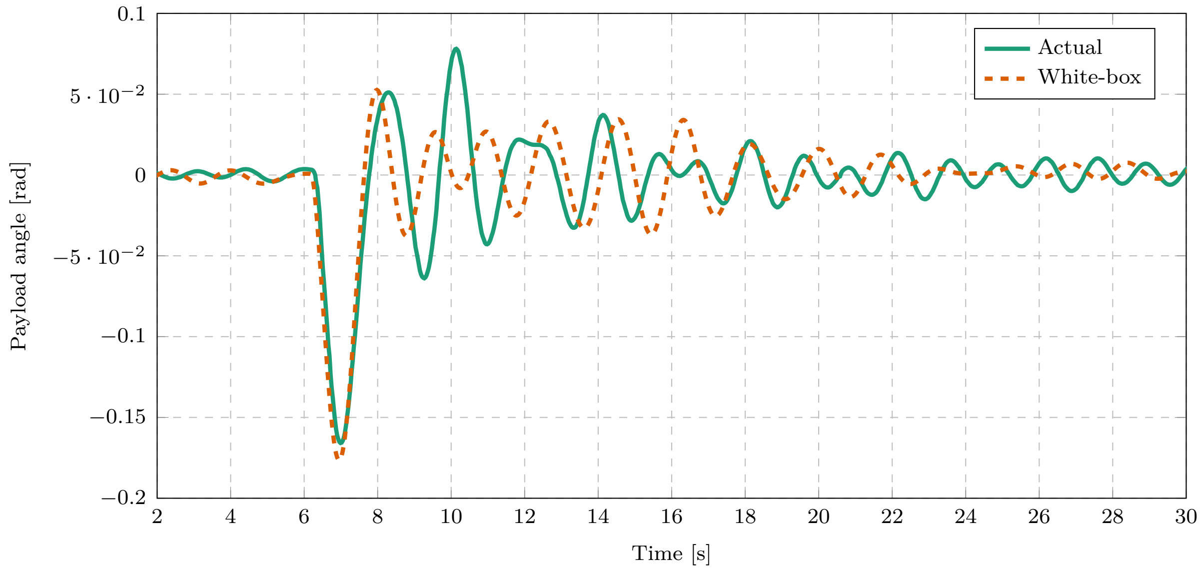
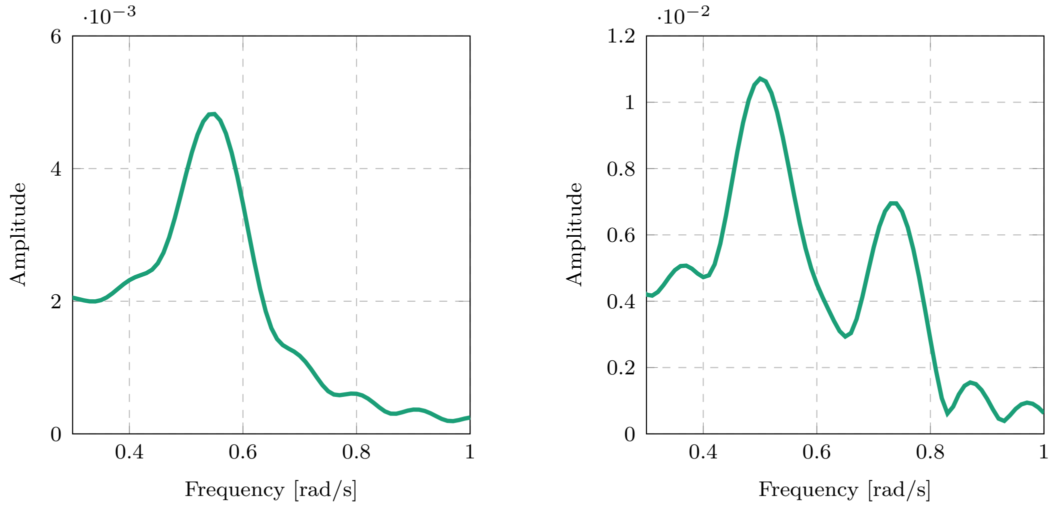



**Figure 4.16:** White-box model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 1\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.3\,\mathrm{m}$)

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.

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 [15]. 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




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

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

method for a much larger range of payload types without needing a redesign for specific payload dynamics.

## 4.7. Conclusion

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 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 a MPC on a multirotor companion computer. 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.

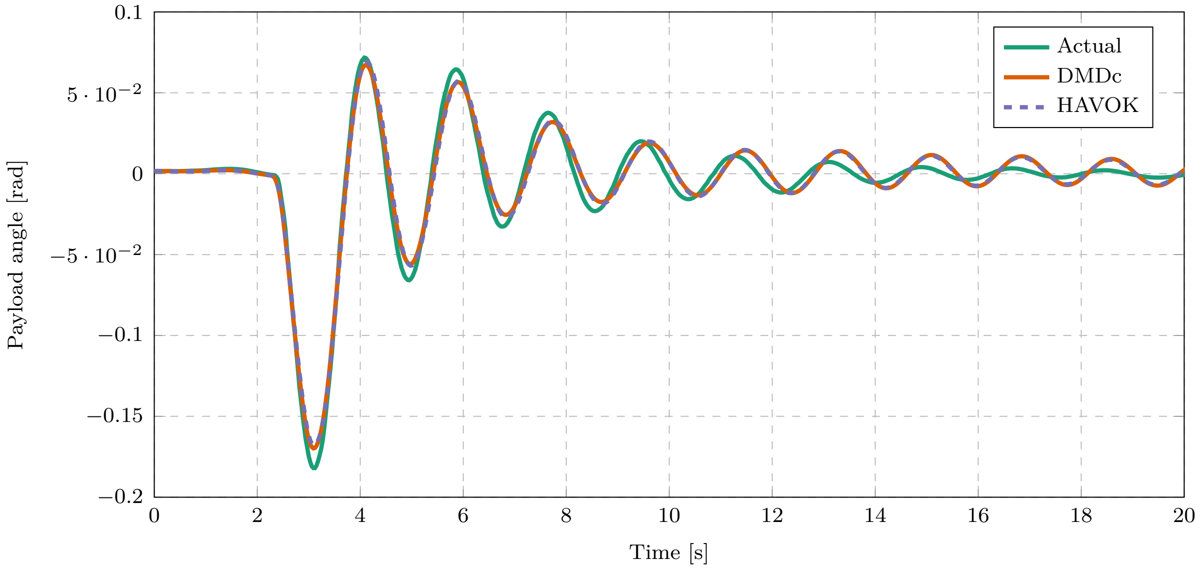



**Figure 4.18:** Data-driven model predictions of a single pendulum for a North velocity step input ($m_p = 0.3\,\mathrm{kg}$, $l = 1\,\mathrm{m}$).

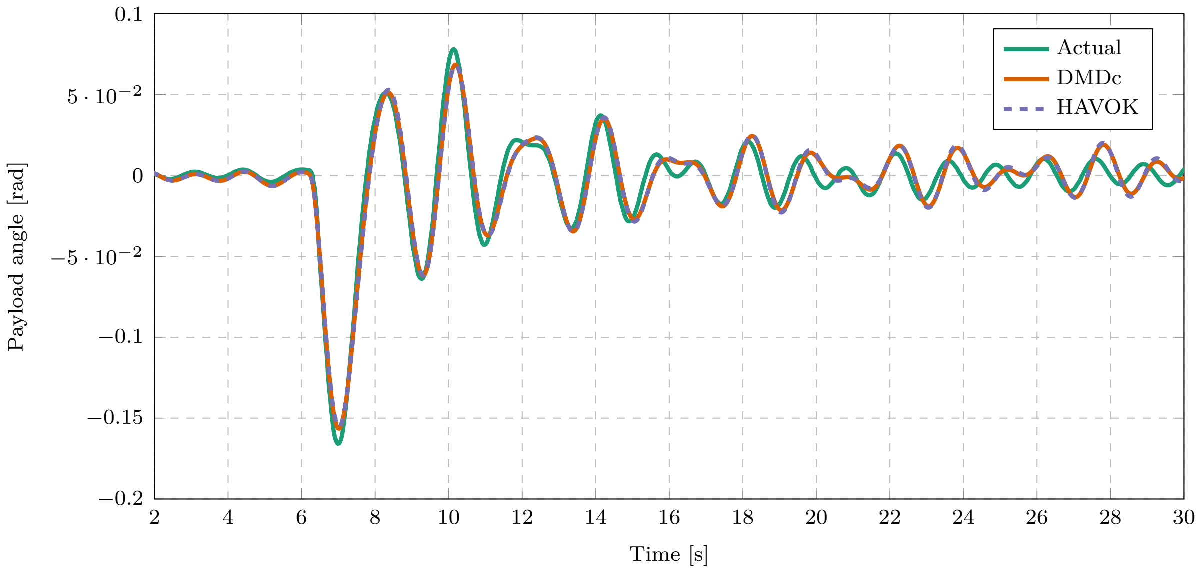



**Figure 4.19:** Data-driven model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 1\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.3\,\mathrm{m}$)

