# Inhoudsopgawe

# Hoofstuk 1

# Introduction

# Hoofstuk 2

# Literature study

# Hoofstuk 3

# Modelling

This chapter discusses the mathematical modelling of a quadrotor with a suspended payload which is based on a practical quadrotor UAV named Honeybee. The model is first derived as a 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 quadrotor and suspended payload. 1

## 3.1. Coordinate frames

## 3.2. States

## 3.3. Forces and moments

## 3.4. Lagrangian mechanics

## 3.5. Linearised model

## 3.6. Discretised model

## 3.7. Model verification

## 3.8. Dynamic payloads

water looks like double payload

# Hoofstuk 4

# System identification

System identification is the process of creating mathematical models of a dynamical system by using input and output measurements of that system. Two major approaches are used to represent the dynamics of such a system:

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

Models determined from a priori modelling and parameter estimation are referred to as white-box models. In contrast, data-driven system identification methods result in black-box models. This chapter discusses these system identification approaches and describes the differences between them. For each approach, different estimation techniques are explained and applied to the quadrotor and payload system. The results of these techniques are then compared to each other.

## 4.1. White-box and black-box techniques

### 4.1.1. White-box techniques

The underlying physics of a white-box model is understood by the user because they are determined from first principles. This is done by modelling physical processes with techniques like Lagrangian mechanics or Newton equations. With system identification techniques that use these models, the mathematical relations between system parameters are predefined in the modelling phase. The system identification process is therefore reduced to parameter estimation to determine values for parameters used in die model.

This approach is used by [1] and [2] for swing damping control of a quadrotor 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 CoM of the quadrotor.

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

The approach works well for systems with predictable dynamics, but it is not very adaptable. The payload considered by [1] and [2] is limited to a small rigid mass suspended from the quadrotor by a non-stretching cable. In this use case it was shown that a LQR controller successfully controls a quadrotor while minimising payload swing angles. However, if a payload or cable is used that violates one of the modelling assumptions, the predefined model no longer accurately represent the system. Since the controller is dependent on this model, the mismatch between the model and actual dynamics may result in undesirable controller behaviour.

### 4.1.2. Black-box techniques

Data-driven system identification methods produce black-box models. In contrast to white-box models, black-box 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.

Black-box models can be categorised as either non-linear or linear models. Non-linear models are often more accurate than linear models because complex, real-world dynamics are better approximated by non-linear systems. The dynamics of a quadrotor and suspended payload are also non-linear. Examples of black box models with quadrotors and payloads in literature ???

However, non-linear models are inherently more complex than linear models. Controllers that use non-linear models are usually more computationally complex than those with linear models. Control archetictures for quadrotors used in practical applications are mostly implemented on onboard hardware. Therefore there is value in low-complexity, linear models since these may be simple enough to execute on low cost hardware. trade-off between accuracy and complexity. Non-linear models may require control implementations that are too computationally expensive and may not be practically realisable on the available hardware on a quadrotor.

DMDc and HAVOK are the two data-driven system identification methods investigated in this paper. These are linear regression techniques that produce a linear model to approximate non-linear dynamics. Non-linear data-driven techniques like Neural Networks and SINDy [**?**] may produce models that are more accurate than linear techniques, but at the cost of greater computational complexity. Name more techniques DMDc and HAVOK are less computationally complex and their models are suitable for linear MPC, which is significantly faster than non-linear MPC. This is desirable for the quadrotor use case, where onboard computational power is limited.

These techniques and their implementation are explained in the sections below. Each technique is considered for use in a velocity controller in the North direction

**Considered controller** The model identified by DMDc will be used to design a longitudinal velocity controller. As shown in **??**, the plant considered for system identification includes the dynamics of the inner loop, attitude controllers. The swing damping controllers which will utilise the identified model act only in the translational velocity loop. Because of the large time-scale separation between the inner and outer loop controllers, the attitude states have a negilable effect on the plant dynamics seen by the velocity controller. As discussed in Section 3.5, the payload minimally effects the quadrotor attitude because it is attached near the CoM of the vehicle. Therefore the attitude states are excluded from the system identification model.

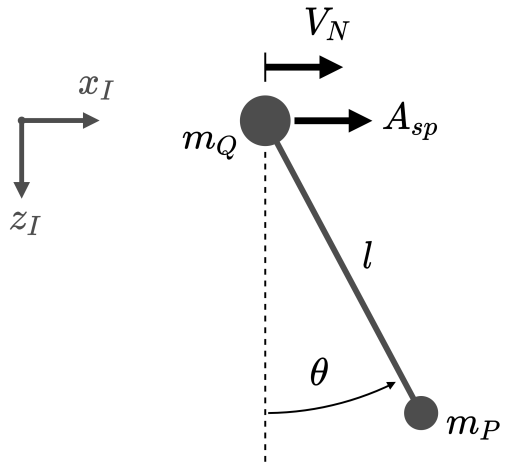
## 4.2. Plant considered for system identification

**Derived model** Figure 4.1 shows the plant considered for system identification. In Chapter 3 the differential equations that describe the motion of this system are derived with Lagrangian mechanics. From these equations it is clear that the considered plant is defined by the state vector,

$$\boldsymbol{x} = \begin{bmatrix} V_N & \dot{\theta} \end{bmatrix}^T, \tag{4.1}$$

and the input vector,

$$\boldsymbol{u} = \begin{bmatrix} A_{N,sp} \end{bmatrix}, \tag{4.2}$$




**Figuur 4.1:** Floating pendulum model considered for system identification for a North velocity controller

## 4.3. Parameter estimation

### 4.3.1. Predetermined linear model

The motivation for paramater estimation is to determine unknown parameter values required by the predetermined model. This model was derived a prioiri in Section 3.5.

### 4.3.2. Payload mass estimation

RLS Cannot work if quadrotor mass also changes, because assume that $m_q$ is known.

### 4.3.3. Cable length estimation

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

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

The natural frequency is measured by performing a FFT on the payload swing angle response after a position step by the quadrotor. The dominant frequency identified by the FFT during free swing is the natural frequency of the payload.

**??** shows the payload swing angle after the system is stimulated by a position step setpoint. As shown in **??** the first few seconds of the step response are not used in the FFT.

This is to minimise the effect of the quadrotor controllers on the swing angle frequency by excluding the transient response in the FFT.

**??** shows the resulting amplitude spectrum of the payload swing angle response. The dominant frequency is clearly identified as ??. Since $m_q$ and $g$ is known, and $m_p$ and $\omega_n$ has been estimated, $l$ can now be determined from 4.3. In this case the estimated length is ??, compared to the actual length of ??.

Frequency resolution ?? error for different lengths??

## 4.4. Dynamic mode decomposition with control

**Intro** DMD is a linear regression technique that can be used to approximate a non-linear dynamical system [3]. 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 [4]. This section provides an overview of the specific implementation of DMDc used in this paper. Note that this implementation is a slight adaptation of DMDc, and includes time-delay-embedding of multiple observables. [5] and [6] use time-delay-embedding in their DMD adaptions in similar ways.

**State space model** DMD produces a linear, discrete state-space model of the 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 [7]. Hence, we define an state delay vector as:

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

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

The discrete state-space model is therefore defined as:

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

$\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$ is the system matrix, $\boldsymbol{A}_1 \in \mathbb{R}^{(q \cdot n_x) \times (q \cdot n_x)}$ is the state delay system matrix and $\boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}$ is the input matrix.

**Training data** The training data consists of full-state measurements, $\boldsymbol{x}_k$, and corresponding inputs, $\boldsymbol{u}_k$, taken at regular intervals of $\Delta t = T_s$, during a simulated flight with Cascaded PID control. In a practical flight, these time-series measurements need to be

saved in memory because it is usd as a single batch by DMD. Note that DMD can be applied in a recursive manner as described in [8], However this implementation is not considered because memory size will not be a limitation since a companion computer will be used.

