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.
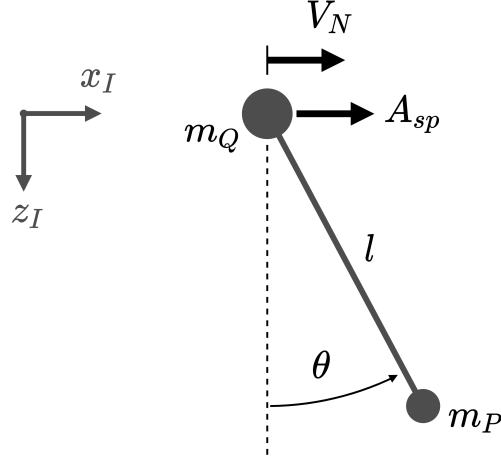
3.2. Plant considered for system identification

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 ˜ 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, ˜ 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:

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

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

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

where ˜ 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 $\tilde{\mathbf{V}}$ are the most significant principal components of the system dynamics [?]. This matrix, $\tilde{\mathbf{V}}$, can be considered to contain a time-series of the pseudo-state, $\mathbf{v}$, such that $\tilde{\mathbf{V}}^T = \begin{bmatrix} \mathbf{v}_q & \mathbf{v}_{q+1} & \cdots & \mathbf{v}_w \end{bmatrix}$, characterises the evolution of the actual dynamics in an eigen-time-delay coordinate system [10]. Consider the following discrete, state-space formulation:

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

Recall that DMDc finds a best fit linear operator that directly maps $\mathbf{a}_k$ to $\mathbf{a}_{k+1}$. Similarly, HAVOK determines the best fit linear operator $\mathbf{\Lambda}$ that maps the pseudo-state $\mathbf{v}_k$ to $\mathbf{v}_{k+1}$. So, in order to setup an over-determined equality for (3.14), $\tilde{\mathbf{V}}^T$ is divided into two matrices:

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

where $\mathbf{V}_2$ is $\mathbf{V}_1$ 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:

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

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:

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

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

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

Note that the matrix entries in (3.18) that map $\mathbf{u}_k$ to $\mathbf{u}_{k+1}$ are meaningless for our purposes and are discarded. Similarly to DMDc, some matrix entries in $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ 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 [11] using the Gazebo simulator [12] 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 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

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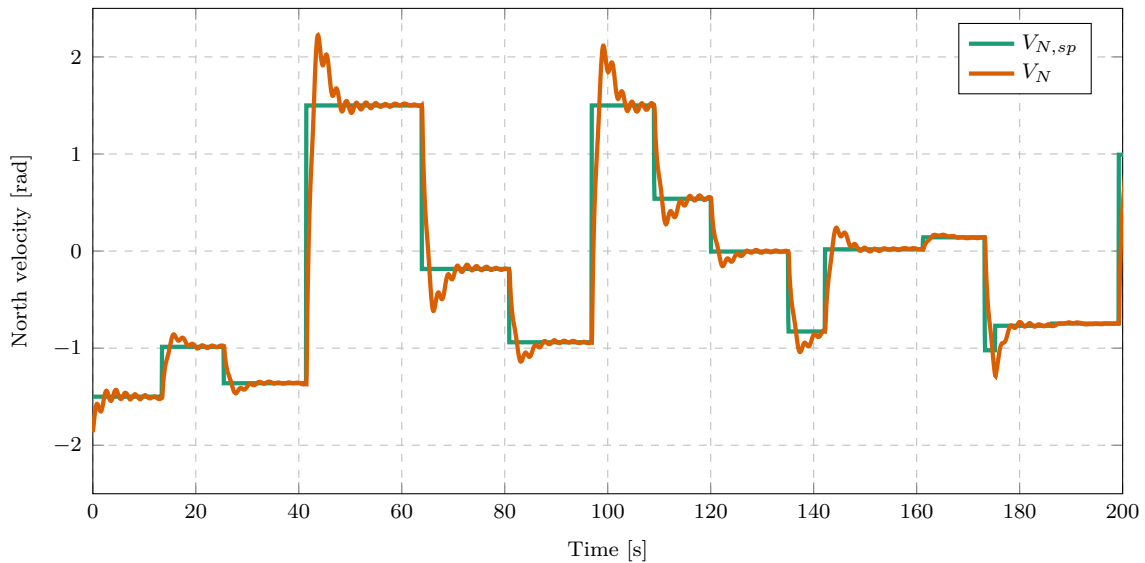
Figuur 3.2: Example of training data with random velocity step inputs ($m = 0.2\,\text{kg}$, $l = 1\,\text{m}$)