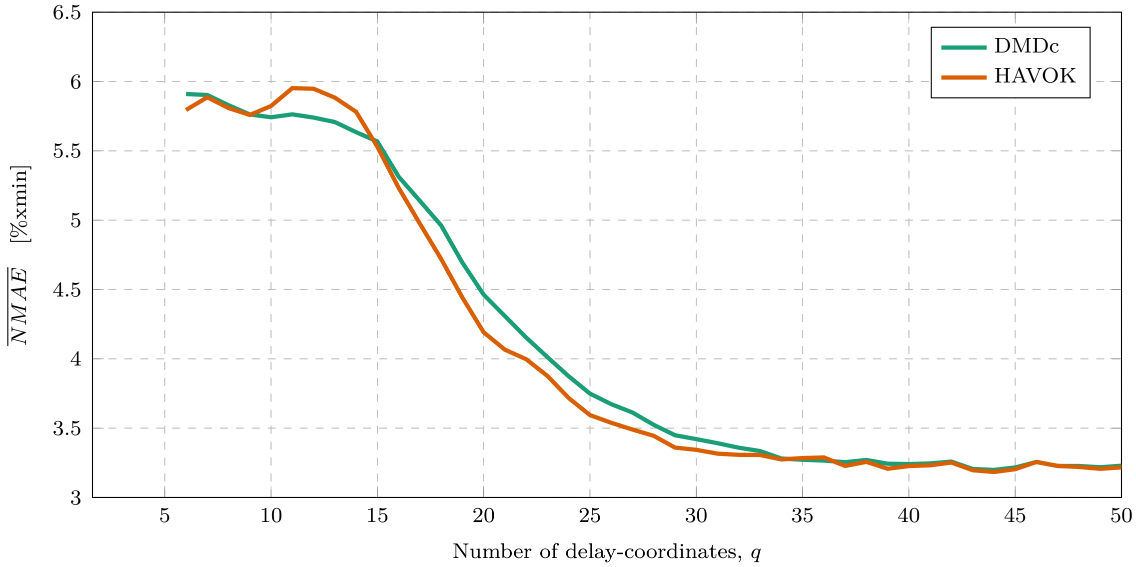



**Figure 4.20:** DMDc and HAVOKc predictions error of double pendulum for different numbers of delay-coordinates ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 1\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.3\,\mathrm{m}$ $T_{train} = 70\,\mathrm{s}$.)

# Chapter 5

# Control systems

## 5.1. Cascaded PID

Control system design Slower for less swing angles

## 5.2. LQR

Anton and Francois stuff

## 5.3. MPC

MATLAB QP solver C+ generation

## 5.4. Implentation 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 along with the appropriate controllers. It was firstly shown in Chapter 4 that the system identifcation techniques worked in simulation. Emphasise that control is now applied in a full SITL simulation.

MATLAB is used to generate a MPC ROS node This ROS node receives state feedback from the Gazebo simulator, computes the optimal control action, and sends the setpoint to PX4 through the package 'mavros'.

**Table 5.1:** Summary of the system identification techniques paired with the active damping controllers.

| System identification | | Controller |
|---|---|---|
| Category | Algorithm | |
| White-box | RLS mass estimator, and FFT cable length estimator | LQR |
| Black-box | DMDc, or HAVOK | MPC |

### 5.4.1. Single pendulum

- subplot prediction of data driven model. subplot prediction of white-box model
- plot MPC vs LQR v PID (no wind) step = 1 m/s
- plot MPC vs LQR v PID (no wind) step = 2 m/s
- plot DMDc vs HAVOKc (DMDc - Havok)
- plot with wind disturbacne control

### 5.4.2. Double pendulum

- subplot prediction of data driven model. subplot prediction of white-box model
- plot MPC vs LQR v PID (no wind)
- plot DMDc vs HAVOKc (DMDc - Havok)

# Chapter 6

# Experimental design

## 6.1. Hardware

Introduce honeybee

## 6.2. Software

# Chapter 7

# Practical implementation and results

In previous chapters, it was shown with simulation data that both the white-box and black-box system identification models can accurately represent the dynamics of a multirotor with a suspended payload. However, practical flights may differ significantly from simulations, which would affect the performance of these techniques. Wind is a common cause of unmeasured disturbances that influences the flight 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. The effect of different system configurations and wind conditions on the performance of these techniques will be investigated. These techniques will also be applied and evaluated with a practical dynamic payload. HITL simulations will also be performed to determine whether the proposed hardware can handle the computational complexity of these algorithms. Finally, the practical feasibility of the proposed system identification and control architectures will be discussed.

## 7.1. Methodology

As discussed in Chapter 4, generating data for the parameter estimation techniques involves two distinct flight stages. Firstly, the multirotor hovers with the suspended payload to gather data for payload mass estimation. A velocity step setpoint is then commanded to stimulate the swinging payload system for cable length estimation. Hence, the same methodology used for simulations will be used for practical flights.

For the data-driven techniques, the generation of practical training and testing data also follows the same general methodology as simulated flights:

1. Data logging starts when the multirotor is armed
2. Takeoff and hover with the multirotor
3. Command velocity step setpoints

4. Land the multirotor

5. Data logging stops when the multirotor is disarmed

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. Perform model predictions over the testing data to calculate an error metric

Figure 7.1 shows the Honeybee multirotor with a suspended payload during a practical flight. Numerous flights were performed with different payload masses, cable lengths, wind conditions, and dynamic payloads. The system identification methods were then performed on data resulting from this wide range of different use cases.

The major differences between the simulated and practical flights involve the attachment of the payload and wind disturbances. 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 disturbances which were not considered in simulations. The measurement noise experienced by a practical multirotor may also differ from the noise models used in simulations. Therefore this effect will also be investigated in the sections below.

**Figure 7.1:** Practical flight with Honeybee and a suspended payload

## 7.2. Parameter estimation with practical data

In Section 4.3, parameter estimation was performed with simulation data. It was shown that the models with the estimated parameter values provide reasonably accurate representations of the simulated dynamics. In this section, practical flight data will be used for parameter estimation. The effect of wind on the parameter estimation techniques will also be investigated.

### 7.2.1. Simple payload cable length estimation

As discussed in Section 4.3.2, 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) will be 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 interpretted as the percentage of the actual length by which the estimated length differs from the actual length.

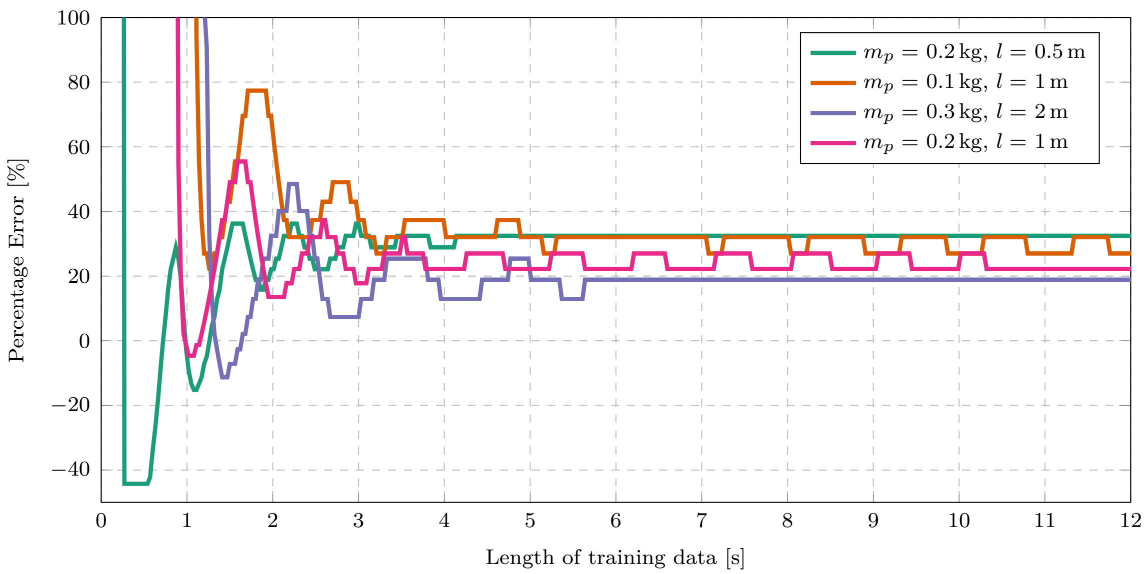



**Figure 7.2:** Plot of the error in cable length estimation as a function of length of training data (wind speed $\approx 0.5\,\mathrm{m/s}$)

Figure 7.2 shows the PE of the cable length estimation with different payload masses and cable lengths. 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 incorrect length is estimated from the frequency peak identified in the FFT.