**Data matrices** The training data is collected into the following matrices:

$$\begin{aligned}
\boldsymbol{X'} &= \begin{bmatrix} \boldsymbol{x}_{q+2} & \boldsymbol{x}_{q+3} & \boldsymbol{x}_{q+4} & \cdots & \boldsymbol{x}_{w+q+1} \end{bmatrix}, \\
\boldsymbol{X} &= \begin{bmatrix} \boldsymbol{x}_{q+1} & \boldsymbol{x}_{q+2} & \boldsymbol{x}_{q+3} & \cdots & \boldsymbol{x}_{w+q} \end{bmatrix}, \\
\boldsymbol{X}_d &= \begin{bmatrix} \boldsymbol{x}_q & \boldsymbol{x}_{q+1} & \boldsymbol{x}_{q+2} & \cdots & \boldsymbol{x}_{w+q-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_2 & \boldsymbol{x}_3 & \boldsymbol{x}_4 & \cdots & \boldsymbol{x}_{w+1} \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 & \cdots & \boldsymbol{x}_w \end{bmatrix}, \\
\boldsymbol{\Upsilon} &= \begin{bmatrix} \boldsymbol{u}_q & \boldsymbol{u}_{q+1} & \boldsymbol{u}_{q+2} & \cdots & \boldsymbol{u}_{w+q-1} \end{bmatrix},
\end{aligned} \tag{4.6}$$

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 $\boldsymbol{\Upsilon}$ [4]. In order to group the unknowns into a single matrix, (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.

**SVD** A 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 [9]. Talk about Reduced Order Modelling?? POD modes?? 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. explain this better 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. maybe insert colour pictures showing matrices

By combining (4.9) with the over-constrained equality in (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 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

q = number of delays, from here up

HAVOK is a data-driven, regression technique that provides a connection between DMD and Koopman operator theory [**?**, 9]. We have adapted the standard HAVOK algorithm slightly to account for the effect of control and to extract a discrete, linear model that approximates the behaviour of a controlled dynamical system. In this section, a brief overview is provided for this implementation and expansion of HAVOK.

The extracted discrete state-space model is defined as:

$$\boldsymbol{a}_{k+1} = \tilde{\boldsymbol{A}}\boldsymbol{a}_k + \tilde{\boldsymbol{B}}\boldsymbol{u}_k, \tag{4.11}$$

where $\boldsymbol{a}_k$ is the state vector previously defined in Section 4.4, $\tilde{\boldsymbol{A}} \in \mathbb{R}^{(q\cdot n_x)\times(q\cdot n_x)}$ is the system matrix, and $\tilde{\boldsymbol{B}} \in \mathbb{R}^{(q\cdot n_x)\times n_u}$ is the input matrix. Here, $\tilde{}$ is used to differentiate these matrices from $\boldsymbol{A}$ and $\boldsymbol{B}$ used in DMDc.

The original HAVOK algorithm, developed by [10], constructs a Hankel matrix from output variables only. In order to incorporate the effect of control, an extended Hankel matrix, $\boldsymbol{\Pi}$, is created by appending a matrix of inputs to a Hankel matrix of measurements:

$$\boldsymbol{\Pi} = \begin{bmatrix} \boldsymbol{a}_q & \boldsymbol{a}_{q+1} & \boldsymbol{a}_{q+2} & \cdots & \boldsymbol{a}_{w+q-1} \\ \boldsymbol{u}_q & \boldsymbol{u}_{q+1} & \boldsymbol{u}_{q+2} & \cdots & \boldsymbol{u}_{w+q-1} \end{bmatrix}, \tag{4.12}$$

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

$$\mathbf{\Pi} \approx \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^T, \tag{4.13}$$

where $\tilde{}$ represents rank-$p$ truncation. It is important to note that the model extracted by HAVOK depends on the choice of hyperparameters, $p$ and $q$. The number of samples in the training data, $N_{train} = w + q - 1$, also influences the accuracy of the model.

The columns of $\tilde{\boldsymbol{V}}$ are the most significant principal components of the system dynamics [11]. 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 [10]. Consider the following discrete, state-space formulation:

$$\boldsymbol{v}_{k+1} = \mathbf{\Lambda}\boldsymbol{v}_k. \tag{4.14}$$

Recall that DMDc finds a best fit linear operator that directly maps $\boldsymbol{a}_k$ to $\boldsymbol{a}_{k+1}$. Similarly, HAVOK determines the best fit linear operator $\mathbf{\Lambda}$ that maps the pseudo-state $\boldsymbol{v}_k$ to $\boldsymbol{v}_{k+1}$. So, in order to setup an over-determined equality for (4.14), $\tilde{\boldsymbol{V}}^T$ is divided into two matrices:

$$\begin{aligned} \boldsymbol{V}_1 &= \begin{bmatrix} \boldsymbol{v}_q & \boldsymbol{v}_{q+1} & \ldots & \boldsymbol{v}_{w-1} \end{bmatrix}, \\ \boldsymbol{V}_2 &= \begin{bmatrix} \boldsymbol{v}_{q+1} & \boldsymbol{v}_{q+2} & \ldots & \boldsymbol{v}_w \end{bmatrix}, \end{aligned} \tag{4.15}$$

where $\boldsymbol{V}_2$ is $\boldsymbol{V}_1$ advanced a single step forward in time. The matrices from Equation (4.15) are now combined with Equation (4.14) and the best fit $\mathbf{\Lambda}$ is determined with the Moore-Penrose pseudoinverse:

$$\boldsymbol{V}_2 = \mathbf{\Lambda}\boldsymbol{V}_1 \qquad \Rightarrow \qquad \mathbf{\Lambda} \approx \boldsymbol{V}_1\boldsymbol{V}_1^{\dagger} \tag{4.16}$$

It can be shown from Equation (4.13) that Equation (4.14) is transformed from the eigen-time-delay coordinate system to the original coordinate system as the following:

$$\begin{bmatrix} \boldsymbol{a}_{k+1} \\ \boldsymbol{u}_{k+1} \end{bmatrix} = (\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}})\mathbf{\Lambda}(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}})^{\dagger} \begin{bmatrix} \boldsymbol{a}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{4.17}$$

This form is used to extract $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ from the matrix, $(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}})\mathbf{\Lambda}(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}})^{\dagger}$, in the following way:

$$\begin{bmatrix} \boldsymbol{a}_{k+1} \\ \boldsymbol{u}_{k+1} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{A}} & \tilde{\boldsymbol{B}} \\ \textit{(discarded)} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{4.18}$$

Note that the matrix entries in (4.18) that map $\boldsymbol{u}_k$ to $\boldsymbol{u}_{k+1}$ are meaningless for our purposes and are discarded. Similarly to DMDc, some matrix entries in $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ are known a

priori due to the relative positions of delay coordinates. These are forced to 1 or 0 to improve the prediction performance of the model.

merge these paragraphs Since the state vector, $\boldsymbol{a}$, includes delay-coordinates, some matrix entries are known a priori and are independent of the dynamics. For example, the values of $\boldsymbol{x}_k$ should be mapped from their position in $\boldsymbol{a}_k$ to specific indices in $\boldsymbol{a}_{k+1}$. Due to the least-squares fitting and coordinate transformation, DMDc will not produce these exact values in $\boldsymbol{A}$ and $\boldsymbol{B}$. By forcing each of these matrix entries to 1 or 0, the state-prediction performance of the model is improved.

## 4.6. Implementation and results

In order to test the system identification techniques, numerous simulations were performed to investigate their performance with different system configurations. Firstly, the influence of design parameters on the algorithm performance are explored. 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 are also explored, like measurement noise, and the physical properties of the payload. Finally, the white-box and black-box techniques are tested on a dynamic payload which does not satisfy the assumptions of a simple pendulum.