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.

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.

3.6.2. Error metric

It is common practice is to select a model for a MPC based on k-step-ahead prediction errors [13]. 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 [13]. 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:
$$\boldsymbol{MAE} = mean\left(\ |\hat{\boldsymbol{x}_k} - \boldsymbol{x_k}|\ \right), \tag{3.19}$$

where $\boldsymbol{x_k}$ is the actual state vector at time-step k, $\boldsymbol{\hat{x}_k}$ 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 [14]. 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 [15], 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 [14]. 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.

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:

$$NMAE_{mm} = \frac{MAE}{x_{max} - x_{min}} \tag{3.20}$$

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

This results in an error metric for each predicted variable, but a single value is required per model to rank the overall accuracy of different models. Therefore the average of the NMAE$_{\text{mm}}$ 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 [16]. 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 3.6.3. 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 $20\,\text{s}$ 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 NMAE_{mm} is determined for each run, whereafter the average of these scores gives the final NMAE_{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 NMAE_{mm} score.

3.6.3. Hyperparameters

As discussed in Section 3.4 and 3.5 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,
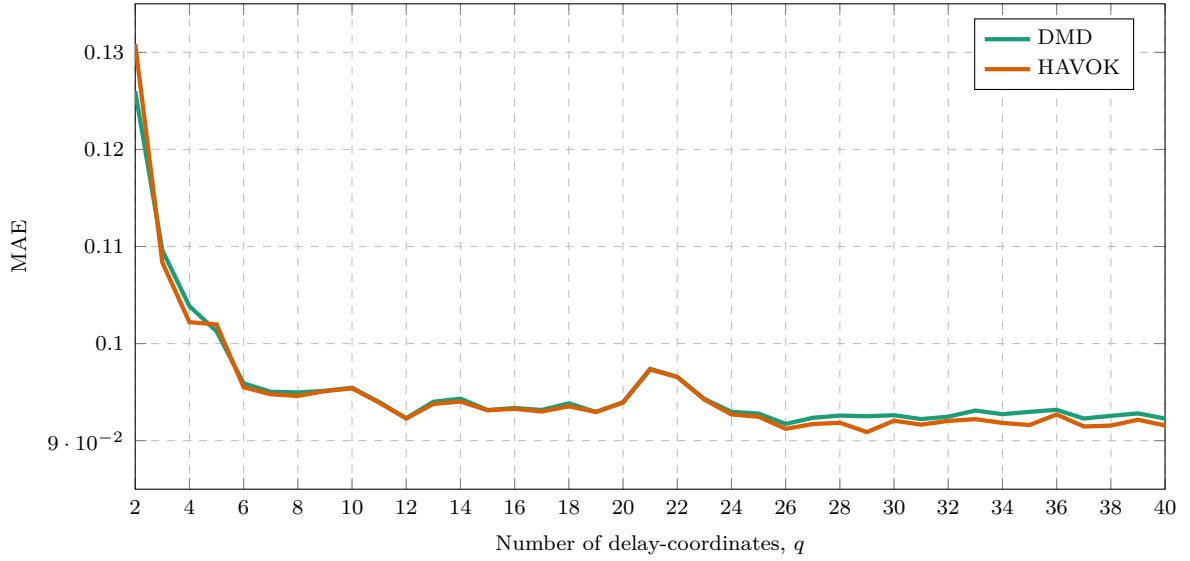
$$5 < q < 30, \qquad 5 < p < 50 \tag{3.21}$$

.