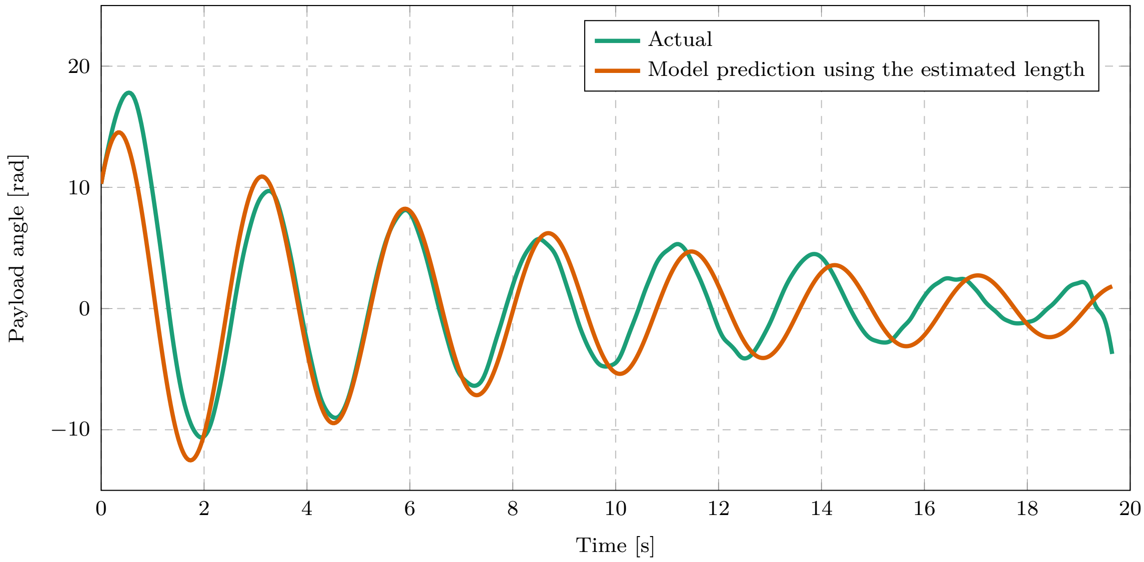



**Figure 7.3:** White-box model prediction for a North velocity step input ($l = 0.5\,\mathrm{m}$, $m_p = 0.2\,\mathrm{kg}$.)

Figure 7.3 compares the actual payload angle to the predicted angle of the white-box model for a practical flight. The cable length estimated from this flight is 0.617 m resulting in a PE of 23.4 %. It is clear that the prediction differs significantly from the actual data in magnitude and in frequency. A large difference is expected at such large swing angles, because the model was a linearised with the small angle approximation (as discussed in Chapter 3). Therefore, the actual non-linear dynamics differ significantly from the linearised model for this data. Note that an incorrect damping coefficient used in the model does can not contribute much to the modelling error. The damping coefficient largely effects the rate of attenuation of the amplitude of the oscillations, but does not have a large effect on the initial swing angle of the predicted by the model. The initial swing angle of the practical data is clearly very different from the model prediction.

The naive assumption regarding the speed of the inner loop PID controller probably adds to the modelling error significantly. As discussed in Chapter 3, the white-box model assumes that the time scale seperation between the velocity and attitude controller is large enough for the acceleration setpoint to approximate the actual seperation. However, it appears that the attitude controller of the practical multirotor is slow enough to significantly affect the transient response of the swing angle. Notice how the first two oscillation peaks differ

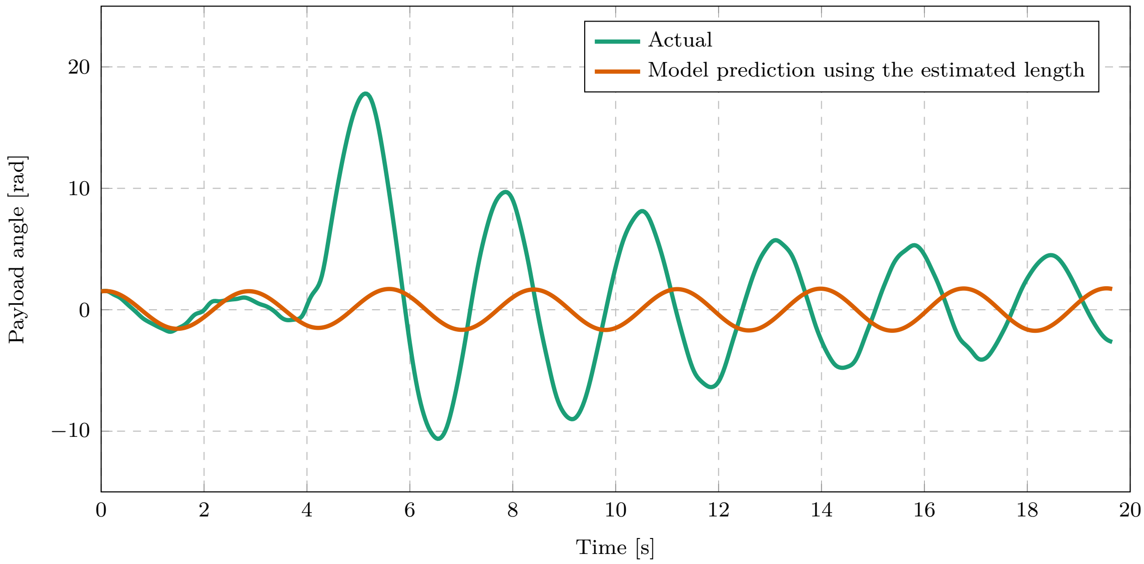



**Figure 7.4:** White-box model prediction for a North velocity step input ($l = 0.5\,\text{m}$, $m_p = 0.2\,\text{kg}$.)

noticeably from the expected linear damping shape. Since the payload is attached below the CoM of the multirotor, the oscillating pitch angle of the multirotor also affects the swing angle of the payload.

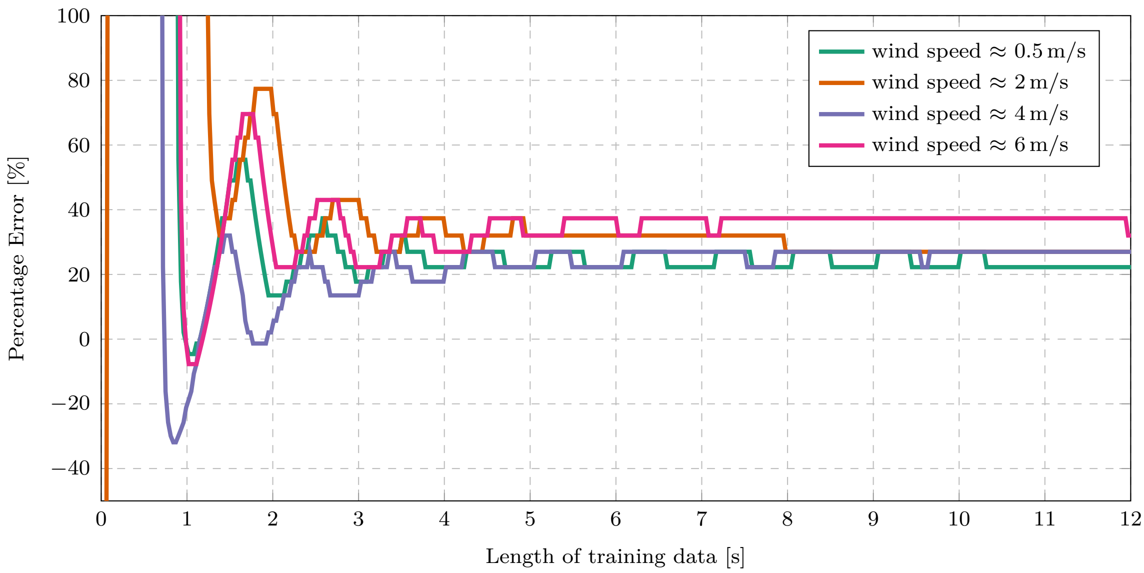



**Figure 7.5:** Cable length estimation error as a function of length of training data with wind distrubances ($m_p = 0.2\,\text{kg}$, $l = 1\,\text{m}$)

Figure 7.5 shows the PE of cable length estimation for flights with different wind conditions. These flights were all performed with the same payload. It appears that the wind speed affects the parameter estimation results since the estimation error differs significantly for different wind speeds. This may be due to the variable damping effect the controllers have on the payload 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 7.5 that the estimation PE converges for each considered flight with a different wind speed, 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.