### 4.6.1. Methodology

A SITL implementation of PX4 [12] using the Gazebo simulator [13] is used to generate data for system identification. Testing these techniques with simulation data allow us to investigate a much larger range 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 also applies realistic measurement noise to the signals received by PX4, which applies an EKF for state estimation. Therefore the the data seen by the system identification techniques will also include the filtering effect of the EKF as it would in practical flights.

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

1. Takeoff and hover with the qaudrotor
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

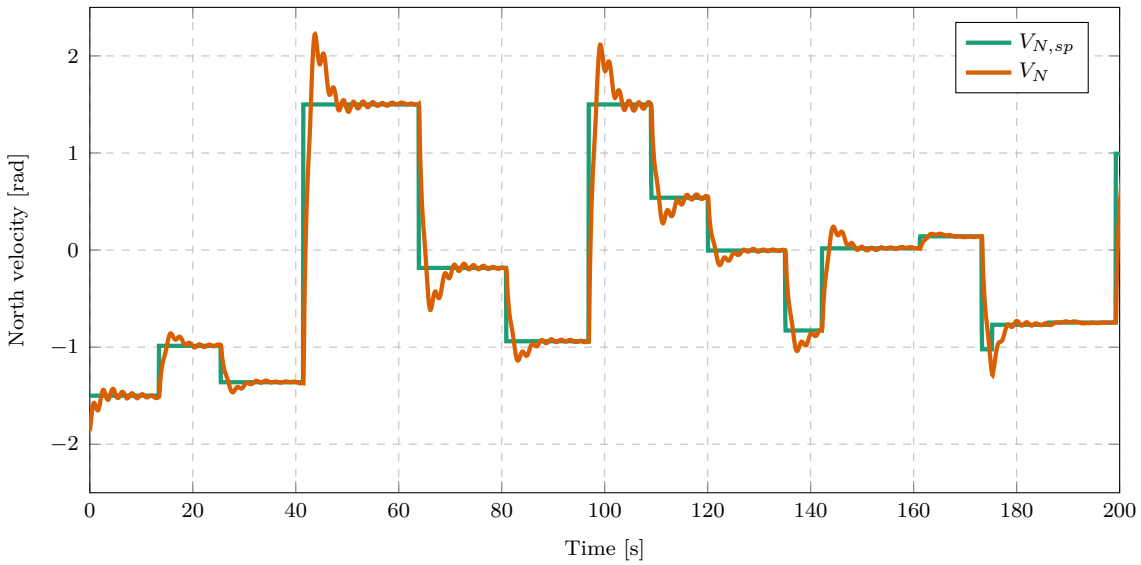
7. Calculate a error metric for the model from the testing data

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

A algorithm schedules the series of velocity step commands by assigning random step values and time-intervals within a specified range. Theses 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 quadrotor 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.

**Tabel 4.1:** Input data ranges.

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




**Figuur 4.2:** Example of training data with random velocity step inputs ($m = 0.2\,\text{kg}$, $l = 1\,\text{m}$)

Figure 4.2 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 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 predictions scores do not result from models that overfit to the training data.

The testing data spans a fixed length of time and taken from the start of the simulation period. The training data is then allocated from the remainder of the data. The same testing data is used to calculate error metrics for different models with different configurations to ensure that the metric for each model is comparable. This error metric calculated from the testing data is used to evaluate and rank the performances of different models.

### 4.6.2. Error metric

It is common practice is to select a model for a MPC based on k-step-ahead prediction errors [14]. 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 [14]. However for the quadrotor 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 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 $\boldsymbol{x_k}$ is the actual state vector at time-step $k$, $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ is the state prediction, and $\boldsymbol{MAE}$ is a vector with the MAE of each state.

Popular, scale-free error metrics, like MAPE, MRAE and MASE, are also based on the $l_1$ norm, but are independent of the scale and units of a variable [15]. 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 [16], 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 naÃŕve forecast [15]. However, an in-sample forecast is a naÃŕve 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.

$\text{NMAE}_{\text{mm}}$ is a scale-free error metric that can compare different variables in our use case. It normalises the MAE of a variable by the range of that variable, thereby variables with different means or scales can be compared. This value is calculated as:

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

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 $\text{NMAE}_{\text{mm}}$ of all state variables is used as the single value representing the overall accuracy of a model. 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 AIC and BIC scores. Thereby they provide a quantitative way of performing a Pareto analysis, which balances model complexity with model accuracy [17]. 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.

Maybe insert example of good initial prediction but bad long term ??

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 $\text{NMAE}_{\text{mm}}$ is determined for each run, whereafter the average of these scores gives the final $\text{NMAE}_{\text{mm}}$ score of the model. In order to balance the variety of testing conditions with the computational time per error metric calculation, 10 was selected as the number of initial conditions used in the final $\text{NMAE}_{\text{mm}}$ score.

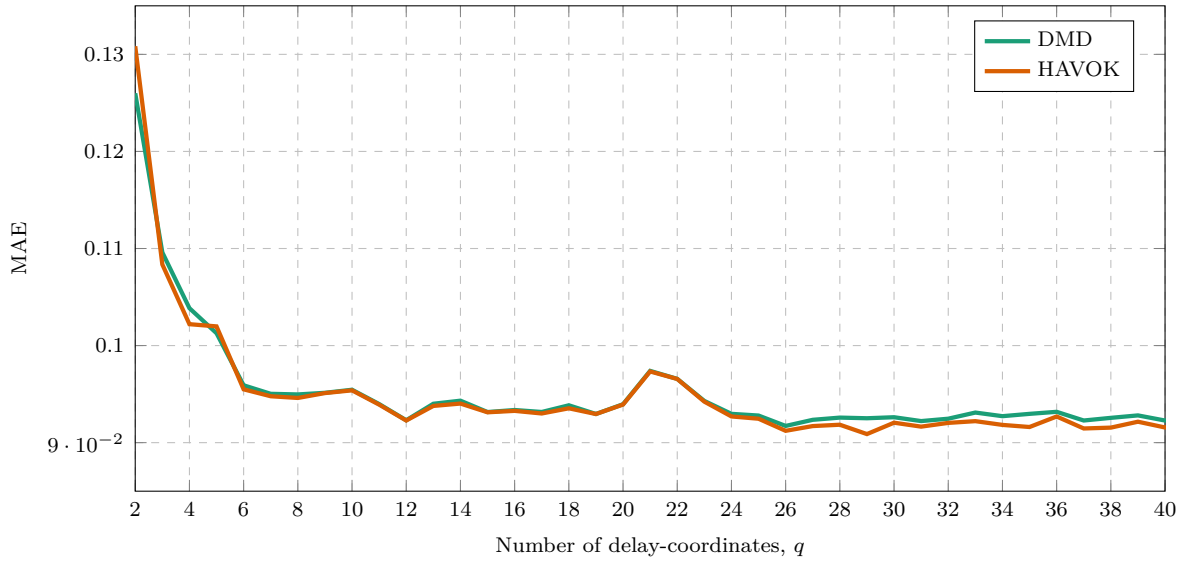
### 4.6.3. Hyperparameters

As discussed in Section 4.4 and 4.5 DMDc and HAVOK 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,