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

Figure 3.3 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 [16]. 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 [17]. These models are less prone to overfitting and also lead to lower computational complexity in the MPC optimisation problem.
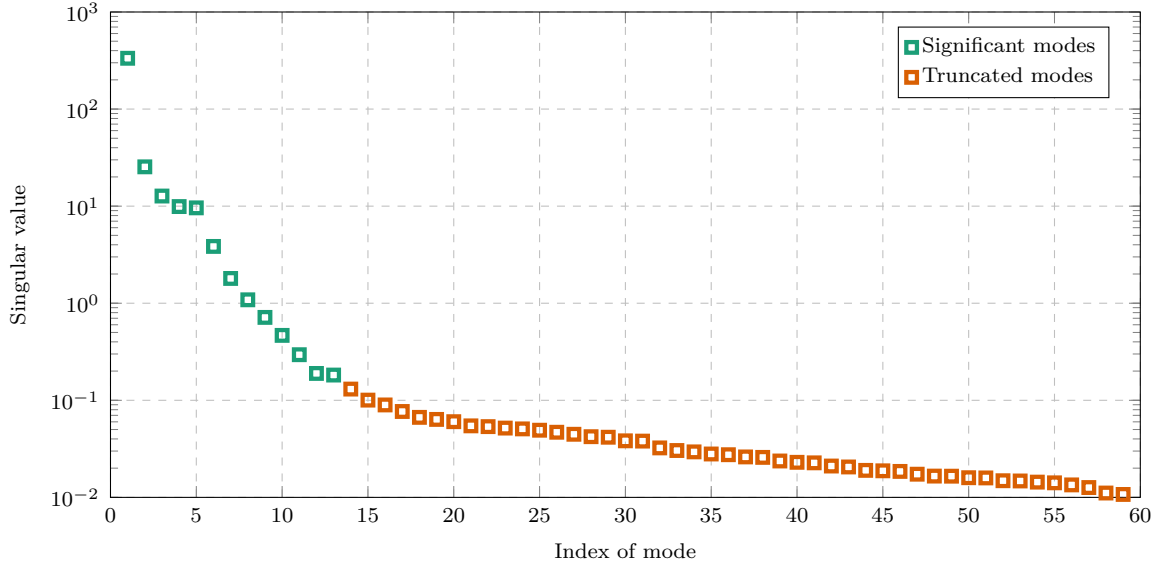
Figuur 3.4: Significant and truncated singular values of a HAVOK model produced from noisy data ($m = 0.2\,\mathrm{kg}$, $l = 0.5\,\mathrm{m}$, $T_s = 0.03\,\mathrm{s}$, $T_{train} = 60\,\mathrm{s}$.)

Figure 3.4 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 [18]. 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 [18].

3.6.4. 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 $50\,\text{Hz}$ which corresponds to $T_s = 0.02\,\text{s}$. Due to the computational complexity of an MPC, the optimiser will struggle to run at $50\,\text{Hz}$ 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 $8.39\,\text{Hz}$ corresponding to a period of $0.119\,\text{s}$.
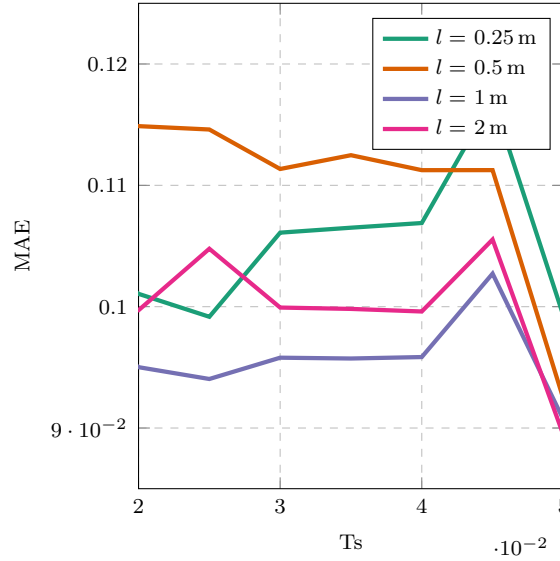
Figuur 3.5: DMD prediction error using different cable lengths with a range of different sample times of noisy training data ($m = 0.2\,\text{kg}$)