### 7.2.2. Dynamic payload cable length estimation

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

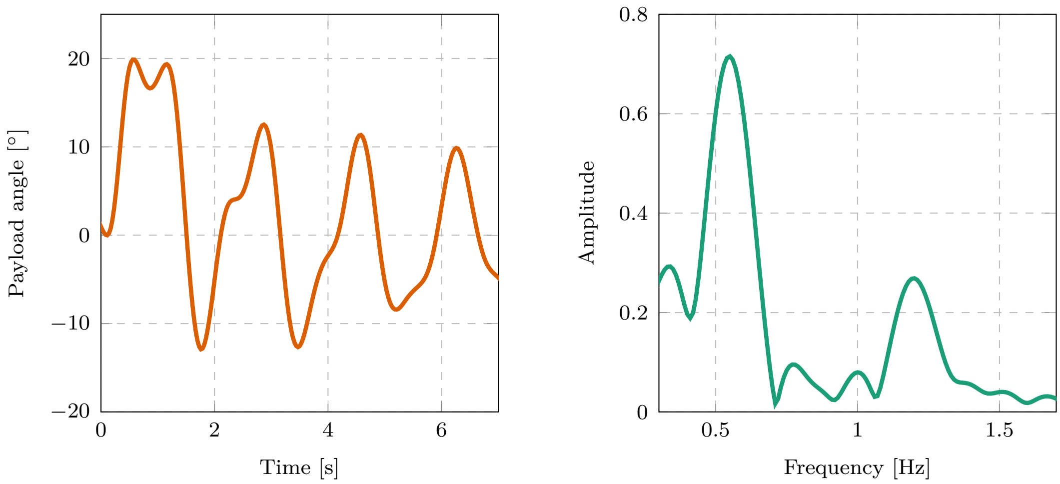
Figure 7.6a shows a snapshot of payload angle data from a practical flight with a dynamic payload. Two superimposed frequencies are clearly visible in the payload oscillations due to the double pendulum action of the elongated pendulum. Two peaks corresponding to these two frequencies can be identified from the resulting FFT amplitude spectrum in Figure 7.6b. 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 7.7 shows the estimated cable length as a function of the length of training data for a practical dynamic payload. Note how the estimated length converges after a sufficient length of training data. The estimated cable length is 1.03 m, which is used in the white-box model. Figure 7.8 plots the measured and predicted payload angle for this flight. It is clear that the model prediction is completely different from the actual payload angle.

As the size of the payload swing angles decrease due to damping, 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. However, the simple pendulum model is still a bad representation of the practical data.

Figure 7.9

Overall,




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

**Figure 7.6:** Data from a velocity step response with a dynamic payload ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 0.5\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.6\,\mathrm{m}$).

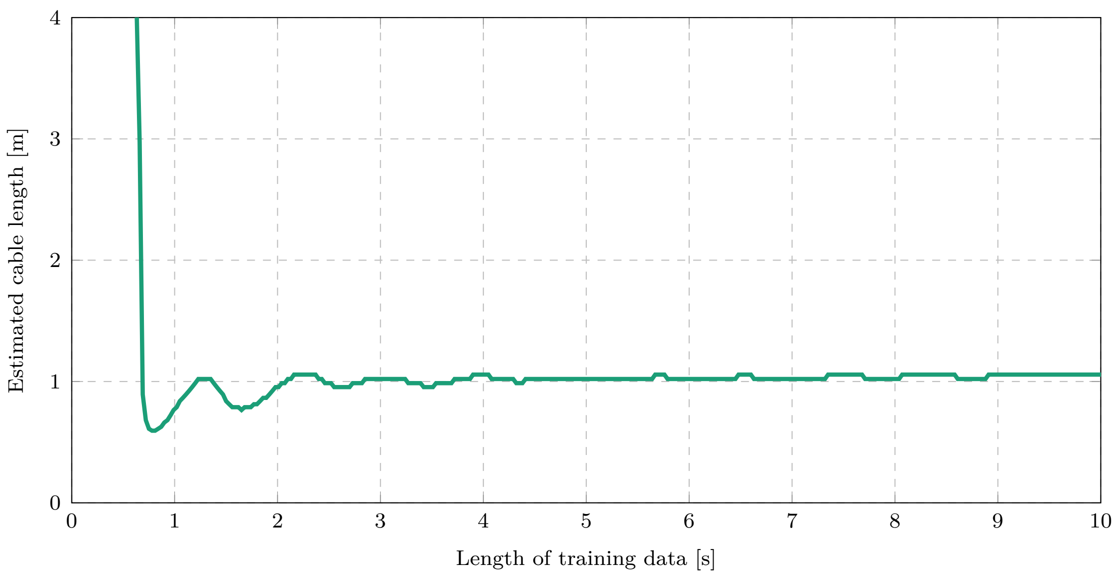



**Figure 7.7:** Estimated cable length as a function of length of training data for a dynamic payload ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 0.5\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.6\,\mathrm{m}$)

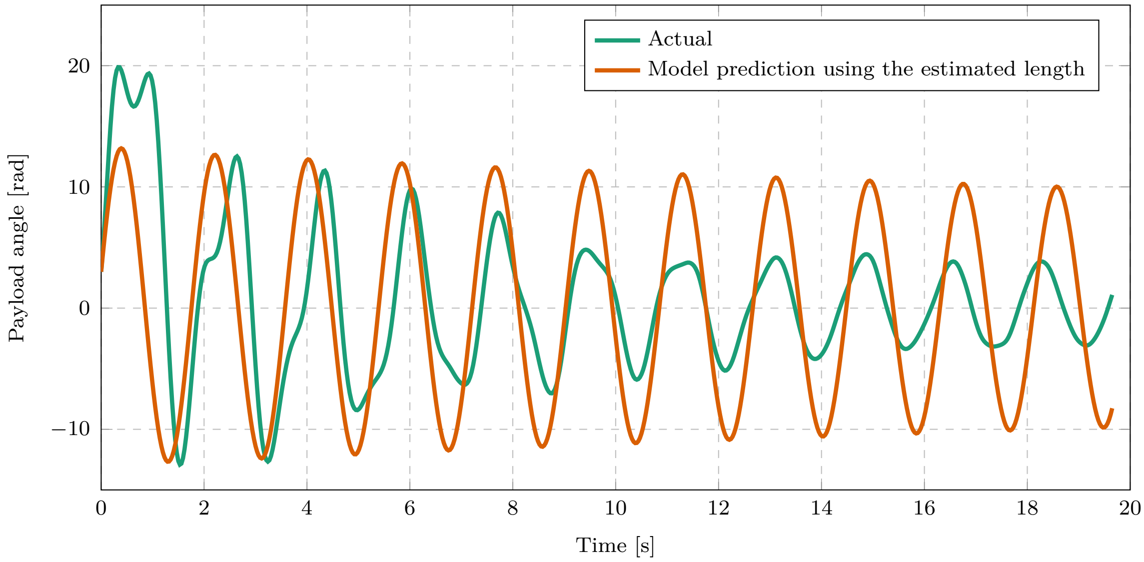



**Figure 7.8:** White-box model prediction for a North velocity step input for a dynamic payload ($m_1 = 0.2\,\text{kg}$, $l_1 = 0.5\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.6\,\text{m}$)

