Inhoudsopgawe

Hoofstuk 1

Introduction

The last few years have seen great advances in speech recognition. Much of this progress is due to the resurgence of neural networks; most speech systems now rely on deep neural networks (DNNs) with millions of parameters [**?**, **?**]. However, as the complexity of these models has grown, so has their reliance on labelled training data. Currently, system development requires large corpora of transcribed speech audio data, texts for language modelling, and pronunciation dictionaries. Despite speech applications becoming available in more languages, it is hard to imagine that resource collection at the required scale would be possible for all 7000 languages spoken in the world today.

I really like apples.

1.1. Section heading

Hoofstuk 2

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

2.1. Coordinate frames

2.2. States

2.3. Forces and moments

2.4. Lagrangian mechanics

2.5. Linearised model

2.6. Discretised model

2.7. Model verification

2.8. Dynamic payloads

water looks like double payload

Hoofstuk 3

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.

3.1. White-box and black-box techniques

3.1.1. 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.

3.1.2. 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 2.5, 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.

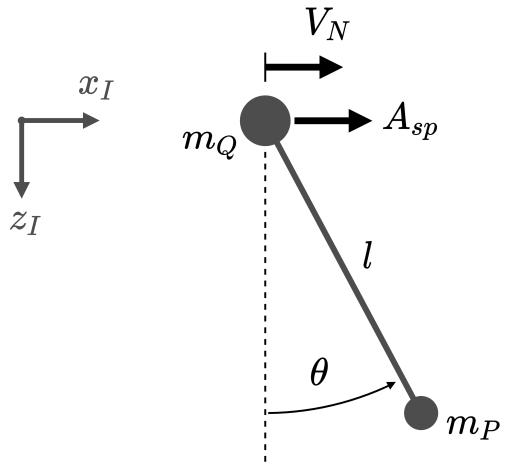
3.2. Plant considered for system identification

Derived model Figure 3.1 shows the plant considered for system identification. In Chapter 2 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,

$$\boldsymbol{x} = \begin{bmatrix} V_N & \dot{\theta} \end{bmatrix}^T, \tag{3.1}$$

and the input vector,

$$\boldsymbol{u} = \begin{bmatrix} A_{N,sp} \end{bmatrix}, \tag{3.2}$$



Figuur 3.1: Floating pendulum model considered for system identification for a North velocity controller

3.3. Parameter estimation

3.3.1. 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 2.5.

3.3.2. Payload mass estimation

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

3.3.3. 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:

$$\omega_n = \sqrt{\frac{g}{l} \cdot \frac{m_q + m_p}{m_q}} \tag{3.3}$$

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 ω_n has been estimated, l can now be determined from 3.3. In this case the estimated length is ??, compared to the actual length of ??.

Frequency resolution ?? error for different lengths??

3.4. 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:

$$\boldsymbol{d}_k = \begin{bmatrix} \boldsymbol{x}_{k-1} & \boldsymbol{x}_{k-2} & \cdots & \boldsymbol{x}_{k-q} \end{bmatrix}^T, \tag{3.4}$$

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

The discrete state-space model is therefore defined as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{A}_d\boldsymbol{d}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{3.5}$$

$\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$ 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:

$$\begin{aligned}
\boldsymbol{X'} &= \begin{bmatrix} \boldsymbol{x}_{q+2} & \boldsymbol{x}_{q+3} & \boldsymbol{x}_{q+4} & \cdots & \boldsymbol{x}_{w+q+1} \end{bmatrix}, \\
\boldsymbol{X} &= \begin{bmatrix} \boldsymbol{x}_{q+1} & \boldsymbol{x}_{q+2} & \boldsymbol{x}_{q+3} & \cdots & \boldsymbol{x}_{w+q} \end{bmatrix}, \\
\boldsymbol{X}_d &= \begin{bmatrix} \boldsymbol{x}_q & \boldsymbol{x}_{q+1} & \boldsymbol{x}_{q+2} & \cdots & \boldsymbol{x}_{w+q-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_2 & \boldsymbol{x}_3 & \boldsymbol{x}_4 & \cdots & \boldsymbol{x}_{w+1} \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 & \cdots & \boldsymbol{x}_w \end{bmatrix}, \\
\boldsymbol{\Upsilon} &= \begin{bmatrix} \boldsymbol{u}_q & \boldsymbol{u}_{q+1} & \boldsymbol{u}_{q+2} & \cdots & \boldsymbol{u}_{w+q-1} \end{bmatrix},
\end{aligned} \tag{3.6}$$

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 (3.5) can be combined with the matrices in Equation (3.6) to produce:

$$\boldsymbol{X'} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{A}_d\boldsymbol{X}_d + \boldsymbol{B}\boldsymbol{\Upsilon}. \tag{3.7}$$

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, (3.5) is manipulated into the form,

$$\boldsymbol{X'} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{A}_d & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}_d \\ \boldsymbol{\Upsilon} \end{bmatrix} = \boldsymbol{G}\boldsymbol{\Omega}, \tag{3.8}$$

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:

$$\boldsymbol{\Omega} \approx \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^T, \tag{3.9}$$

where $\tilde{}$ represents rank-p truncation. maybe insert colour pictures showing matrices

By combining (3.9) with the over-constrained equality in (3.8), the least-squared solution, $\boldsymbol{G}$, can be found with:

$$\boldsymbol{G} \approx \boldsymbol{X'}\tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{U}}. \tag{3.10}$$

By reversing 3.8, $\boldsymbol{G}$ 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.

3.5. 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:

$$\boldsymbol{a}_{k+1} = \tilde{\boldsymbol{A}}\boldsymbol{a}_k + \tilde{\boldsymbol{B}}\boldsymbol{u}_k, \tag{3.11}$$

where $\boldsymbol{a}_k$ is the state vector previously defined in Section 3.4, $\tilde{\boldsymbol{A}} \in \mathbb{R}^{(q\cdot n_x)\times(q\cdot n_x)}$ 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 [**?**], 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:

$$\boldsymbol{\Pi} = \begin{bmatrix} \boldsymbol{a}_q & \boldsymbol{a}_{q+1} & \boldsymbol{a}_{q+2} & \cdots & \boldsymbol{a}_{w+q-1} \\ \boldsymbol{u}_q & \boldsymbol{u}_{q+1} & \boldsymbol{u}_{q+2} & \cdots & \boldsymbol{u}_{w+q-1} \end{bmatrix}, \tag{3.12}$$

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

$$\mathbf{\Pi} \approx \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^T, \tag{3.13}$$

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 [**?**]. 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 [**?**]. Consider the following discrete, state-space formulation:

$$\boldsymbol{v}_{k+1} = \mathbf{\Lambda}\boldsymbol{v}_k. \tag{3.14}$$

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

$$\begin{aligned} \boldsymbol{V}_1 &= \begin{bmatrix} \boldsymbol{v}_q & \boldsymbol{v}_{q+1} & \ldots & \boldsymbol{v}_{w-1} \end{bmatrix}, \\ \boldsymbol{V}_2 &= \begin{bmatrix} \boldsymbol{v}_{q+1} & \boldsymbol{v}_{q+2} & \ldots & \boldsymbol{v}_w \end{bmatrix}, \end{aligned} \tag{3.15}$$

where $\boldsymbol{V}_2$ is $\boldsymbol{V}_1$ advanced a single step forward in time. The matrices from Equation (3.15) are now combined with Equation (3.14) and the best fit $\mathbf{\Lambda}$ is determined with the Moore-Penrose pseudoinverse:

$$\boldsymbol{V}_2 = \mathbf{\Lambda}\boldsymbol{V}_1 \qquad \Rightarrow \qquad \mathbf{\Lambda} \approx \boldsymbol{V}_1\boldsymbol{V}_1^{\dagger} \tag{3.16}$$

It can be shown from Equation (3.13) that Equation (3.14) is transformed from the eigen-time-delay coordinate system to the original coordinate system as the following:

$$\begin{bmatrix} \boldsymbol{a}_{k+1} \\ \boldsymbol{u}_{k+1} \end{bmatrix} = (\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}})\mathbf{\Lambda}(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}})^{\dagger} \begin{bmatrix} \boldsymbol{a}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{3.17}$$

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:

$$\begin{bmatrix} \boldsymbol{a}_{k+1} \\ \boldsymbol{u}_{k+1} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{A}} & \tilde{\boldsymbol{B}} \\ \textit{(discarded)} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{3.18}$$

Note that the matrix entries in (3.18) 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.

3.6. 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.

3.6.1. Methodology

Simulation environment A SITL implementation of PX4 [10] using the Gazebo simulator [11] 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 2. 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.