Figure 3.5 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 3.5 plots the experiment result for a different cable lengths to see if it has an effect on the prediction error of the models.

It appears that for $l = 0.25\,\text{m}$, the prediction error increases for larger sample times. This is because the shorter cable length results in a higher natural frequency of the payload, hence a faster sample time is required for accurate system identification. 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.

3.6.5. 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,

$$\dot{\theta}_k = (\frac{1}{T_s}) \cdot \theta_k - (\frac{1}{T_s}) \cdot \theta_{k-1}. \tag{3.22}$$

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.
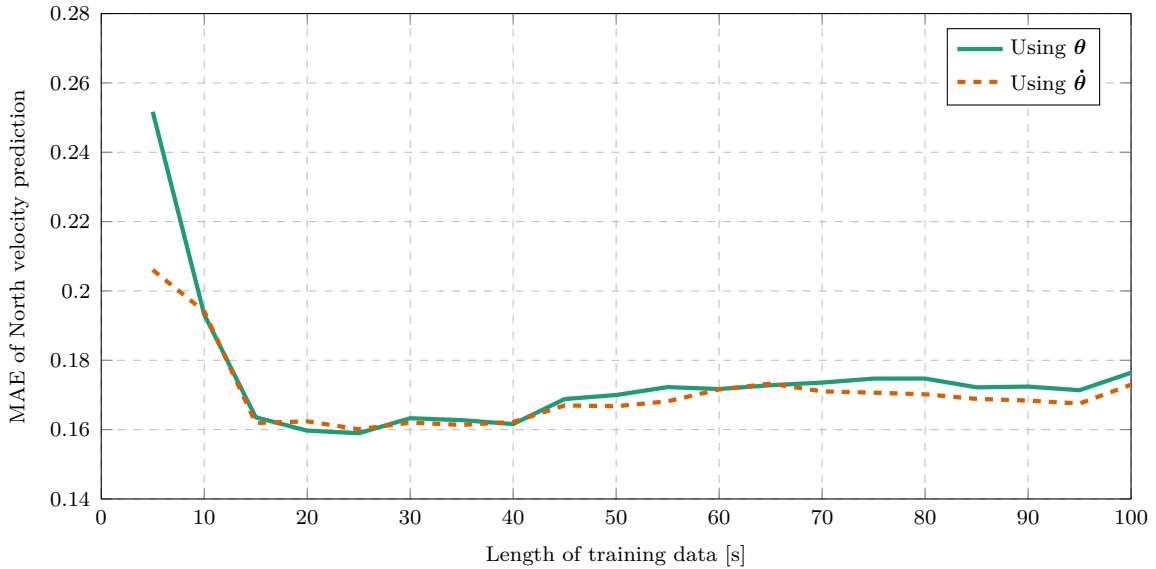
Figuur 3.6: Prediction MAE for HAVOK models using either angle or angular rate measurements ($m = 0.2\,\text{kg}$, $l = 1\,\text{m}$, $T_s = 0.03\,\text{s}$).

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 $\dot{\theta}$.