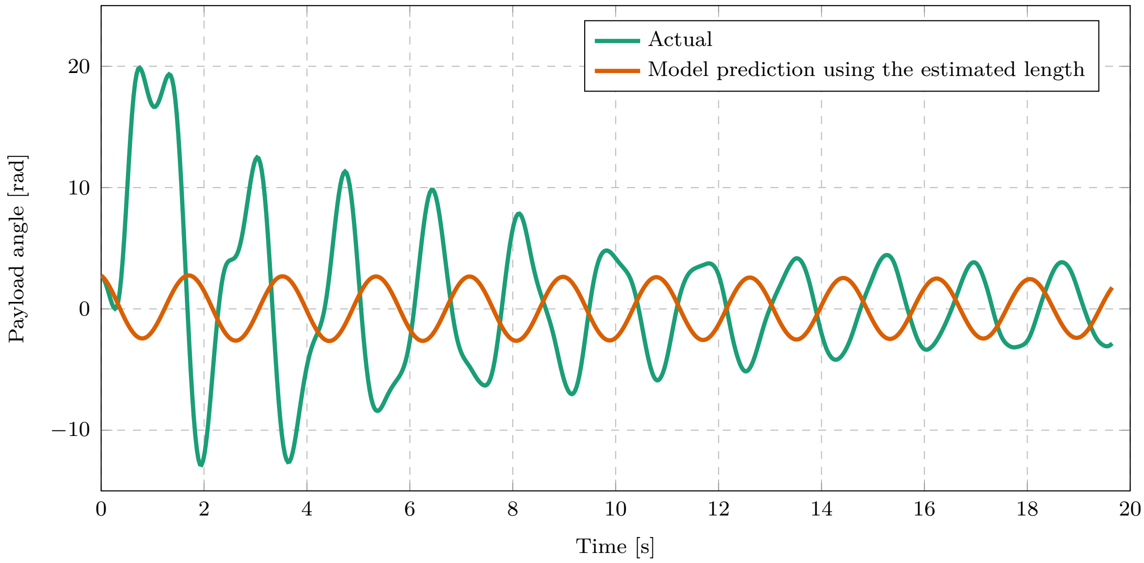



**Figure 7.9:** Example of a bad prediction with a white-box model for a dynamic payload ($m_1 = 0.2\,\text{kg}$, $l_1 = 0.5\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.6\,\text{m}$)

## 7.3. Data-driven system identification with practical data

In Chapter 4 it was shown that the 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 flight. Therefore the data-driven algorithms will be applied to practical flight data in this chapter to investigate their performance in a practical implementation.

### 7.3.1. 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 (also referred to as process noise) 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 determine accurate drag coefficients of the practical system for a realistic simulation . The mean force applied to the multirotor by the wind affects the mean offset in acceleration setpoint data because the velocity controller integrators compensates 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 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.

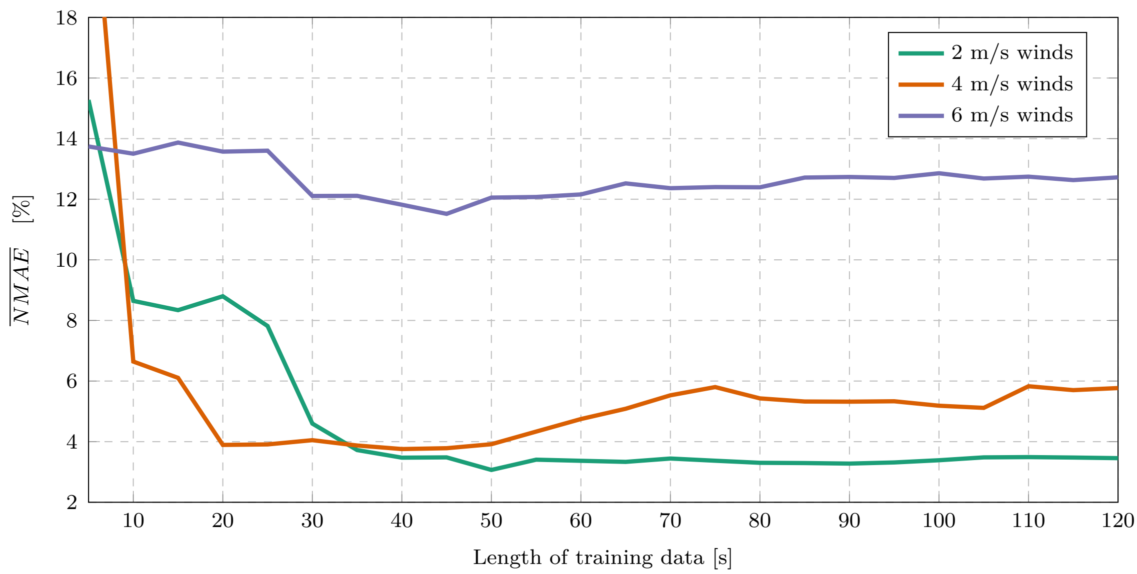



**Figure 7.10:** DMDc prediction errors as a function of length of training data for practical data with different wind conditions ($m_p = 0.2\,\mathrm{kg}$, $l = 1\,\mathrm{m}$, $T_s = 0.03\,\mathrm{s}$).

Figure 7.10 shows prediction error as a function of training data length for data with different wind conditions. Wind conditions are referenced here by the wind speed recorded by the website, www.yr.no, for the hour of day of the flight. From 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 benificial for system identifcation. Note that the prediction error corresponding to 6 m/s winds does not vary much with length of training data. The prediction error at this wind speed is significantly large and a model generated from such data will probably not be useful for control.

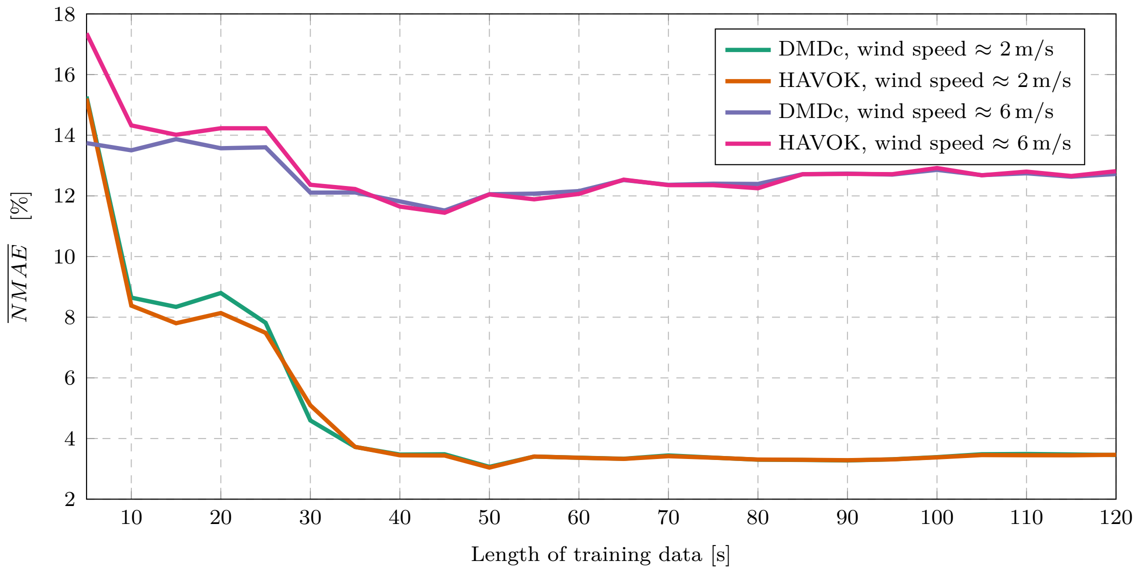



**Figure 7.11:** DMDc and HAVOKc prediction errors for different lengths of practical training data ($m_p = 0.2\,\mathrm{kg}$, $l = 1\,\mathrm{m}$, $T_s = 0.03\,\mathrm{s}$).

Figure 7.11 compares the prediction errors of DMDc and HAVOKc models. It is evident that the two techniques produce similar prediction errors for different wind conditions. The difference in prediction error is very small and will probably not effect the performance of the controllers using these models. This shows that the minor difference in the implementation of the algorithms has a negilible effect on the resultant models. The DMDc implementation is therefore prefered over HAVOKc due to a lower computational complexity.

### 7.3.2. 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 anymore. Figure 7.12 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 with simulation data, the elbow can still be identified.