Method overview The procedure used to generate data for 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

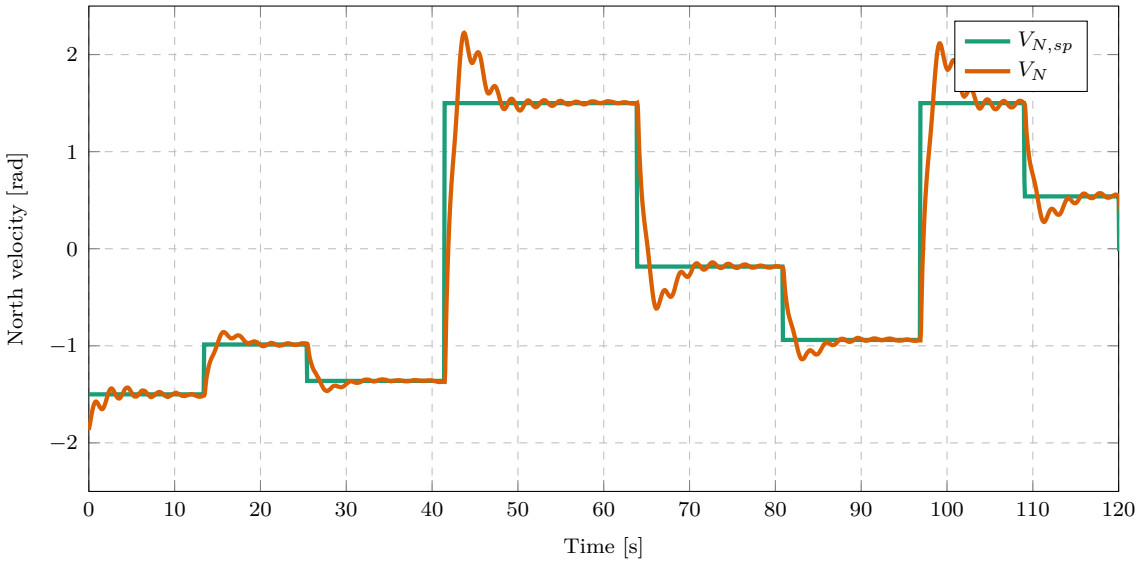
5. Split data into separate training and testing periods

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'.

Steps and intervals 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 3.1 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 3.1: Input data ranges.

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



Figuur 3.2: Example of training data with random velocity step inputs ($m = 0.2\,\text{kg}$, $l = 1\,\text{m}$)

Figure 3.2 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.

Testing data Cross-validate

Error metric Each state error signal is scaled by the reciprocal of the maximum value of that state variable in the training data. This is to ensure that a scale difference in the variable types create a bias in the error metric. For example, the quadrotor velocity reaches values of 3 m/s but the payload swing angle has a maximum of only 0.526 rad. The velocity prediction error is therefore inherently larger than the payload angle prediction error and will bias the error metric towards favouring models with good velocity predictions. The proposed scaled error metric ensures that the MAE of each state variable can be compared to each other. It also provides an error metric that is better and unbiased representative of the model prediction performance across all state variables.

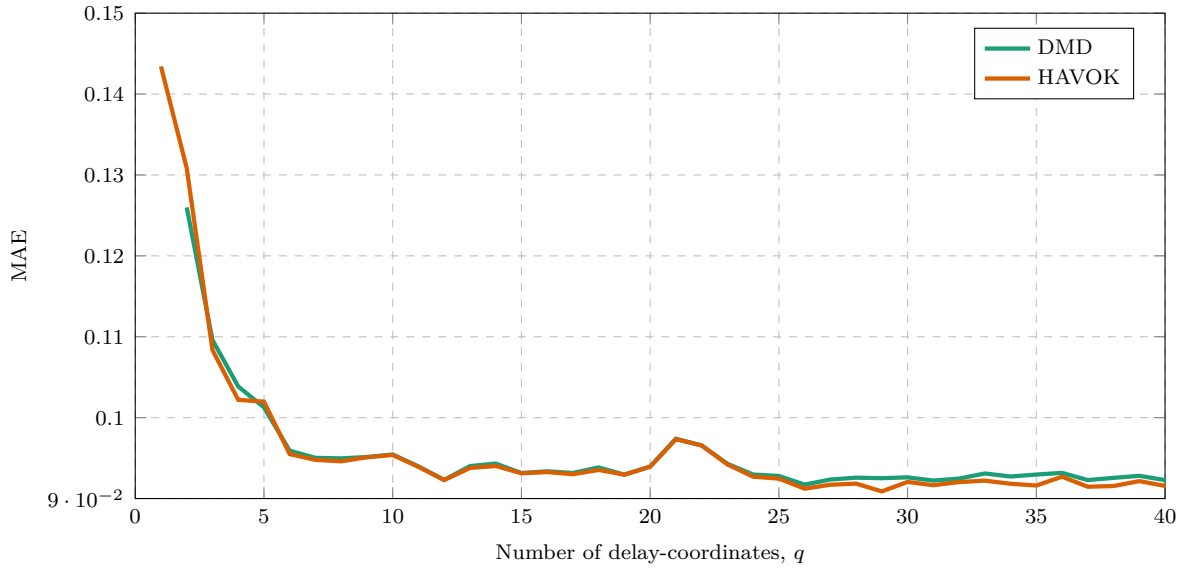
Add Information Criteria ?? AIC Brunton Kutz??

3.6.2. Hyperparameters

As discussed in Section 3.4 and 3.5 DMDc and HAVOK are dependent on two hyperparameters: the number of delay-coordinates, q, and the SVD truncation rank, p.

Parsimony Pareto front cite Data-Driven book

The more terms better chance to overfit, lower generalisation



Figuur 3.3: DMD and HAVOK predictions error for different lengths of noisy training data ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$, $T_{train} = 60\,\text{s}$.)

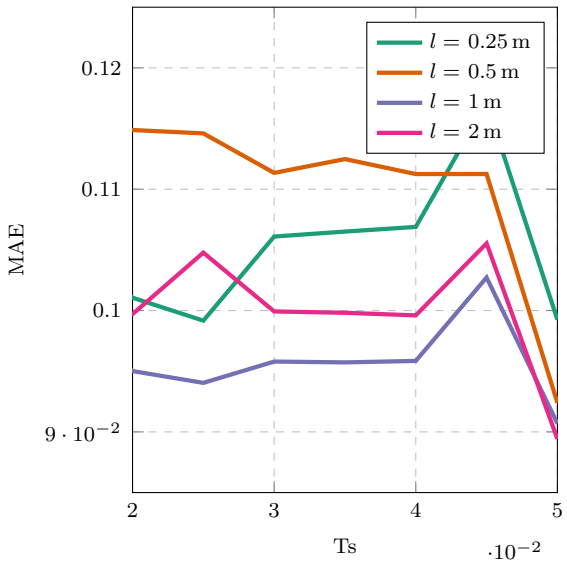
Fixed size of data Fixed sample time Fixed pendulum params Talk about the ”front”Also

about singular values For each of the experiments shown in this chapter, a hyperparameters selected tuned to produced

3.6.3. Sample time

The sample time, T_s, used for system identification sets 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 quadrotor companion computer. This is because the



Figuur 3.4: DMD prediction error using different cable lengths with a range of different sample times of noisy training data ($m = 0.2\,\text{kg}$, T_{train} = various??.)