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

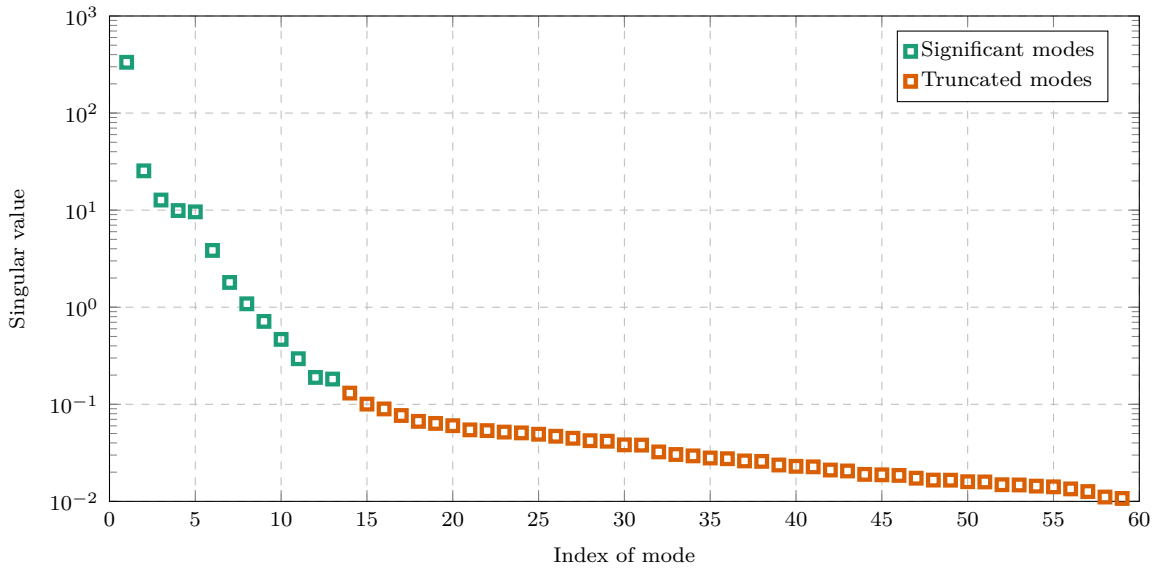
.




**Figuur 4.3:** DMD and HAVOK predictions error for different lengths of noisy training data ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$, $T_{train} = 60\,\text{s}$.)

Figure 4.3 shows the prediction error of DMDc and HAVOK 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 HAVOK. 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 [17]. 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 [18]. These models are less prone to overfitting and also lead to lower computational complexity in the MPC optimisation problem.




**Figuur 4.4:** Significant and truncated singular values of a HAVOK model produced from noisy data ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$, $T_{train} = 60\,\text{s}$.)

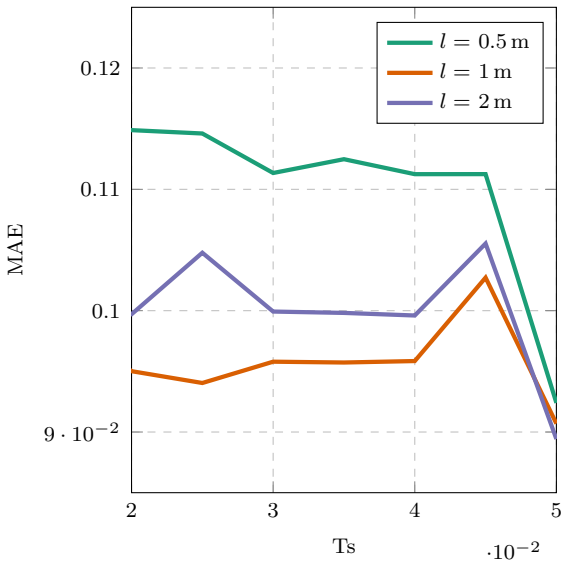
Figure 4.4 plots the singular values of the SVD from a HAVOK model in a log scale. The singular values of the SVD can be loosely interpreted as a measure of significance of the corresponding POD mode to the plant dynamics [19]. That is, modes with higher singular values contain more relevant information about the plant dynamics than modes with lower singular values. The $p$ number of significant singular values, which correspond to the POD modes used to reconstruct the observed dynamics, are specifically shown in the plot. The truncated singular values are also shown, which correspond to the discarded modes.

The Pareto 'elbow' is also visible in this plot where there is noticeable change in gradient roughly at the split between significant and truncated values. This change in gradient shows that after $p$ modes, there is a significant drop in information contributed per remaining mode. Note that the number $p$ was selected from a hyperparameter search using an error metric which did not consider the singular values. This seems to confirm the notion that the Pareto optimal solution is often the most accurate representation of the actual dynamics [19].

### 4.6.4. Sample time

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

The MPC acts in the velocity loop and commands an acceleration setpoint. The default PID velocity controller runs at 50 Hz which corresponds to $T_s = 0.02\,\text{s}$. Due to the computational complexity of an MPC, the optimiser will struggle to run at 50 Hz on a companion computer on a quadrotor. However, the controller needs to run as fast as possible to have significant time-scale separation from the quadrotor 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.




**Figuur 4.5:** DMD prediction error using different cable lengths with a range of different sample times of noisy training data ($m = 0.2\,\text{kg}$)

Figure 4.5 shows the prediction error of different DMD 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 4.5 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 $T_s > 0.045\,\text{s}$. 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 quadrotor. $T_s = 0.03\,\text{s}$ is selected as the sample time for system identification because it provides a good balance between being fast enough for the quadrotor dynamics and slow enough for a practical MPC implementation.

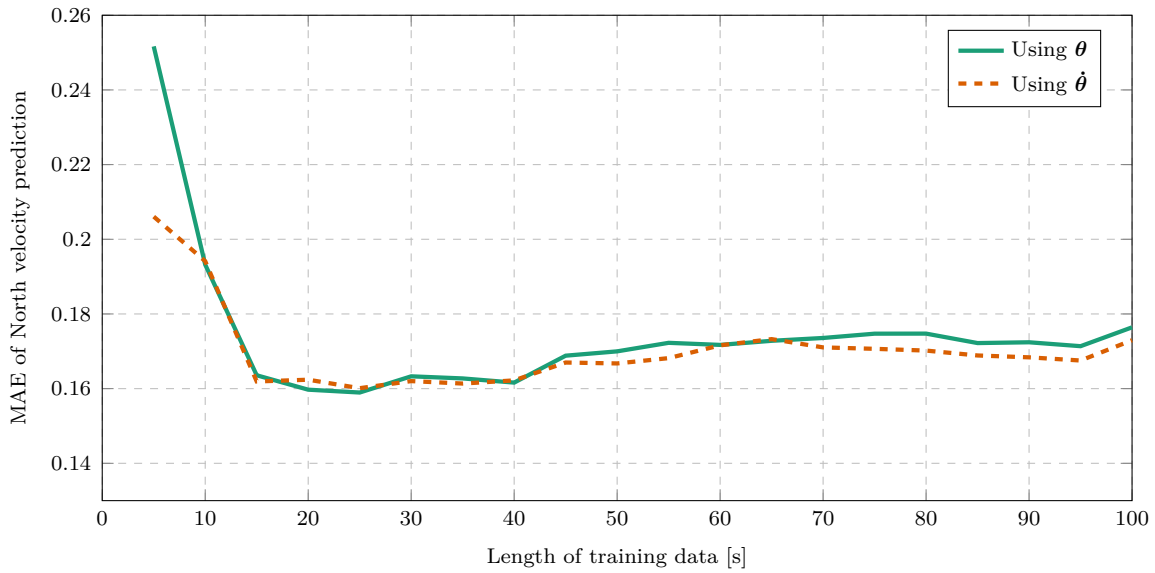
### 4.6.5. Choice of payload variable in the state vector

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.




**Figuur 4.6:** Prediction MAE for HAVOK models using either angle or angular rate measurements ($m = 0.2\,\text{kg}$, $l = 1\,\text{m}$, $T_s = 0.03\,\text{s}$).