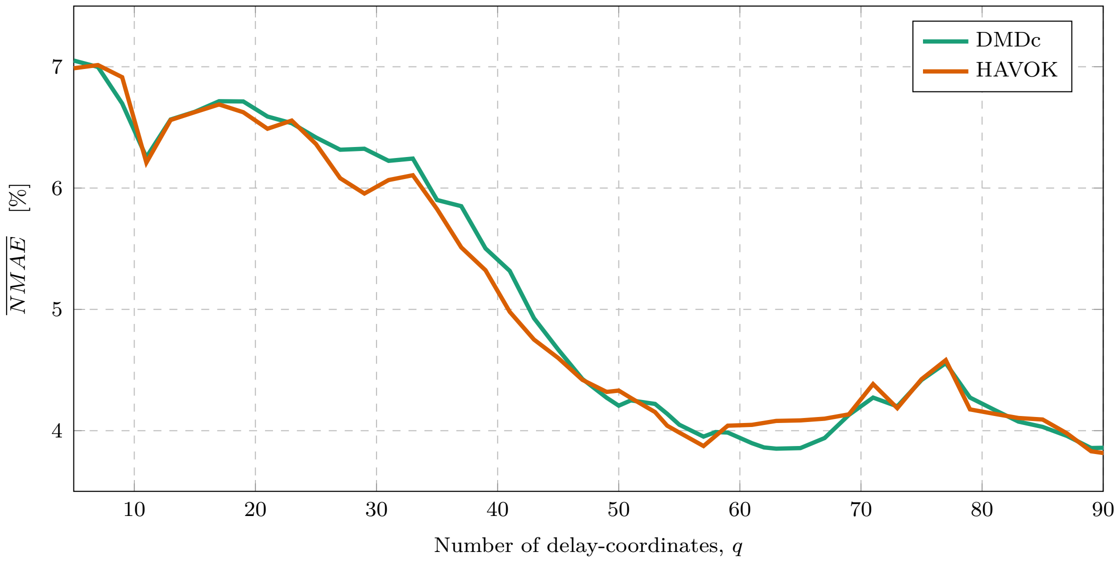



**Figure 7.12:** DMDc and HAVOKc prediction errors for different number of delays included in the model ($m_p = 0.2\,\mathrm{kg}$, $l = 1\,\mathrm{m}$, $T_s = 0.03\,\mathrm{s}$, wind speed $\approx 2\,\mathrm{m/s}$).

### 7.3.3. 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 7.13 shows the prediction error for different payloads with practical data. Only DMDc predictions are plotted here because HAVOK predictions are so similar. This shows that the proposed methods also work in practice with different payload configurations. The 'double-descent' trend is clearly seen in the plots where the prediction error increases slightly after a specific length of training data.

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. Hence the effect of system parameters such as multirotor mass, payload mass, cable length, and damping coefficients are inherintly included in the estimated model. Therefore these other parameters can also be varied and the system identification algorithm will still be able to determine a prediction model of the system.

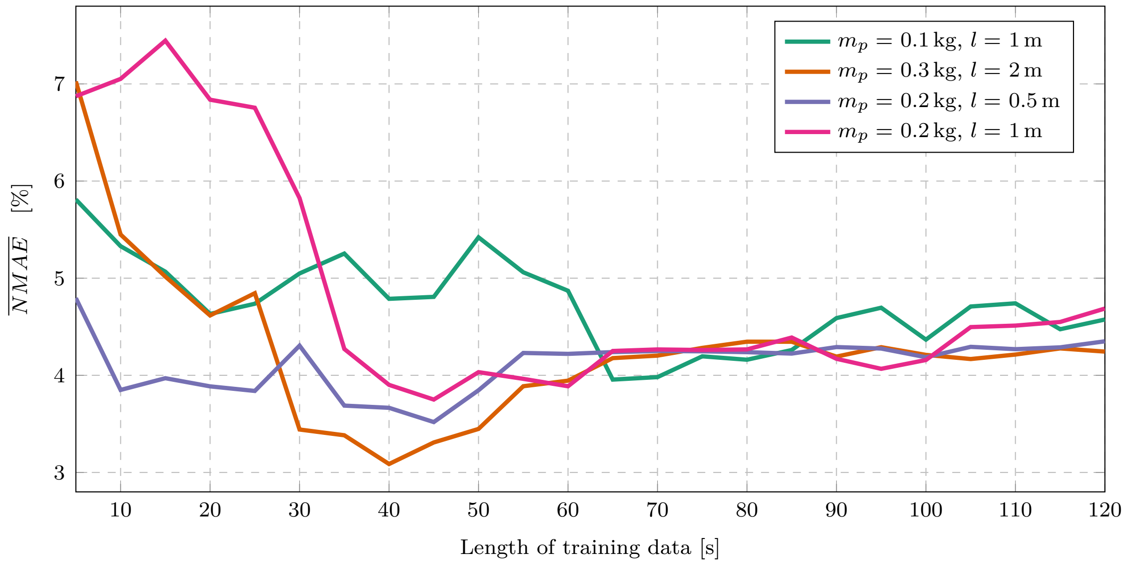



**Figure 7.13:** DMDc prediction error as a function of training data length for different payload parameters

## 7.3.4. State predictions

Figure 7.14 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 7.14 plots the state prediction against testing data. Recall from Section 4.4 that DMDc produces a discrete, state-space model in the form:

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

The state prediction starts at the initial condition, $\boldsymbol{x}_0$ and $\boldsymbol{d}_0$, and predicts the state vector for each succesive 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 which accumulates with each succesive prediction. Therefore the prediction error increases as the prediction horizon increases, as shown in Figure 7.14.

Figure 7.15 shows the measured and predicted North velocity of the same flight. The oscillations in the velocity response due to the swinging payload is clearly visible in this plot. The model predicts the frequency and size of these oscillations reasonably well. Note that prediction of the payload angle is significantly easier than predicting the velocity response. The payload angle prediction inherantly 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 7.3.1) may result in a large error in the velocity prediction due to a build up in integration error. However, despite this challenge, the model accurately

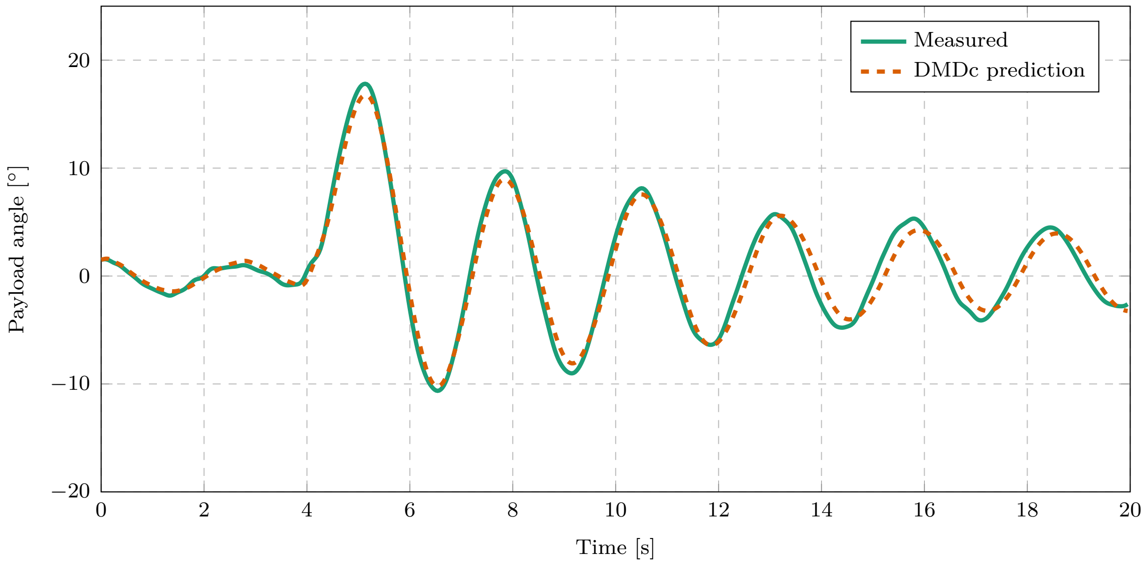



**Figure 7.14:** Model predictions of practical flight data with an suspended payload for a North velocity step input ($l = 2\,\mathrm{m}$, $m_p = 0.3\,\mathrm{kg}$)

predicts the velocity step size of the practical data.