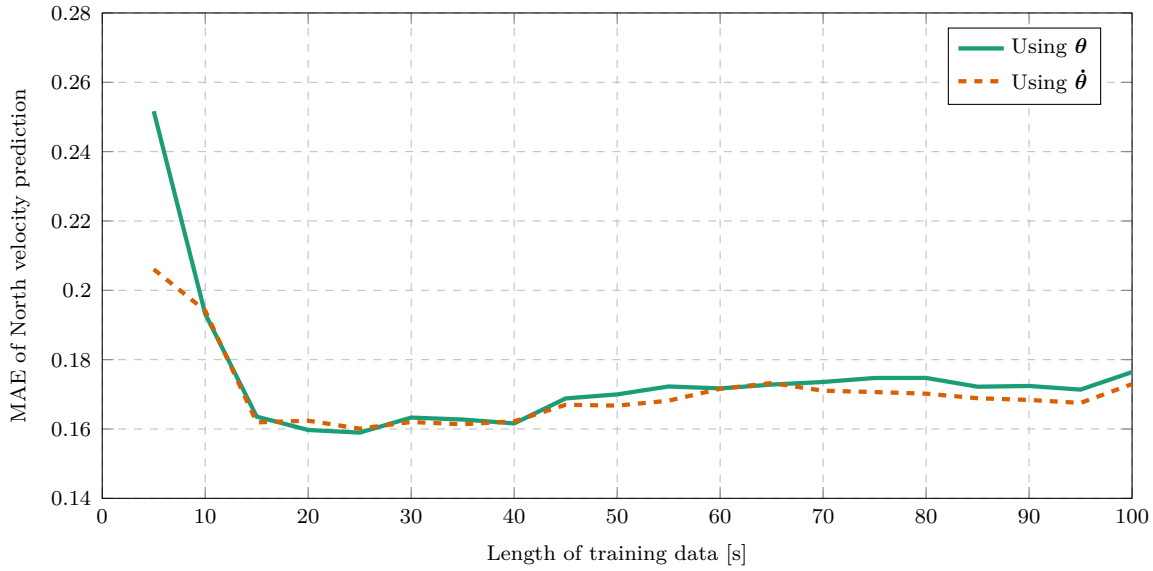
3.6.4. Choice of payload variable in the state vector

As discussed in Section 3.2, the equations of motion of a floating pendulum in continuous-time are dependent on $\dot{\theta}$ and V_N, but are not dependent on θ. Therefore it is expected that $\boldsymbol{x} = \begin{bmatrix} V_N & \dot{\theta} \end{bmatrix}^T$ is 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 like the backward Euler form,

$$\dot{\theta}_k = (\frac{1}{T_s}) \cdot \theta_k - (\frac{1}{T_s}) \cdot \theta_{k-1}. \tag{3.19}$$

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

Based on the floating pendulum equations, it is expected that a model derived with $\dot{\theta}$ data will better approximate the actual dynamics than one using θ. This is because $\dot{\theta}$ contains more direct information about the dynamics compared to θ. A model using θ needs to "learn"numerical differentiation and the effect of $\dot{\theta}$ on the other variables. A model using $\dot{\theta}$ only needs to consider its relationship with other variables.



Figuur 3.5: Prediction MAE for HAVOK models using either angle or angular rate measurements ($m = 0.2\,\text{kg}$, $l = 1\,\text{m}$, $T_s = 0.03\,\text{s}$).

For each length of training data, the hyperparameter combination producing the lowest prediction error was determined and used. From this plot it is clear that models with θ produce more accurate predictions than those with $\dot{\theta}$.

3.6.5. Noise

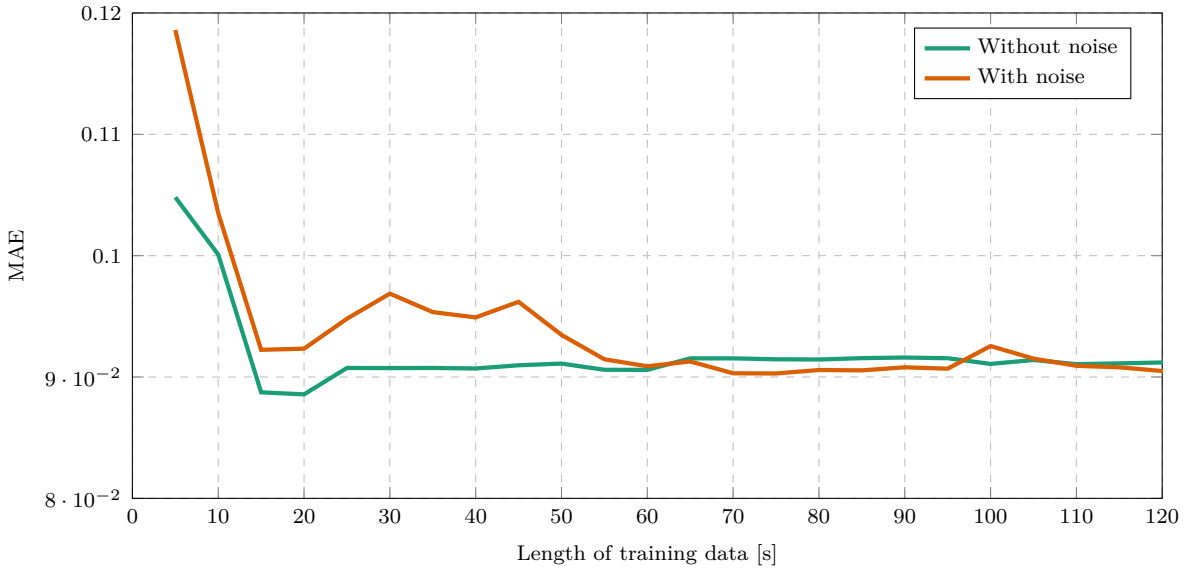
Measurement noise is Find reference for measurement noise definition This is bad for system identification because the output signals no longer represent the actual process hides the actual dynamics of the system under The IMU, barometer, magnetometer and GPS sensors on the practical quadrotor are used for state estimation and all 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.

The potentiometer and ADC which measure the payload angle on the quadrotor alos has quite a lot of measurement noise. However, this signal is not smoothed by an onboard

EKF. Figure **??** shows the noisy payload angle measurement for a practical pendulum test while the quadrotor is held stationary. For models using θ in the state vector instead of $\dot{\theta}$, this noisy signal can be smoothed with matlab smoother. Figure **??** compares the noisy payload angle measurement to the smoothed signal and actual payload angle for a simulated flight. The is applied as band-limited white-noise and the noise power was iteratively adjusted to match that of the practical payload measurements.

However, since there is no direct measurement of $\dot{\theta}$, numerical differentiation is performed on the noisy θ measurement to estimate $\dot{\theta}$. This amplifies the noise and results in inaccurate $\dot{\theta}$ signal. Total variation differentiation is implemented to estimate $\dot{\theta}$ from the noisy measurements more accurately. [] Figure **??** shows

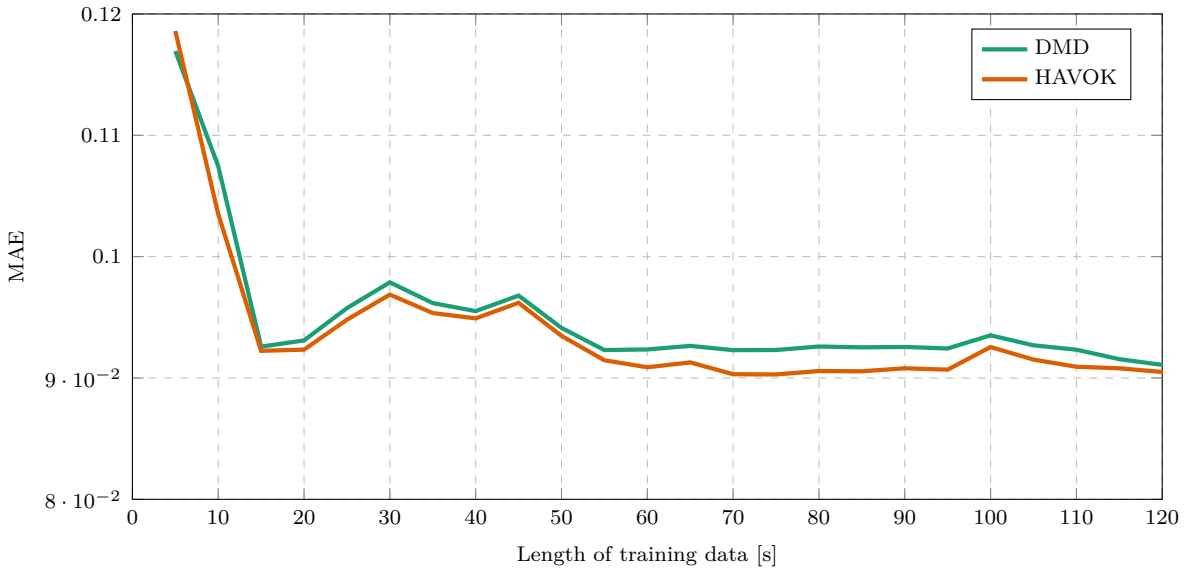
Noise also affects model prediction accuracy and the length of training data required for adequate predictions.



Figuur 3.6: HAVOK prediction error for different lengths of training data with and without noise ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).

HAVOK performs better than DMD. This slight difference in prediciton performance has a negligible effect on control.

Input data needs to be adjusted. What about disturbances



Figuur 3.7: DMD and HAVOK prediction error for different lengths of noisy training data ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).