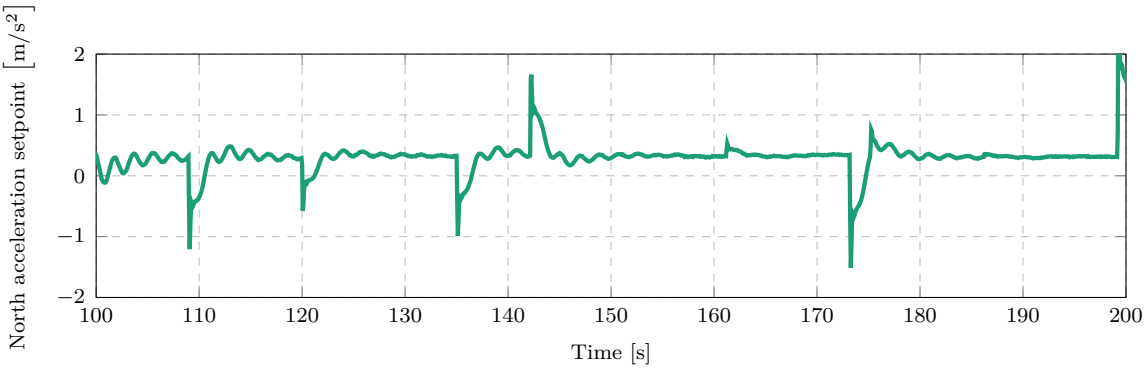
Figure 4.6 shows the prediction error of HAVOK models using $\dot{\theta}$ or $\theta$ for a range training data lengths. Only for very short lengths of training data, do models using $\dot{\theta}$ outperform those using $\theta$. For longer lengths of training data there is a negligible difference 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 quadrotor.

### 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 quadrotor the 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 ADC which measure the payload angle on the quadrotor 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 quadrotor 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.

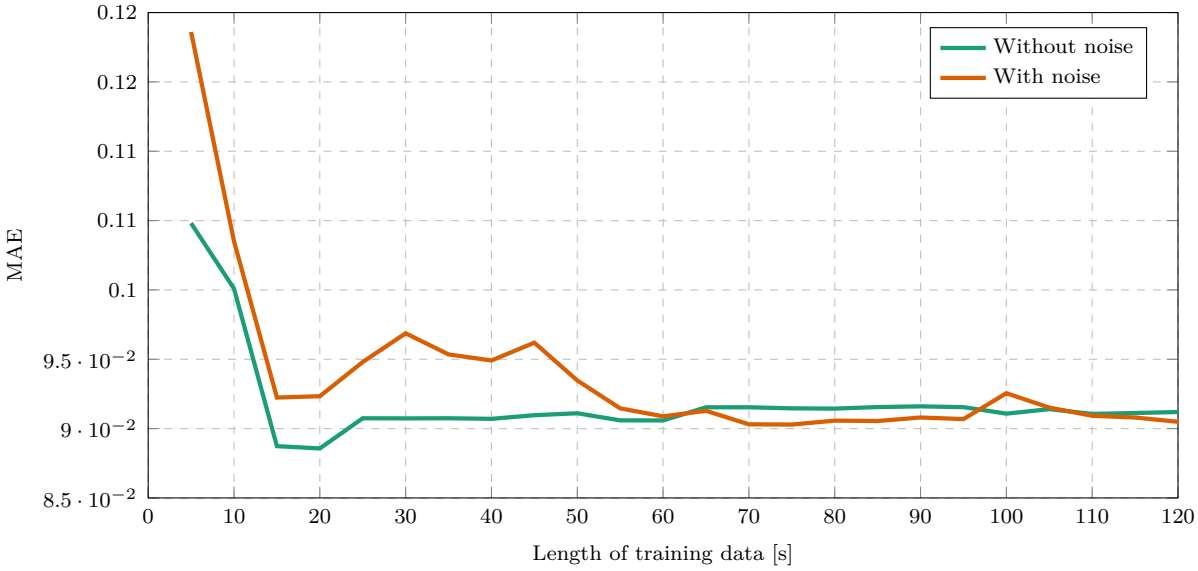



**Figuur 4.7:** Accelleration setpoint training data from random velocity step inputs ($m = 0.2\,\text{kg}$, $l = 1\,\text{m}$)

The input signal also needs to be smoothed to remove high frequency noise from the logged signal. The quadratic regression smoother does not fit the shape of the input data well because of the sharp, non-differentiable edges in the acceleration setpoint signal. Therefore a Gaussian-weighted moving average smoother from MATLAB® is used for the input signal.

Figure 4.7 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

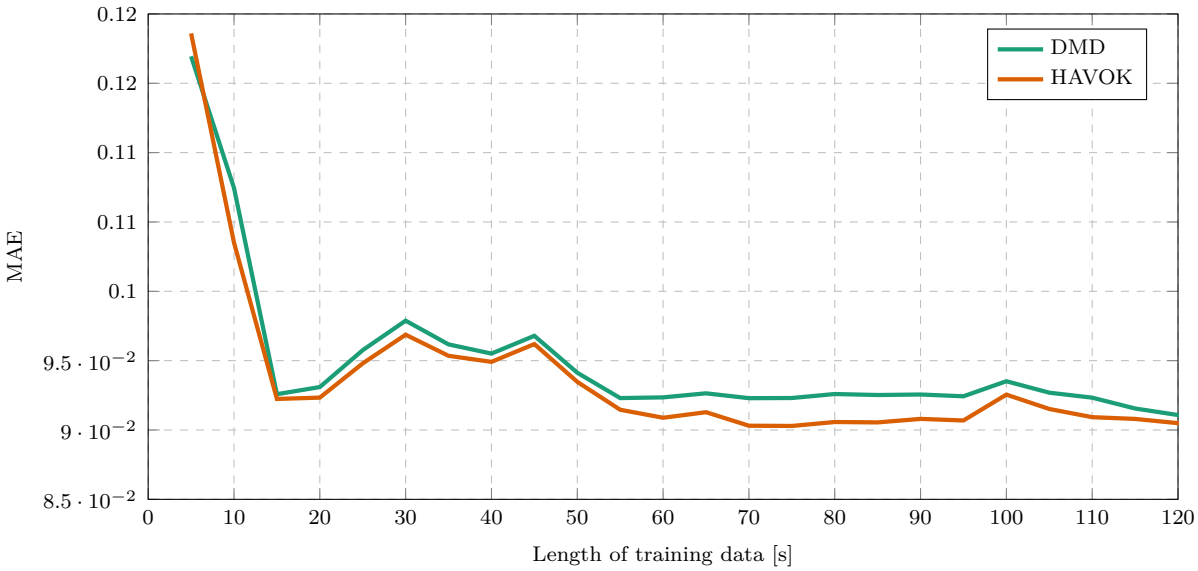
a constant offset. The is due to a measurement offset in the IMU which causes a offset in the orientation vector and therefore affects the control signals. The setpoint mean is calculated from the training data and subtracted from the signal to correct for the offset. This results in a input signal with a zero mean. The calculated mean is reapplied to the MPC control signal during implementation to readjust for the required offset.




**Figuur 4.8:** HAVOK prediction error for different lengths of training data with and without noise ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).

Figure 4.8 compares the prediction errors of HAVOK models generated from data with or without noise. The plot shows that when using short lengths of training data, the prediction errors are smaller for model generated with noiseless signals. However, it appears that the prediction errors are almost equal with longer lengths of training data. This is because with a short length of data, the signal variation or energy contributed of the noise is a significant part of the data and has a strong influence on the model. However, with longer lengths of data, the variation caused by the actual plant dynamics dominates the low energy contribution of the measurement noise. Hence, the noise has a smaller influence on the model. It also appears that at long training data lengths noise has a negligible effect on the prediction error of the resulting models.

Figure 4.9 compares the performance of HAVOK and DMDc model when using noisy data. The prediction error curves of the two methods are very simlar, with HAVOK producing slightly lower prediction errors than DMDc. However, this difference in error may be so small that it has a negligible effect on control.