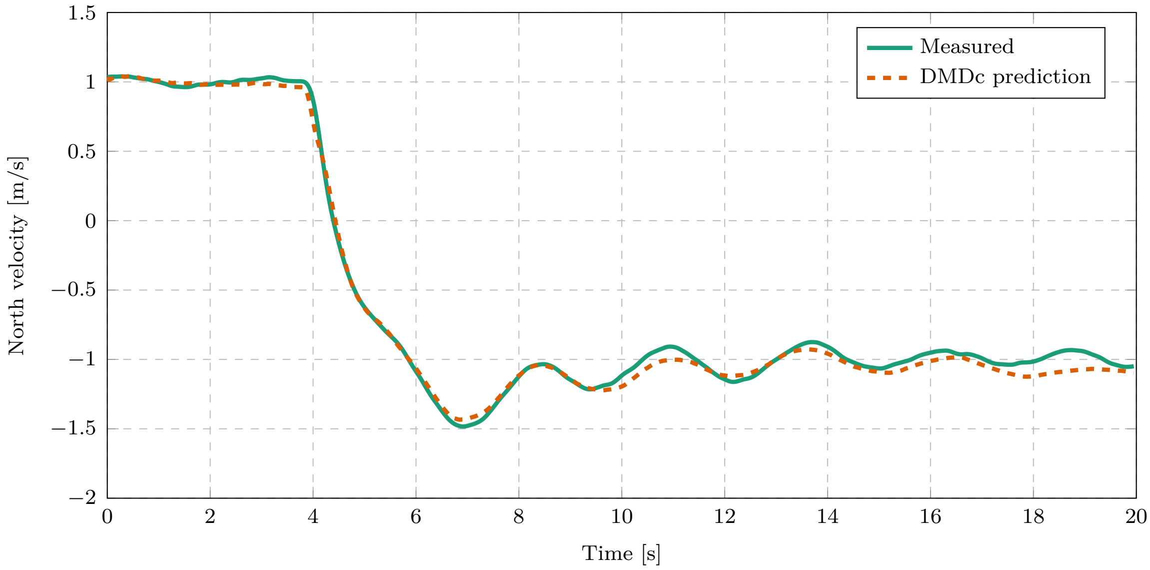



**Figure 7.15:** Model predictions of practical flight data with a suspended payload for a North velocity step input ($l = 2\,\mathrm{m}$, $m_p = 0.3\,\mathrm{kg}$, wind speed $\approx 0.5\,\mathrm{m/s}$)

## 7.3.5. Extended dimensions

Since the data-driven methods are only dependant on the input and output data used, the prediction model can be easily extended to include more dimentions 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 axis simulateously.

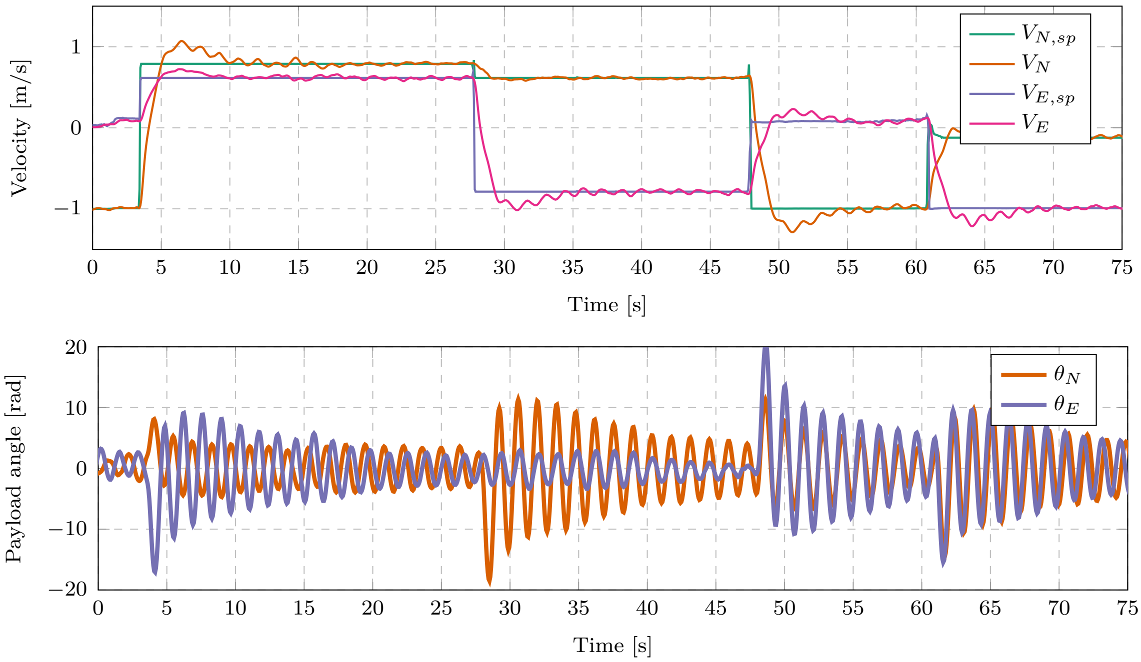



**Figure 7.16:** Snapshot of training data with random velocity step inputs for the North and East axes ($m_p = 0.2\,\mathrm{kg}$, $l = 0.5\,\mathrm{m}$)

For this model, the new 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 simulateously to stimulate the dynamics in both axes. Figure 7.16 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 7.17 shows the state variable predictions of a DMDc model build from the data in Figure 7.16. It is clear that this model provides an accuarte prediction of each state variable considered. This shows that the data-driven methods can be effectively extended to include both the North and East axes dynamics.

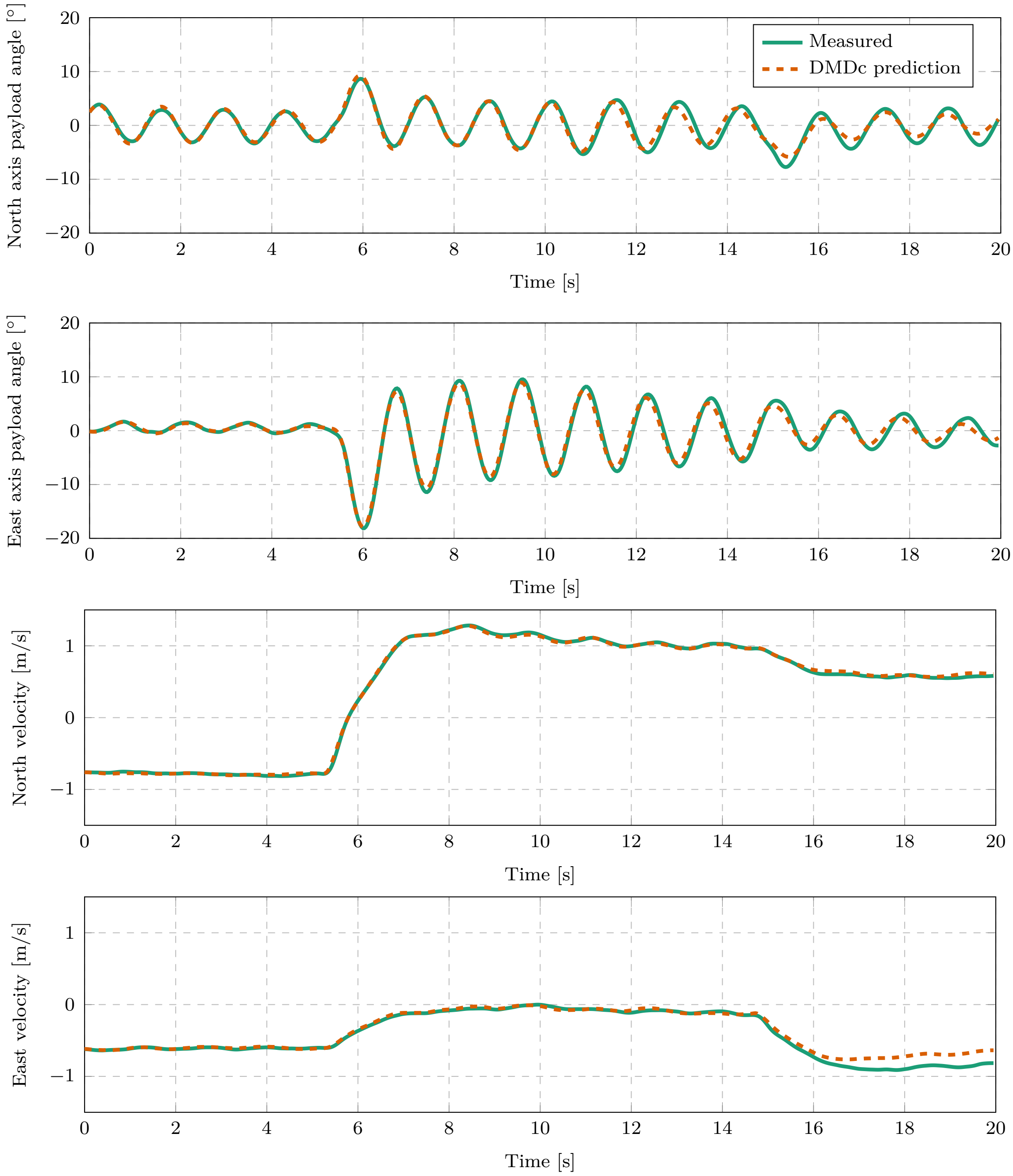



**Figure 7.17:** Data-driven predictions of practical data for a model with both North and East axis dynamics

### 7.3.6. Dynamic payload

As mentioned in Chapter 4, one of the disadvantages of the white-box system identification approach is that it relies heavily on a priori modeling 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 7.18:** Practical flight with an suspended elongated payload attached to Honeybee

One of the a prioir 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 7.18 shows a photo of an elongated payload suspended from Honeybee during a practical flight This is a practical example of a dynamic payload which deviates significantly from the point mass assumption. The mass distribution causes a relative rotation of the payload with respect to the suspended cable, which significantly affects the flight dynamics.