3.6.6. 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.
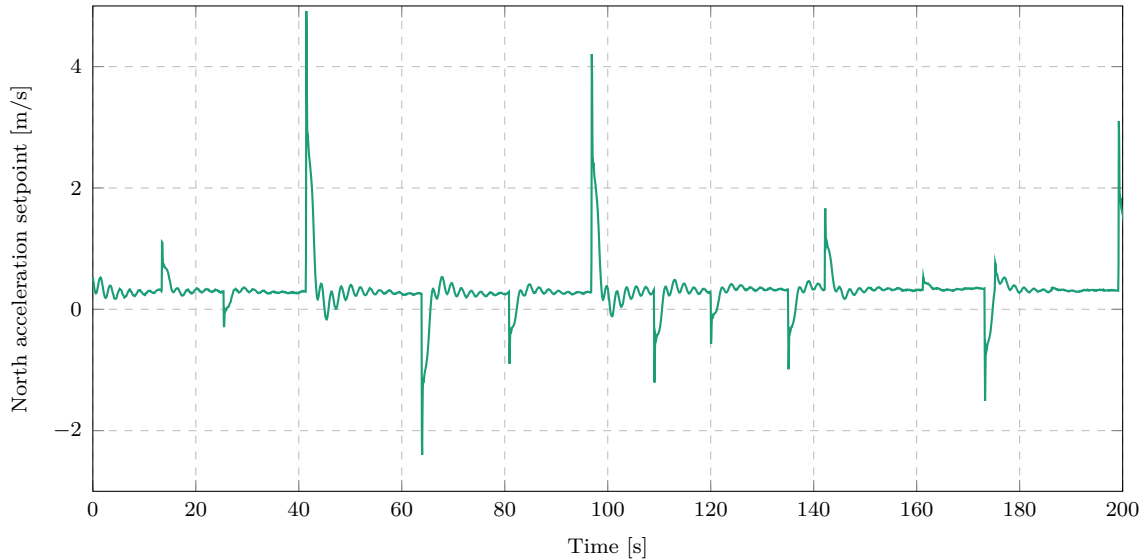
Figuur 3.7: Accelleration setpoint training data from random velocity step inputs ($m = 0.2\,\text{kg}$, $l = 1\,\text{m}$)

The input signal used for system identification is affected by high and low frequency noise. This enters the signal because the acceleration setpoint implementation depends on a noisy orientation vector. which is determined from onboard sensors. The high frequency noise is removed by the quadratic regression smoother, but the low frequency drift is more difficult to address. Figure 3.7 shows the North acceleration setpoint during the training period. Without noise the signal should have a zero mean, however the signal mean shows a constant offset and slow drift.