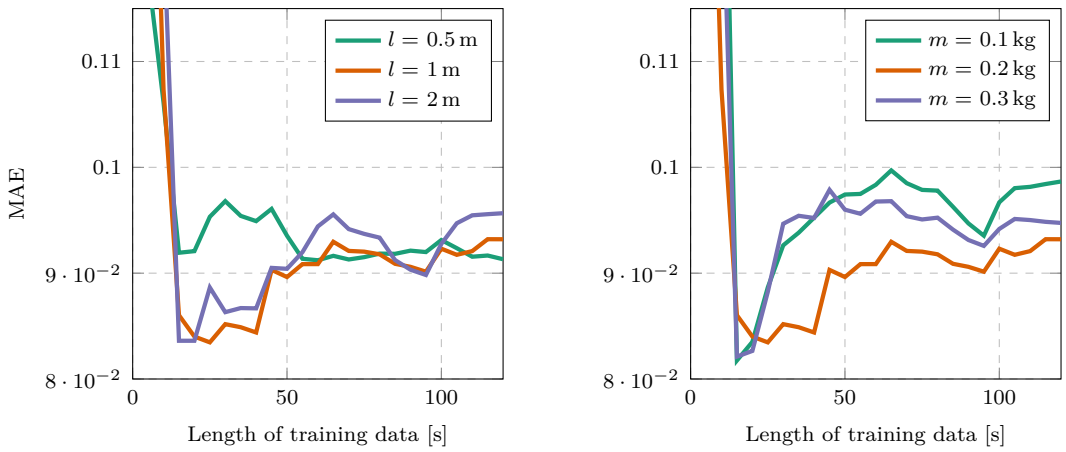



**Figuur 4.9:** DMD and HAVOK prediction error for different lengths of noisy training data ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).

### 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 quadrotor 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 4.10 shows the prediction error of HAVOK 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 HAVOK results.

From Figure 4.10a it seems that there is not a great difference in prediction error for different cable length setups. From Figure 4.10b 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




**(a)** $m_p = 0.2\,\text{kg}$, with varying $l$ **(b)** $l = 1\,\text{m}$, with varying $m_p$

**Figuur 4.10:** HAVOK prediction error for different system parameters

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 **??**. 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, $N_{train}$ [20]. In this critical region at the transition between over-parameterized and under-parameterized models, the prediction error initially decreases, then increases to a peak whereafter it decreases again [20].

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:

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

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 $l = 0.5\,\text{m}$ in Figure 4.10a 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 quadrotor can fly with a updated controller. Less training data also corresponds

to lower memory usage on the quadrotor 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

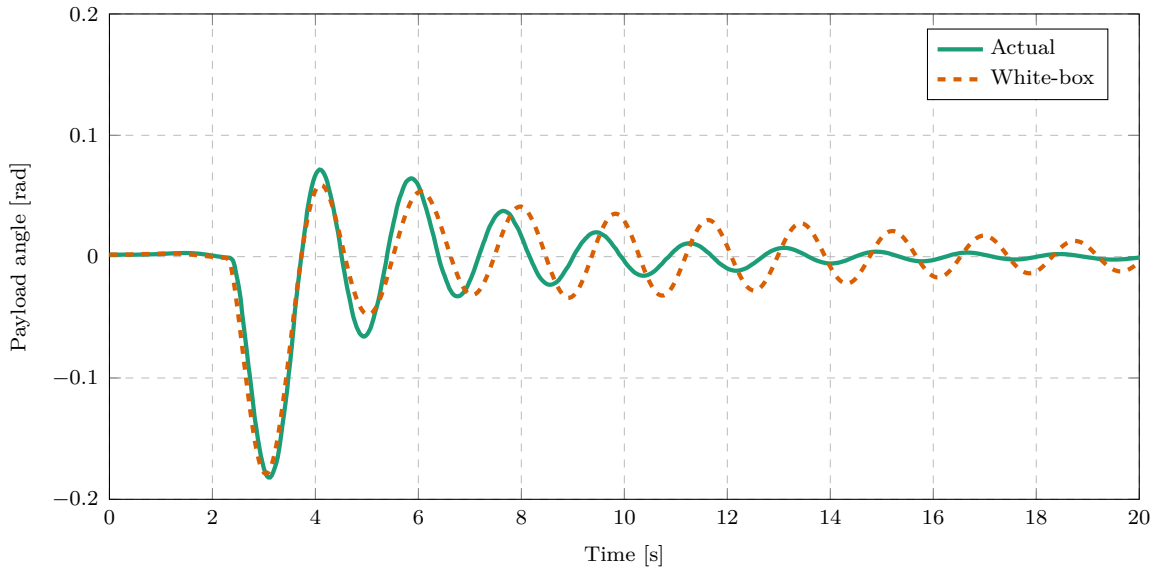
In Section 4.3 it was shown that the white-box system identification method perform well for the suspended payload use case. It was also shown in [2] and [1] that this method can be used in conjunction with LQR control to minimise swing angles of an unknown payload. However, many payloads do not satisfy 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 single pendulum.

Figure ?? shows a practical double pendulum use case. This payload is better represented in 2D by the double pendulum modelled in Figure ??, than the model defined in Section 4.2.

insert picture a practical quad with long payload insert diagram of double pend

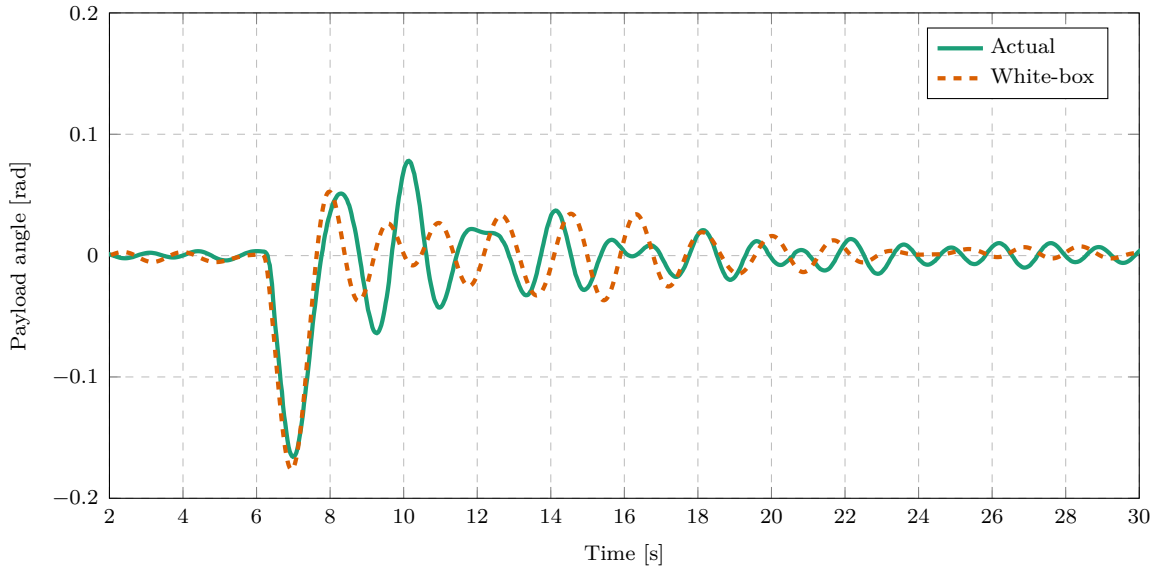
Figure 4.11 shows the k-step-ahead prediction of a white-box models for a single pendulum simulation. The exact $m_p$ is used for the model, but the cable length is estimated from a FFT of the payload swing angle as described in Section 4.3.3. It is clear that the prediction is accurate for the first few oscillations, but the slight difference in frequency accumulates over time causing an increasingly large offset. However, the general shape of the dynamics is still captured by the model. Also note that the prediction oscillations are damped linearly but the actual oscillations experience non-linear damping. This difference is an approximation error because the non-linear plant is modelled with a linear model.