3.6.6. Size of training data

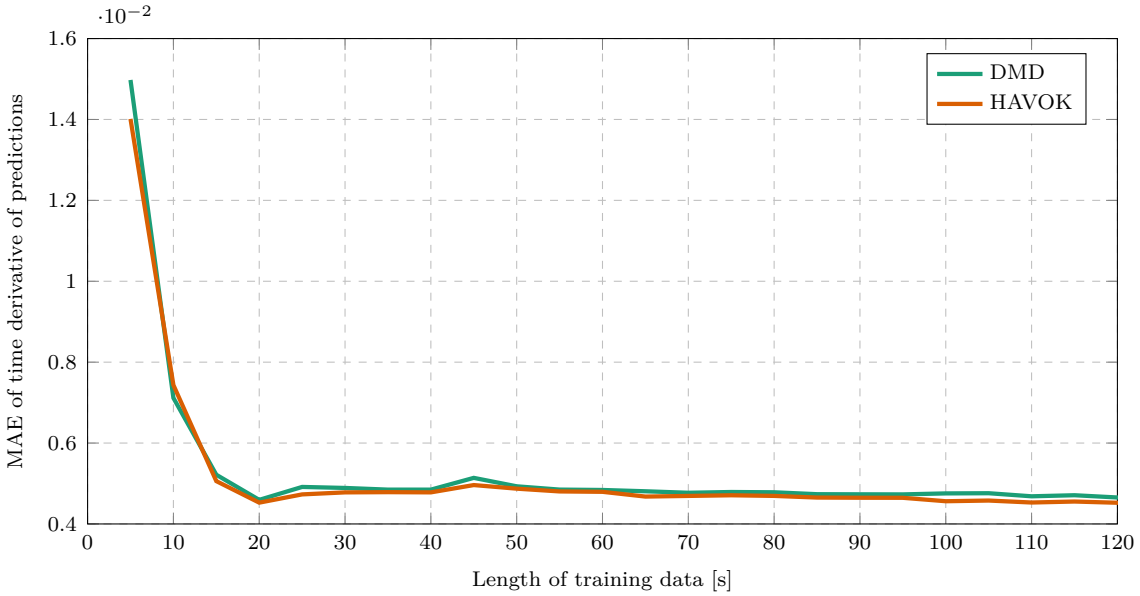
The length of training data used for system identification affects the quality of the model produced. In Figure 3.5 it is clear that prediction error decreases as the amount of training data increases. As more training data is used in the regression problem, the determined model better approximates the actual dynamics because a large range of the dynamics is ”seen”by the algorithm.

Models produced from data lengths as short as 5 s predict the movement of state variables surprisingly well. Note how the general shape of the prediction represents the training data, even though it contains a lot more high frequency oscillations.

Define MAE diff with equation ??.

The models produced from HAVOK appear to produce slightly better prediction errors, however this small difference has a negligible effect on control performance.

the prediction error does not significantly improve with more training data. It practice less training data is desirable because 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 quadrotor hardware. Such a slight improvement in prediction error also has a negligible effect on control performance and is therefore not worth the increased data requirement.



Figuur 3.8: DMD and HAVOK error of time derivative of predictions for different lengths of noisy training data ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).

3.6.7. System parameters

Works across a range of parameters.

The payload acting as a single floating pendulum, as described in Section 3.2, has two system parameters, m_p and l. For the practical quadrotor considered, the payload mass is limited to:

$$0.01 \leq m_p \leq 0.4\,\text{kg}. \tag{3.20}$$

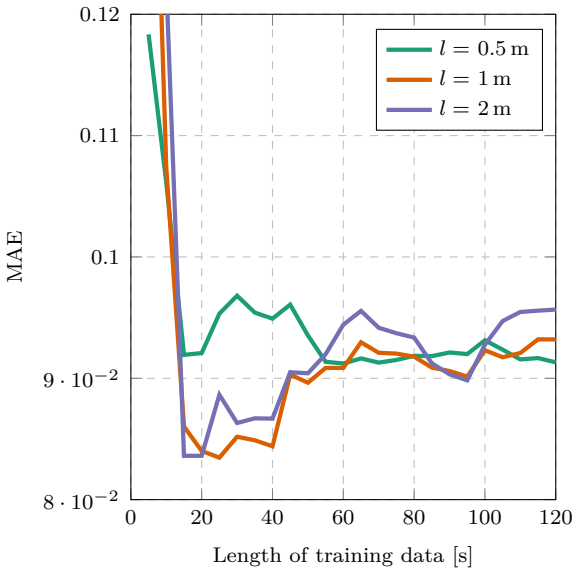
When no external payload is attached, the connection device attached to the end of the cable is $m_p = 0.01\,\text{kg}$. On the other limit, $m_p = 0.4\,\text{kg}$ is determined to be the maximum payload mass the quadrotor can carry safely based on the maximum thrust of the motors.

The cable length is limited to:

$$0.5 \leq l \leq 2\,\text{m}. \tag{3.21}$$

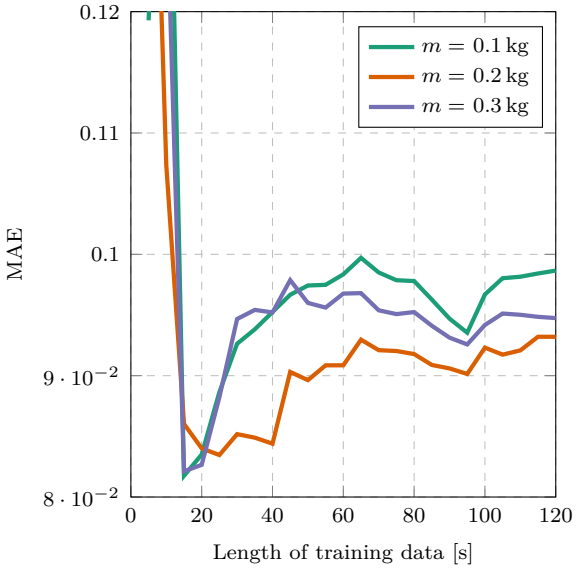
A cable length shorter than 0.5 m is quite impractical and may rather be attached as a rigid payload. There are very few practical applications that may require a shorter cable length. It is also unsafe to fly with a shorter cable length, since the payload may collide with the quadrotor during an uncontrolled swing. A longer cable guards against a payload and vehicle collision, because more energy needs to be transferred to the payload to reach the height of the vehicle. The maximum cable length is selected as $l = 2\,\text{m}$ by intuition since a cable much longer than this may not be practically useful for a drone delivery flight with the considered quadrotor.

Plot MAE vs l DMD shows the same trend revealed in Figure 3.9.



Figuur 3.9: HAVOK prediction error using different cable lengths with different lengths of noisy training data ($m = 0.2\,\mathrm{kg}$, T_{train} = various??.)

Plot MAE vs m DMD shows the same trend revealed in Figure 3.10.



Figuur 3.10: HAVOK prediction error using different payload masses with different lengths of noisy training data ($l = 1\,\mathrm{m}$, T_{train} = various??.)

Best hyperparameters. Fixed size of data. Fixed sample time.

3.6.8. Dynamic payload

Some payloads attached to the cable may not satisfy the assumptions made in Section 3.2. For example, if a long payload is attached to the cable, the CoM of the payload will be quite a distance below the attachment of the cable. This creates a double pendulum model with dynamics that differ significantly from a single pendulum. Figure ?? shows a practical double pendulum use case. In 2D this payload is better represented by the floating double pendulum modelled in Figure ??.

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

It is a non-trivial task to compare the data-driven models with the white-box models, because they are different forms and are used in different types of controllers. The a white-box system identification technique determines a continuous state space model which is used in an LQR controller. This is in the form:

$$\dot{x} = Ax + Bu \tag{3.22}$$

For this model, it is important that the time-derivative estimate of the model at the current time-step is similar to the actual time-derivative of the system state at that time-step. The LQR controller applies an gain to the current state estimate to determine the input signal at that time-step only. Hence, the state prediction for multiple time-steps into the future is not as important.