Figure 7.19 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 is clearly seen in the angle data. 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 7.20 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 these oscillations in the velocity response still appear irregular compared

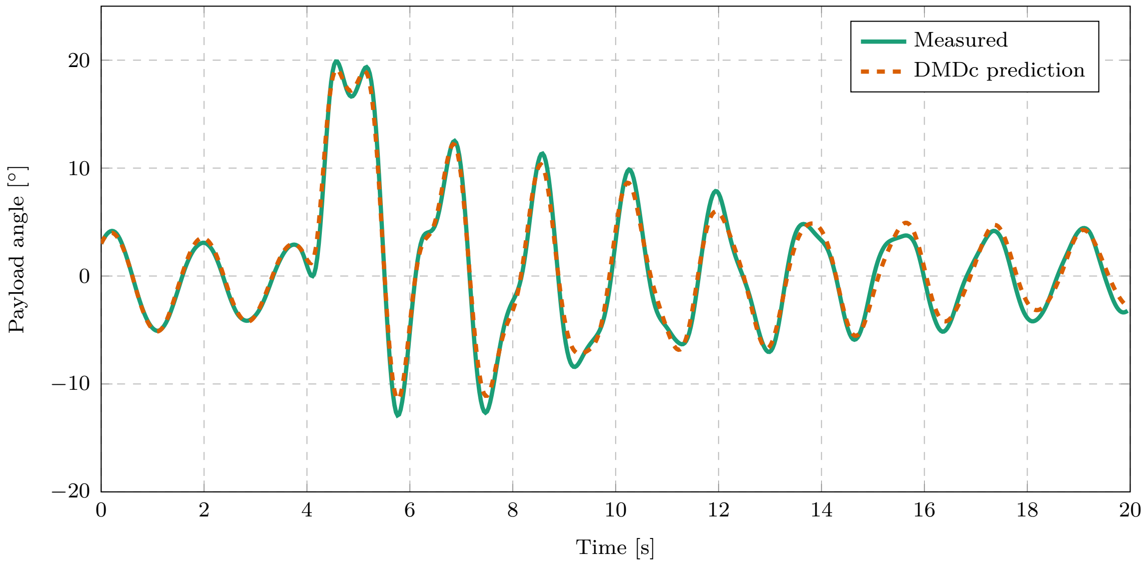



**Figure 7.19:** Practical flight data and model predictions with an elongated payload for a North velocity step input ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 0.5\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.6\,\mathrm{m}$)

to the simple payload data shown in Figure 7.15. In Figure 7.20, 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 using this model. The shape of the velocity oscillations also appear to be captured well in the prediction. Recall that the prediction is propagated from an initial condition and the given input data only. The model does not use state measurements to readjust after the initial condition is taken. Therefore a 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.

## 7.4. HITL

## 7.5. Conclusion

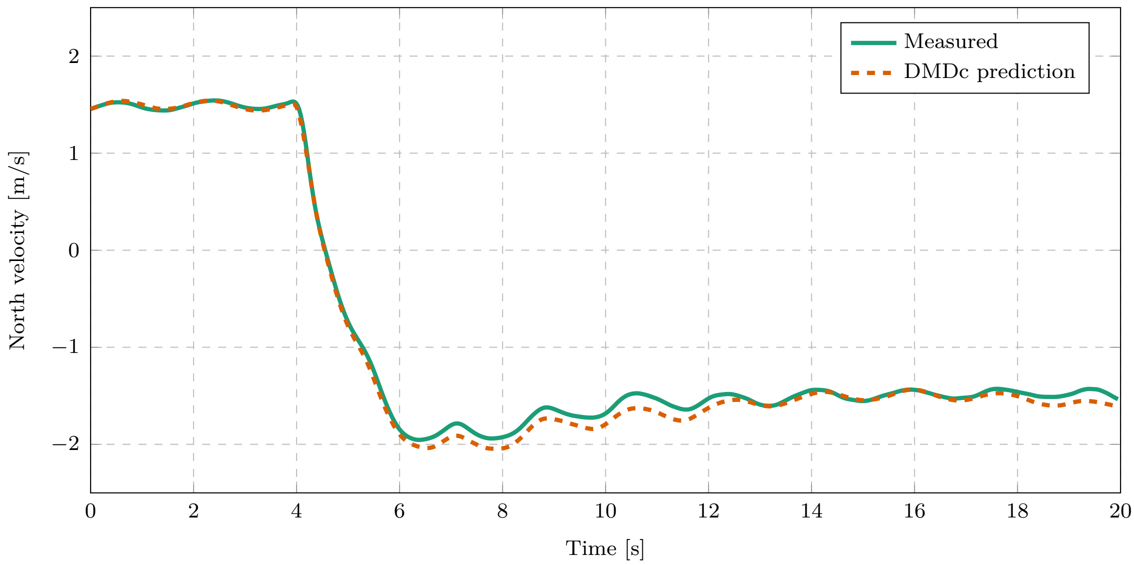



**Figure 7.20:** Practical flight data and model predictions with an elongated payload for a North velocity step input ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 0.5\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.6\,\mathrm{m}$)

# Chapter 8

# Summary and Conclusion

# Bibliography


[1] Compare Commander, “Drones for search & rescue missions,” aug 2020. [Online]. Available: https://altigator.com/en/drones-for-search-rescue-missions/

[2] A. P. Erasmus and H. W. Jordaan, “Stabilization of a Rotary Wing Unmanned Aerial Vehicle with an Unknown Suspended Payload,” no. March, 2020.

[3] J. F. Slabber and H. W. Jordaan, “Vision-Based Control of an Unknown Suspended Payload with a Multirotor Unmanned Aerial Vehicle,” Ph.D. dissertation, 2020.

[4] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, apr 2016. [Online]. Available: https://www.pnas.org/content/113/15/3932https://www.pnas.org/content/113/15/3932.abstract

[5] 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. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2008-6964

[6] 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, dec 2014. [Online]. Available: https://www.aimsciences.org/article/doi/10.3934/jcd.2014.1.391

[7] 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.

[8] M. Korda and I. Mezić, “Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control,” *Automatica*, vol. 93, pp. 149–160, nov 2018. [Online]. Available: http://arxiv.org/abs/1611.03537http://dx.doi.org/10.1016/j.automatica.2018.03.046

[9] H. Arbabi, M. Korda, and I. Mezic, “A Data-Driven Koopman Model Predictive Control Framework for Nonlinear Partial Differential Equations,” *Proceedings of the IEEE Conference on Decision and Control*, vol. 2018-Decem, pp. 6409–6414, 2018.

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

[11] 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.

[12] 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, dec 2017.

[13] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22 445–22 451, nov 2019. [Online]. Available: https://www.pnas.org/content/116/45/22445https://www.pnas.org/content/116/45/22445.abstract

[14] S. L. Brunton and J. N. Kutz, *Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control*, 2017. [Online]. Available: databook.uw.edu

[15] M. Kamb, E. Kaiser, S. L. Brunton, and J. N. Kutz, "Time-Delay Observables for Koopman: Theory and Applications," *https://doi.org/10.1137/18M1216572*, vol. 19, no. 2, pp. 886–917, apr 2020.

[16] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., jun 2015, pp. 6235–6240.

[17] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149–2154, 2004.

[18] 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, jan 2014.

[19] 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, oct 2006.

[20] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2219, 2018.

[21] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, "Model selection for dynamical systems via sparse regression and information criteria," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2204, p. 20170009, aug 2017. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspa.2017.0009

[22] S. L. Brunton and J. N. Kutz, "Regression and Model Selection," in *Data-Driven Science and Engineering*, 2019, vol. 1, pp. 117–153.

[23] ——, "Balanced Models for Control," in *Data-Driven Science and Engineering*, 2019, pp. 321–344.

[24] 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.

# Appendix A

# PID gains

This is an appendix about PID gains used for Honeybee.

# Appendix B

# Random appendix

This is another appendix.