Figure 4.12 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 4.11 are regular compared to double pendulum oscillations in Figure 4.12 which are noticeably irregular. The a priori model expects regular, single frequency oscillations. This can be seen in Figure 4.12 where the predicted peaks are



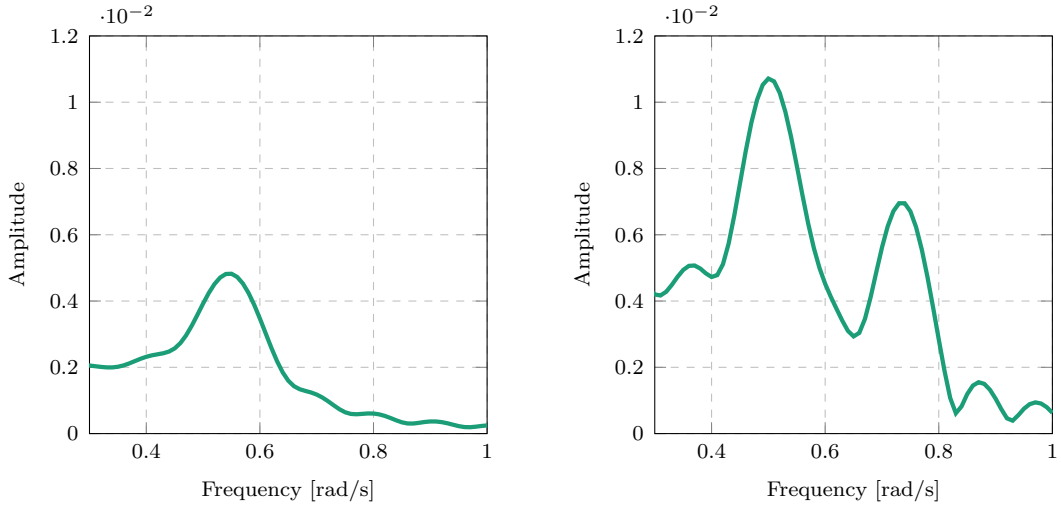

**Figuur 4.11:** White-box model predictions of a single pendulum for a North velocity step input ($l = 1\,\text{m}$, $T_{train}$ = various??.)

equidistant. However, the actual dynamics have a superposition of two frequencies due to the double pendulum payload.




**Figuur 4.12:** White-box model predictions of a double pendulum for a North velocity step input ($m = 0.3\,\text{kg}$, $l = 1\,\text{m}$)

The FFT amplitude spectrum of the single pendulum is shown in Figure 4.13b. This plot shows a single peak which corresponds to the natural frequency of the suspended payload. Figure 4.13a 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




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

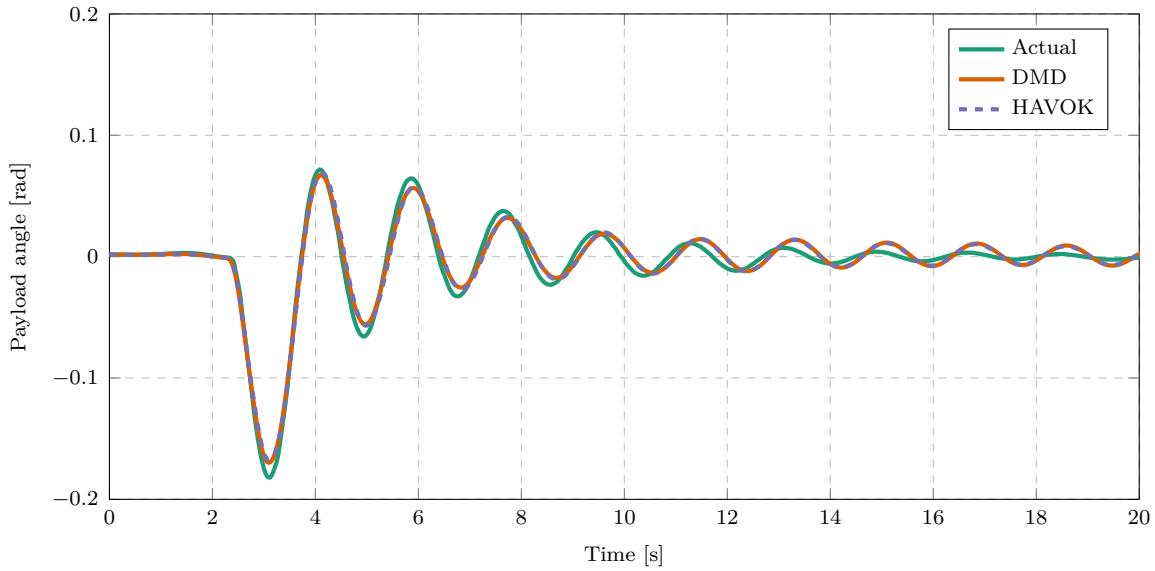
**Figuur 4.13:** The single-sided amplitude spectrum of the swing angle FFT ($m = 0.3\,\text{kg}$, $l = 1\,\text{m}$).

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 4.14 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 4.11. 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.

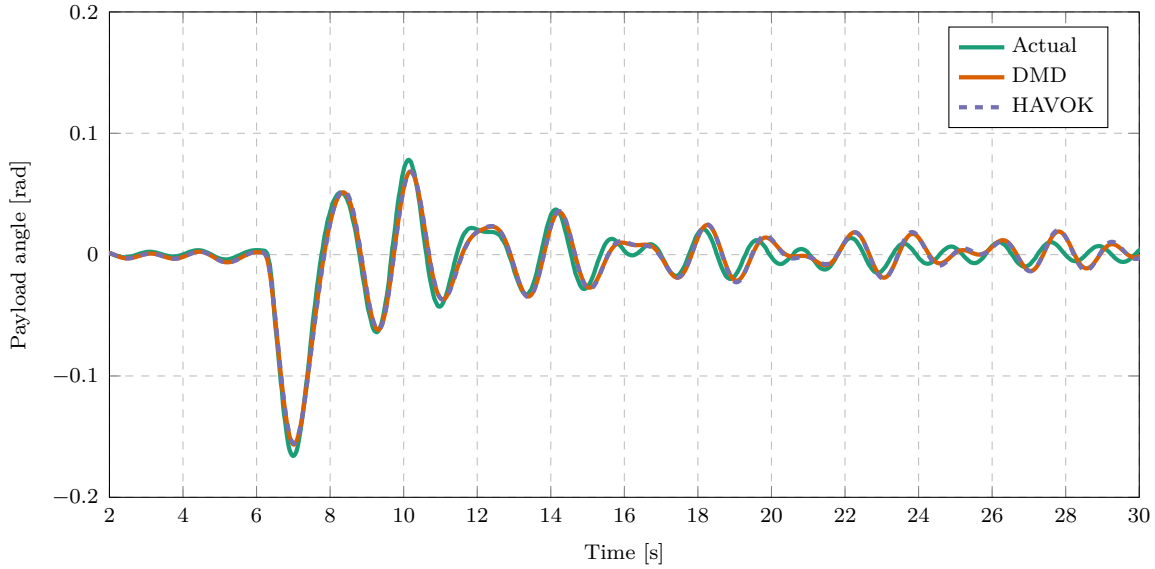
The damping seen in these plots is a complicated effect which depends on the payload connection, the aerodynamic drag, and the controller gains. An advantage of data-driven system identification techniques is that the effect of damping is inherently included in the estimated model without specifically 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 4.15 shows the prediction of the same data-driven methods but for a double 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.




**Figuur 4.14:** Data-driven model predictions of a single pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).

As expected, the prediction accuracy is also much better than the white-box models for both the single and double pendulum simulations.