In contrast, the data-driven techniques result in a discrete state space model,

$$x_{k+1} = Ax_k + Bu_k, \tag{3.23}$$

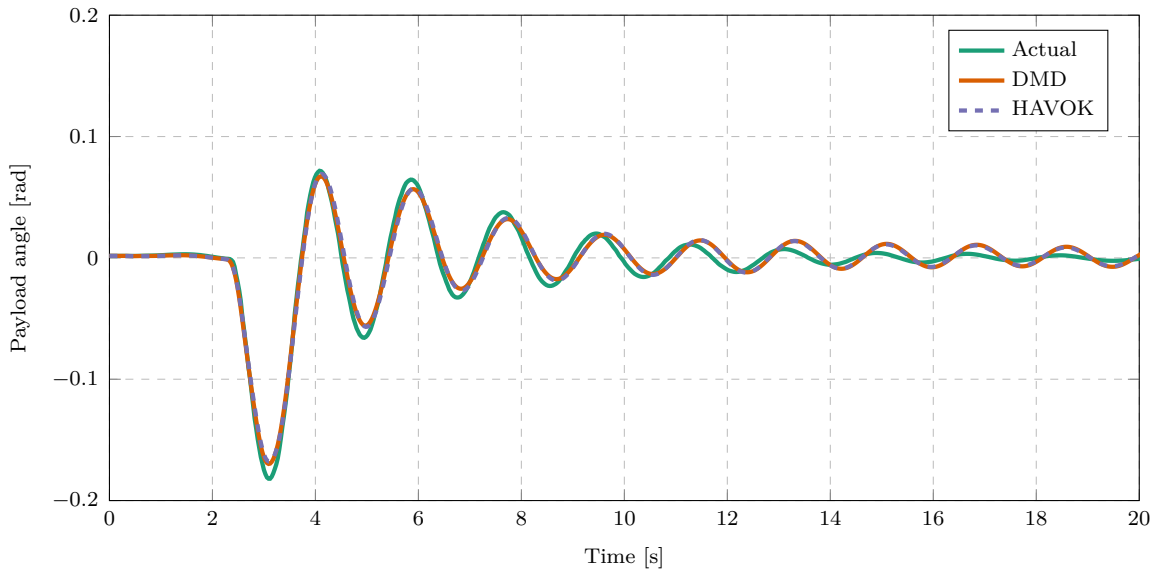
which is used in a MPC. Therefore the prediction accuracy of the model for multiple time-steps over a time horizon is important, since the MPC optimises the control input based on this state prediction.

Even though the models cannot easily be compared to each other, the performance of the controllers using these models can be compared. This comparison will be investigated in

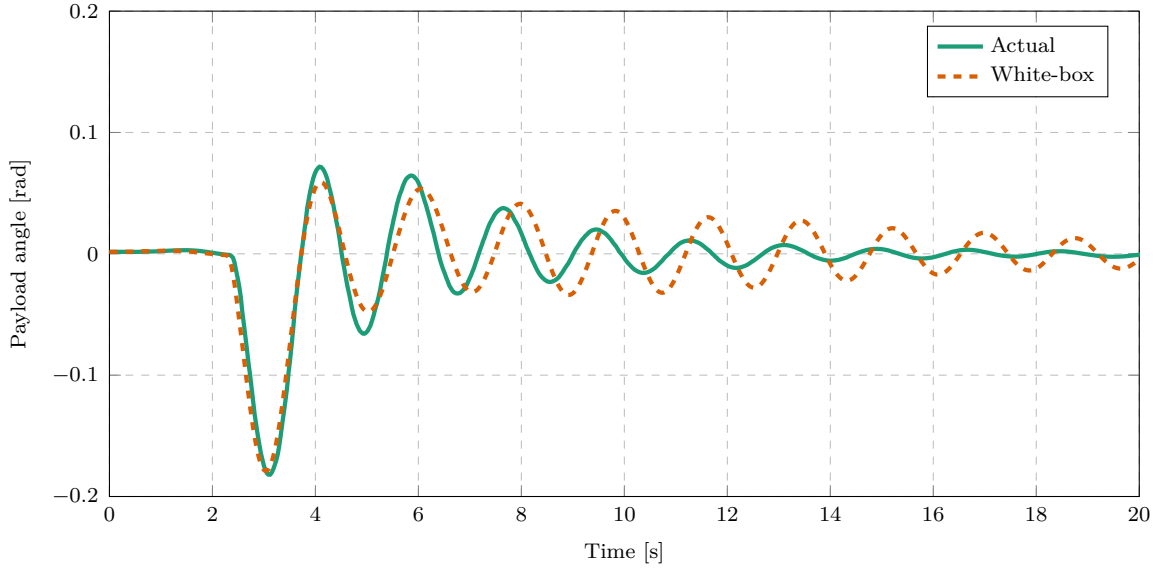
For single pend plot theta prediction dmd vs havok va white-box

Note that the payload oscillations are damped slightly by the PID controller. It appears that this damping is more complex than the linear damping model used to model the white-box model, since the actual swing amplitude does not linearly decrease as expected. The damping effect is a function of the controller gains, the payload connection and the

aerodynamic drag of the payload. An advantage of the data-driven system identification techniques is that the effect of damping is inherently included in the estimated model without specifically estimating it. In contrast, the white-box estimation technique requires a designed algorithm to estimate every parameter that effects the dynamics, namely the payload mass, cable length and damping coefficient



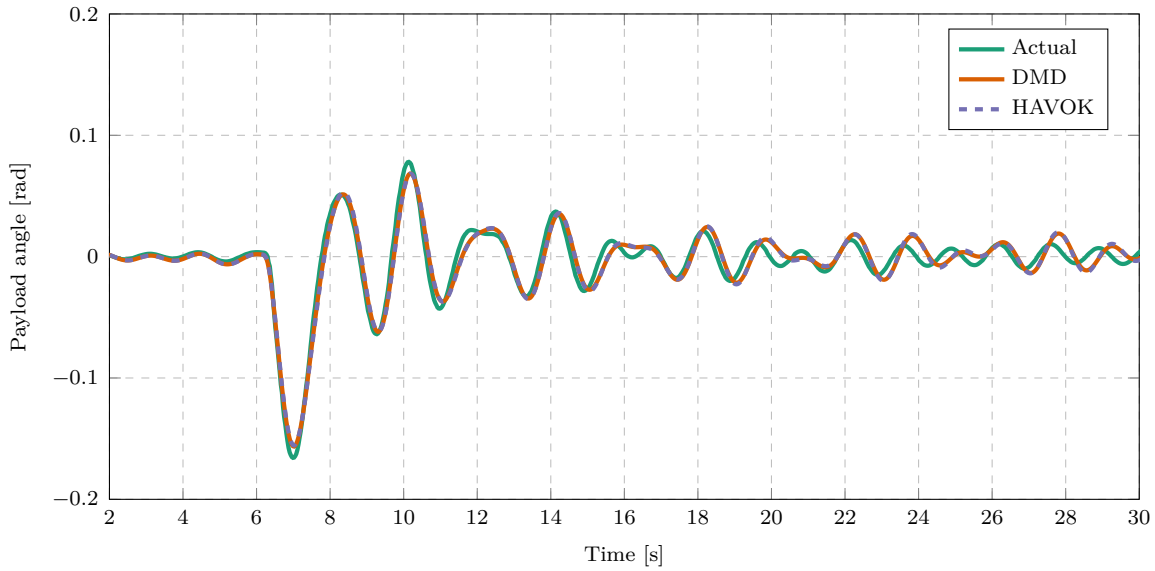
Figuur 3.11: Data-driven model predictions of a single pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).



Figuur 3.12: White-box model predictions of a single pendulum for a North velocity step input ($l = 1\,\text{m}$, T_{train} = various??.)

The FFT of the single pendulum leads to an estimated cable length of 1.19 m

The FFT of the double pendulum leads to an estimated cable length of 1.39 m



Figuur 3.13: Data-driven model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$)

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.

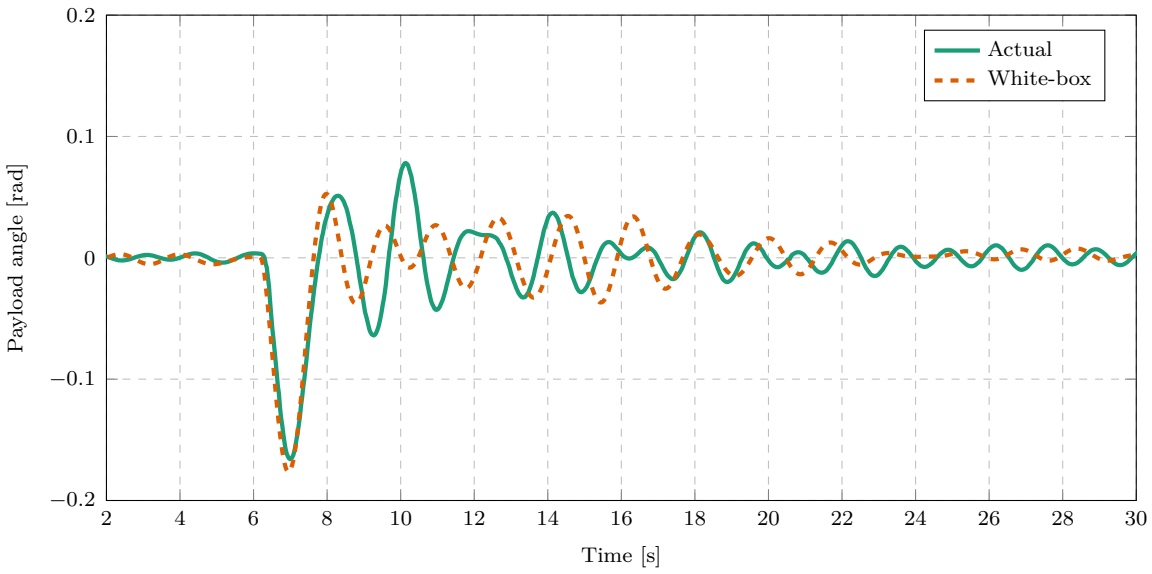
For each of these payload cases, a different parameter estimation based techniques would needs to be designed for effective control. This is undesirable for practical drone deliveries, especially when the type of paylaod is not known well in advance or changes regularly. A data-driven technique provides a more general solution since it accommodates a larger range of payload types and does not require a prioir modelling information.

plot hyperparameterss MAE. Not how much more delays are required

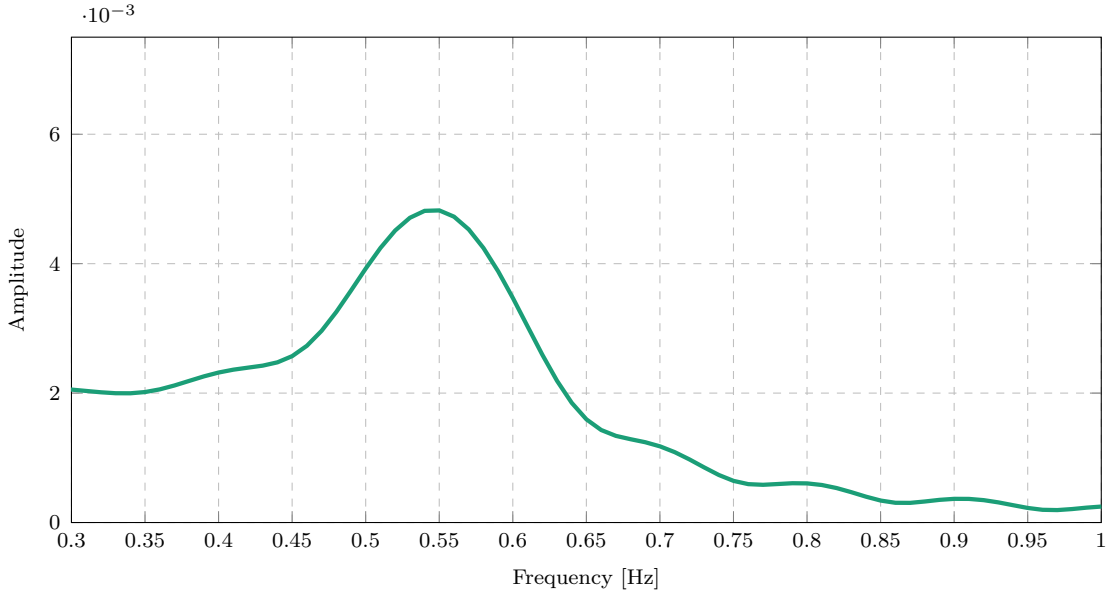
Takens theorum. Add DMD to plot

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:

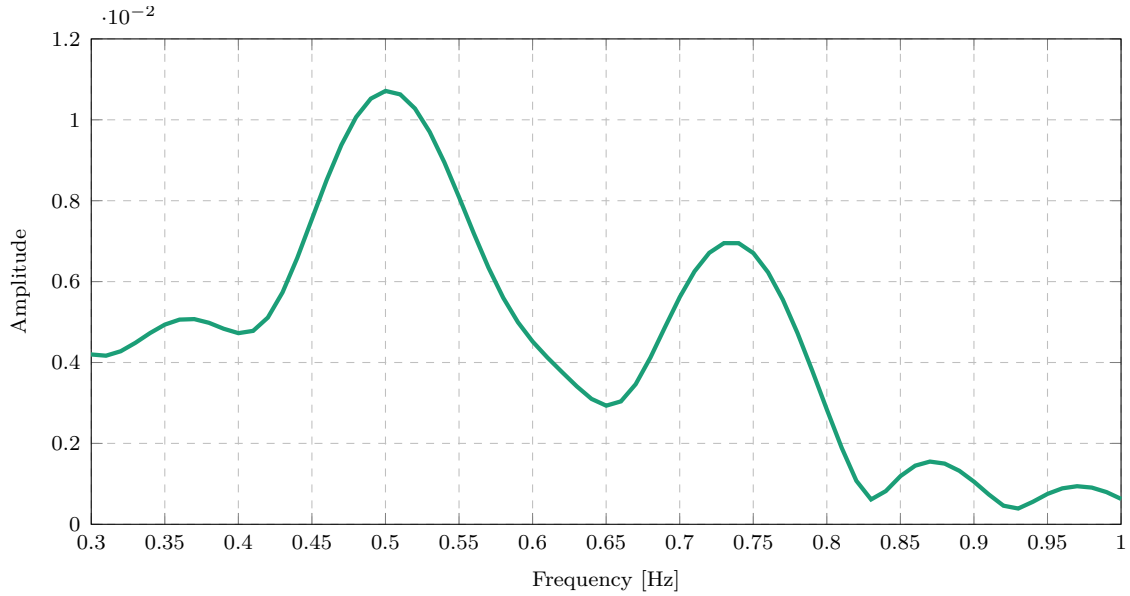
Another payload case that will cause inaccuracies in estimated white-box model is if a payload is attached rigidly to the quadrotor while also carrying a suspended payload. The payload mass estimation is based on the assumption that the quadrotor mass is known. However if a mass is rigidly attached to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



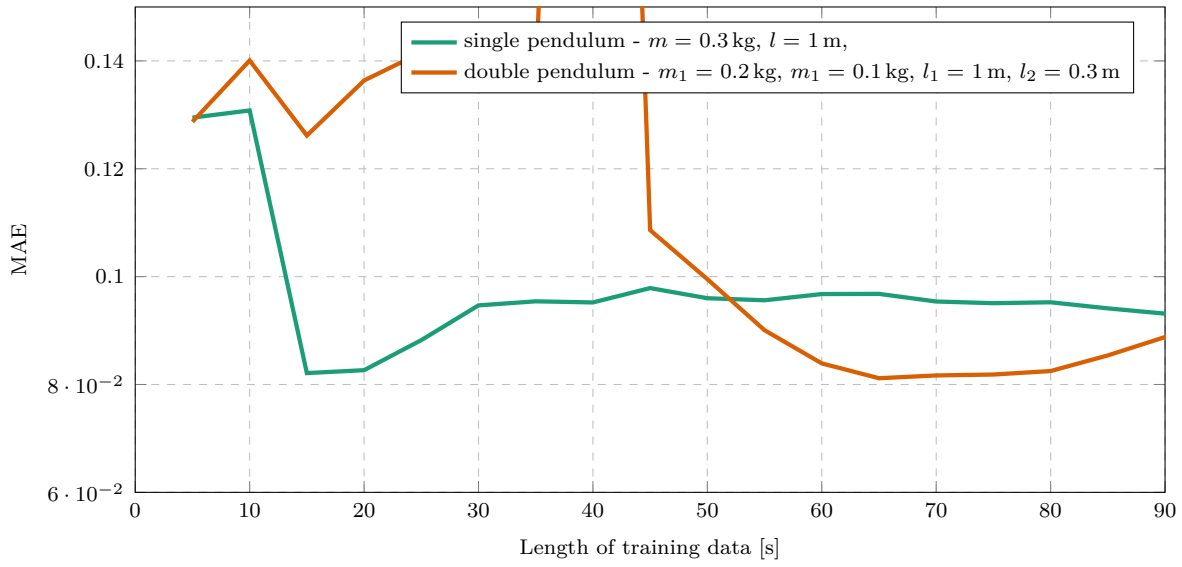
Figuur 3.14: White-box model predictions of a double pendulum for a North velocity step input ($m = 0.3\,\text{kg}$, $l = 1\,\text{m}$)