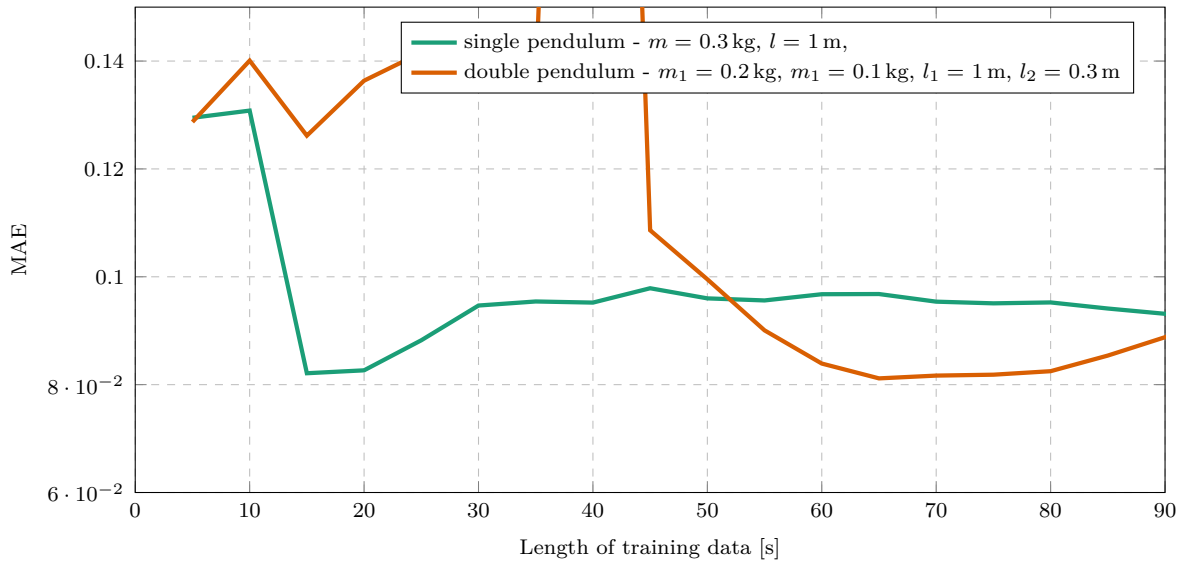



**Figuur 4.15:** Data-driven model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$)

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. The DMDc and HAVOK models can still approximate the dynamics without measuring this state variable because of the delay embedding and Taken's Embedding Theorem [11].

'Takens' Embedding Theorem asserts that when the states of a hidden dynamical system are confined to a low-dimensional attractor, complete information about the states can be preserved in the observed time-series output through the delay coordinate map.

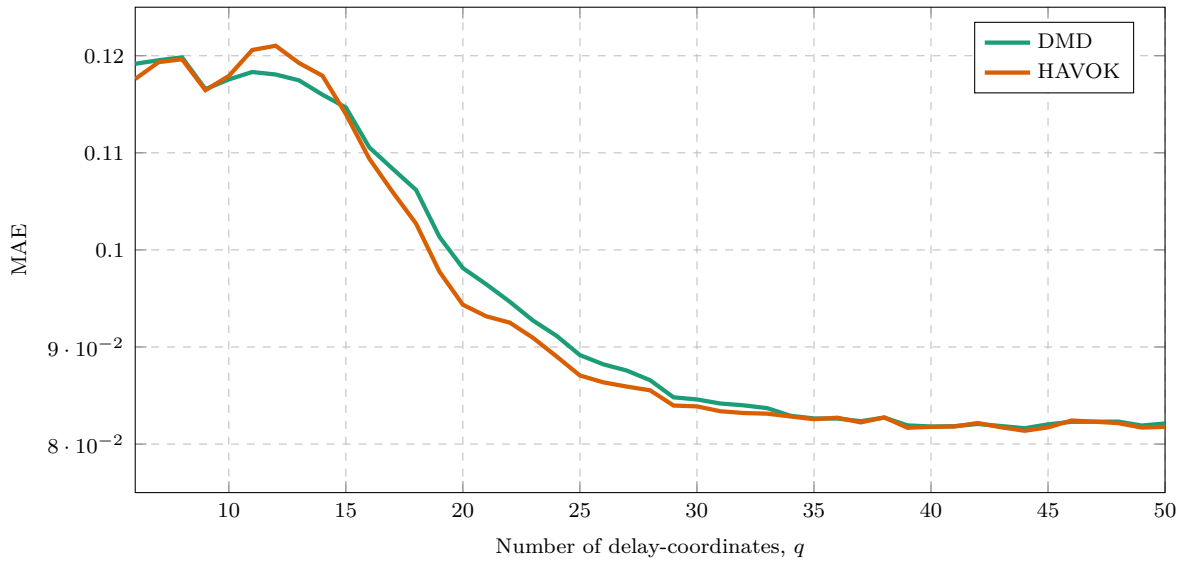
However, the conditions for the theorem to hold ignore the effects of noise and time-series analysis' [21]




**Figuur 4.16:** HAVOK prediction error using different cable lengths with a range of different lengths of noisy training data ($l = 1\,\mathrm{m}$, $T_{train}$ = various??.)

Not how much more delays are required

Takens theorum. Add DMD to plot




**Figuur 4.17:** DMD and HAVOK predictions error of double pendulum for different numbers of delay-coordinates ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 1\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.3\,\mathrm{m}$ $T_{train} = 70\,\mathrm{s}$.)

The double pendulum is one example of a payload case that would require a redesign of the control architecture. Other dynamic payloads that are difficult to model with a white-box method are containers holding a fluid. Examples:

Another payload case that will cause inaccuracies in estimated white-box model is if a payload is attached rigidly to the quadrotor while also carrying a suspended payload. The payload mass estimation is based on the assumption that the quadrotor mass is known. However if a mass is rigidly attached to the vehicle, the effective quadrotor mass is changed and the RLS payload mass estimation is no longer accurate.

To determine whether the white-box or data-driven methods perform better with these payloads, the performance of these methods need to be compared to each other. However, comparing these methods is a non-trivial task because they result in different types of models which are used in different types of controllers. The white-box system identification technique generates a continuous-time, state space model which is used in an LQR controller. This is in the form:

$$\dot{x} = Ax + Bu \tag{4.25}$$

For this model, the accuracy of the time-derivative estimate at the current time-step is important for control. Hence, a k-step-ahead prediction error is not a good measure of performance for this model. In contrast, the data-driven techniques result in a discrete state space model,

$$x_{k+1} = Ax_k + Bu_k, \tag{4.26}$$

which is used in a MPC. The k-step-ahead prediction accuracy of this model is important for the performance of the MPC. Therefore comparing the performance of the controllers which use these models is a better approach than comparing the model accuracy. This will be investigated in Chapter 5.

## 4.7. Conclusion

DMD and HAVOK work very similarly with single pend. HAVOK has slightly better prediction accuracy, but this small difference has neglible effect on control. It is difficult to compare the white-box to the black-box models because the real effect will only be seen during control. However it is clear that the accuracy of the white-box model degrades significantly when a payload that causes double pendulum dynamics.

The major advantage of the data-driven approach which was demonstrated in this section is that the method was applied to different dynamical systems

For each of these payload cases, a different parameter estimation based techniques would needs to be designed for effective control. This is undesirable for practical drone deliveries, especially when the type of paylaod is not known well in advance or changes regularly. A

data-driven technique provides a more general solution since it accommodates a larger range of payload types and does not require a prioir modelling information.

Since the a priori white-box model is based on a single pendulum model, the dynamics described by the model are significantly different from the actual dynamics. This will have a detrimental effect on the control performance of a controller based on such a model, since the controller will be designed for different plant than what it is controlling actually controlling.

# Hoofstuk 5

# Control systems

## 5.1. Cascaded PID with payload

## 5.2. LQR

## 5.3. MPC

# Hoofstuk 6

# System overview

# Hoofstuk 7

# Implementation and results

# Hoofstuk 8

# Summary and Conclusion

# Bibliografie


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

[2] 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.

[3] 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

[4] 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.

[5] 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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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

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

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

[20] 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.

[21] H. L. Yap, A. Eftekhari, M. B. Wakin, and C. J. Rozell, "A first analysis of the stability of Takens' embedding," in *2014 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2014*. Institute of Electrical and Electronics Engineers Inc., feb 2014, pp. 404–408.

# Bylae A

# PID gains

This is an appendix about PID gains used for Honeybee.

# Bylae B

# Random appendix

This is another appendix.