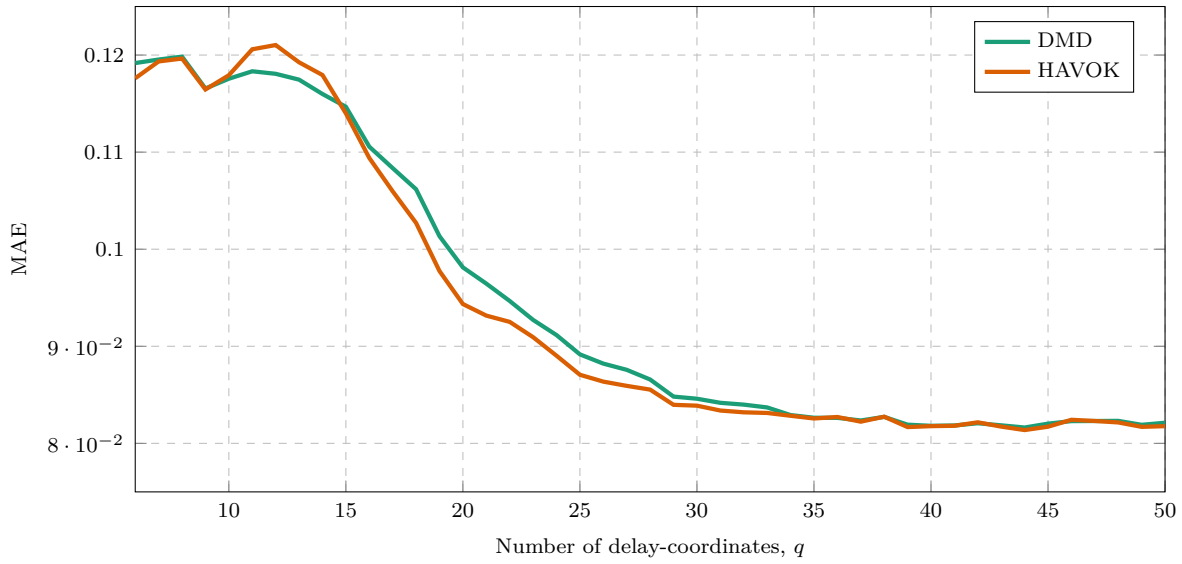
Figuur 3.15: The single-sided amplitude spectrum of the FFT of the single pendulum swing angle ($m = 0.3\,\text{kg}$, $l = 1\,\text{m}$).



Figuur 3.16: The single-sided amplitude spectrum of the FFT of the double pendulum swing angle ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).



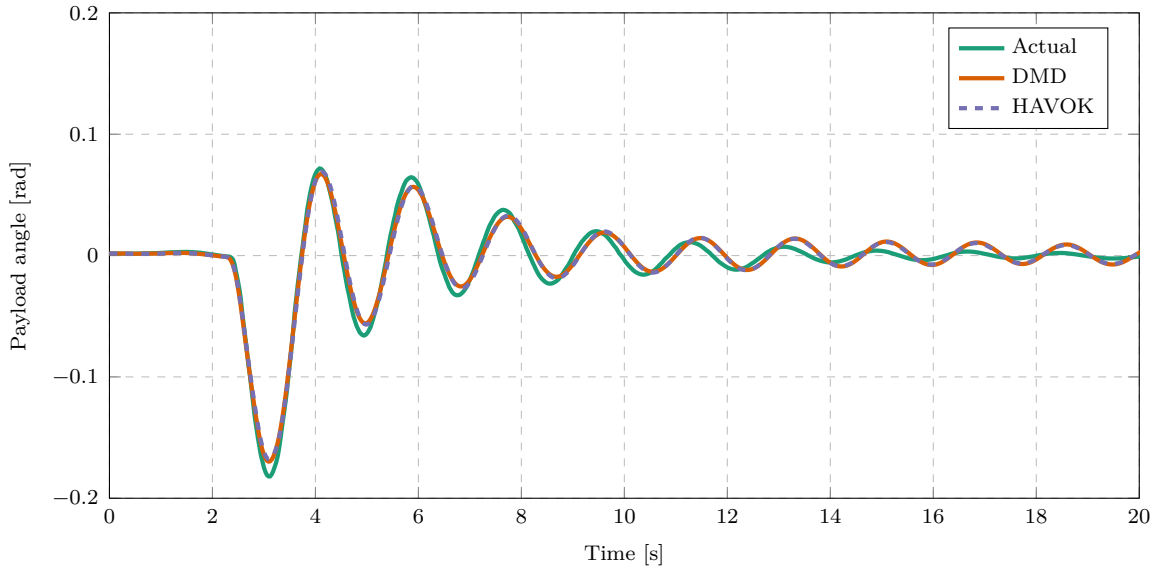
Figuur 3.17: HAVOK prediction error using different cable lengths with a range of different lengths of noisy training data ($l = 1\,\text{m}$, T_{train} = various??.)



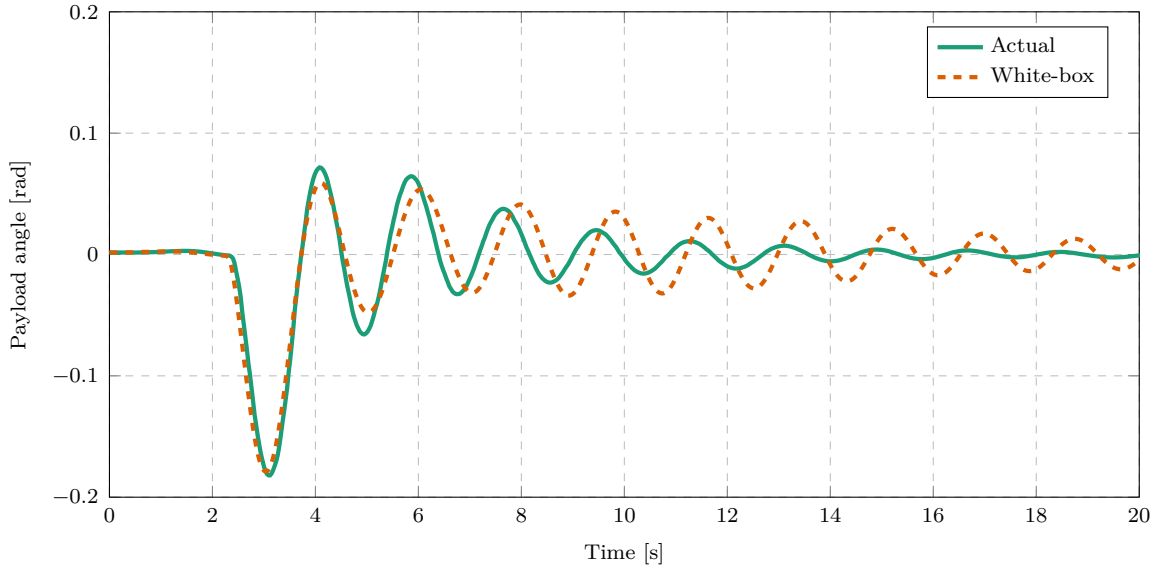
Figuur 3.18: DMD and HAVOK predictions error of double pendulum for different numbers of delay-coordinates ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$ $T_{train} = 70\,\text{s}$.)

3.7. Conclusion

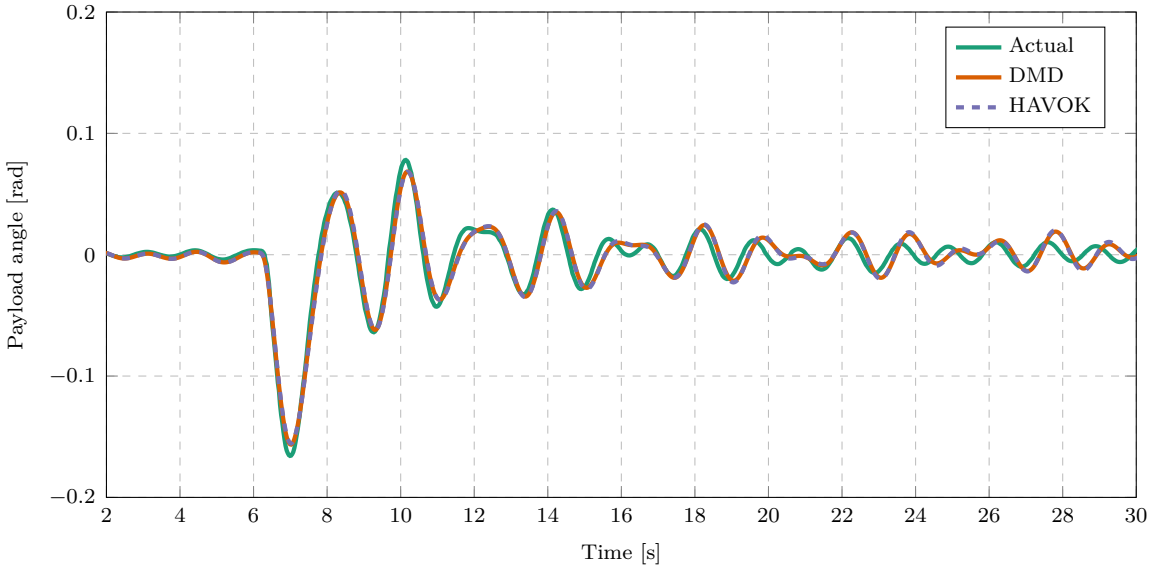
DMD and HAVOK work very similarly with single pend. HAVOK has slightly better prediction accuracy, but this small difference has neglible effect on control. It is difficult to compare the white-box to the black-box models because the real effect will only be seen during control. However it is clear that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



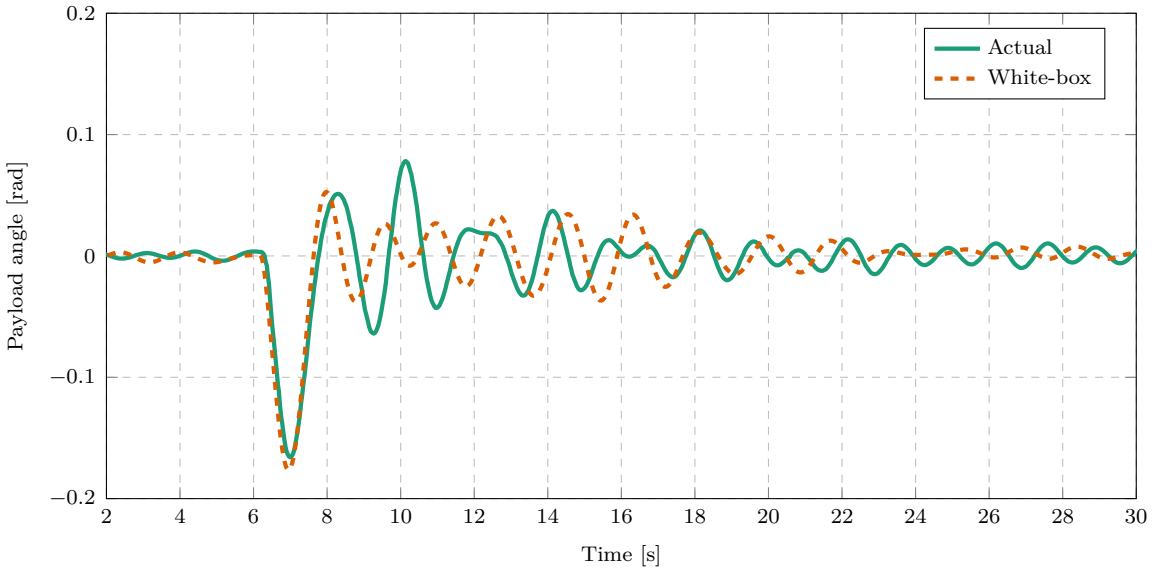
Figuur 3.19: Data-driven model predictions of a single pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).



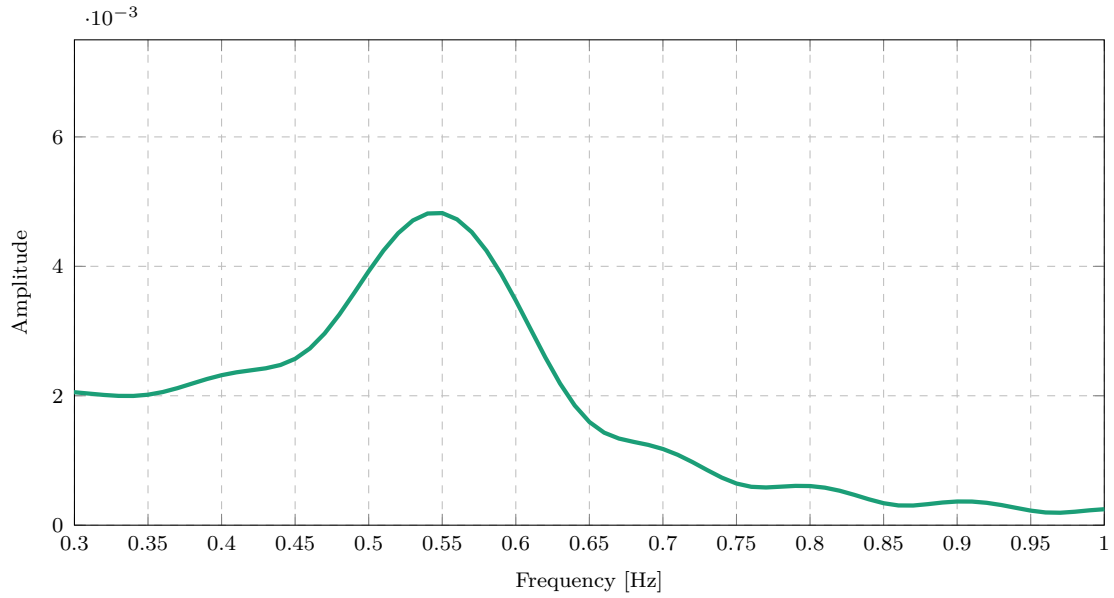
Figuur 3.20: White-box model predictions of a single pendulum for a North velocity step input ($l = 1\,\text{m}$, T_{train} = various??.)



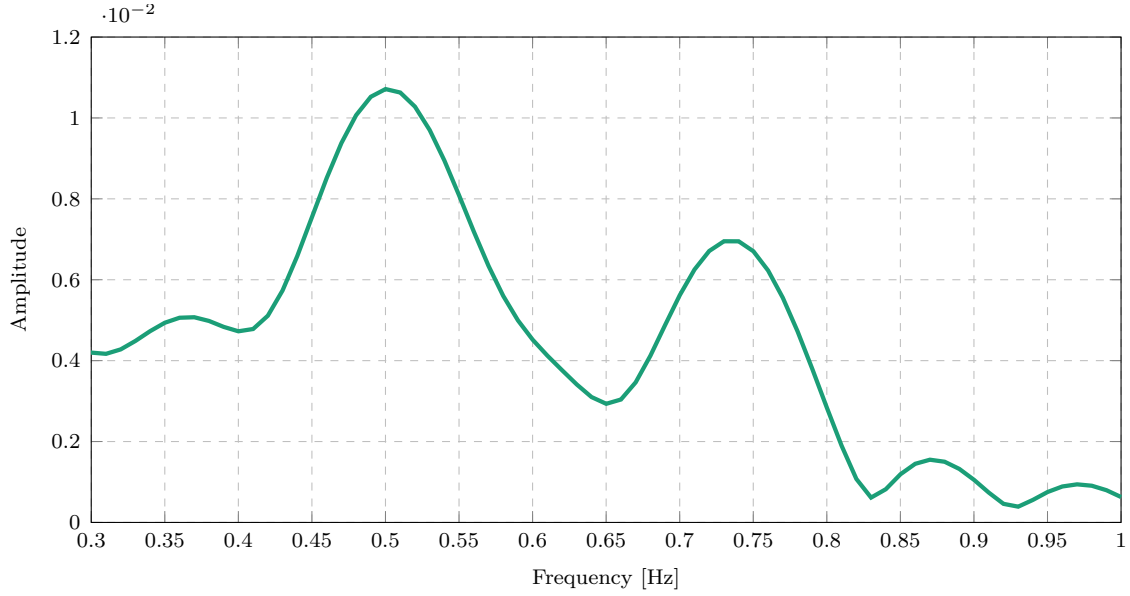
Figuur 3.21: Data-driven model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$)



Figuur 3.22: White-box model predictions of a double pendulum for a North velocity step input ($m = 0.3\,\text{kg}$, $l = 1\,\text{m}$)



Figuur 3.23: The single-sided amplitude spectrum of the FFT of the single pendulum swing angle ($m = 0.3\,\mathrm{kg}$, $l = 1\,\mathrm{m}$).



Figuur 3.24: The single-sided amplitude spectrum of the FFT of the double pendulum swing angle ($m_1 = 0.2\,\mathrm{kg}$, $l_1 = 1\,\mathrm{m}$, $m_2 = 0.1\,\mathrm{kg}$, $l_2 = 0.3\,\mathrm{m}$).

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] 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.

[11] 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.

Bylae A

PID gains

This is an appendix about PID gains used for Honeybee.

Bylae B

Project Planning Schedule

This is an appendix.