The constant offset may be due to an imbalance in the CoM of the quadrotor, so the integrators in the PID controllers add a offset to keep the vehicle steady. The IMU may also have a constant measurement offset which affects the orientation vector and control signals. The slow drift is due to low frequency sensor drift. This is addressed by calculating a moving window mean and adjusting for the offset of the signal, which results in a zero mean input signal. The moving window mean is then reapplied to the MPC control signal to readjust for the required offset.
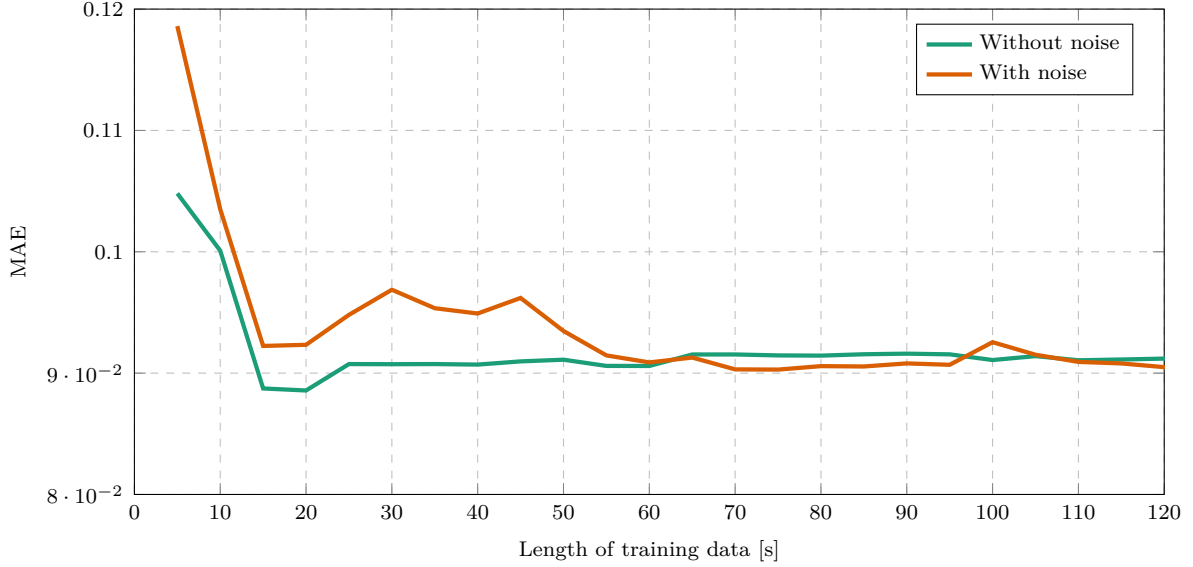
Figuur 3.8: HAVOK prediction error for different lengths of training data with and without noise ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).
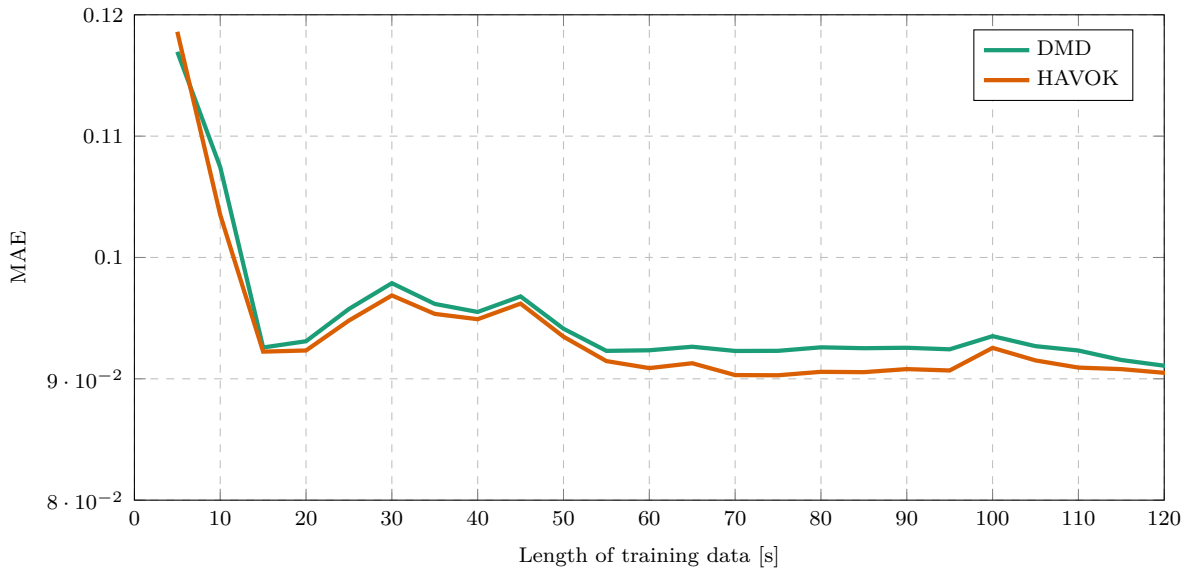
Figuur 3.9: DMD and HAVOK prediction error for different lengths of noisy training data ($m = 0.2\,\text{kg}$, $l = 0.5\,\text{m}$, $T_s = 0.03\,\text{s}$).

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

3.6.7. Size of training data

The length of training data used for system identification affects the quality of the model produced. In Figure 3.6 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.
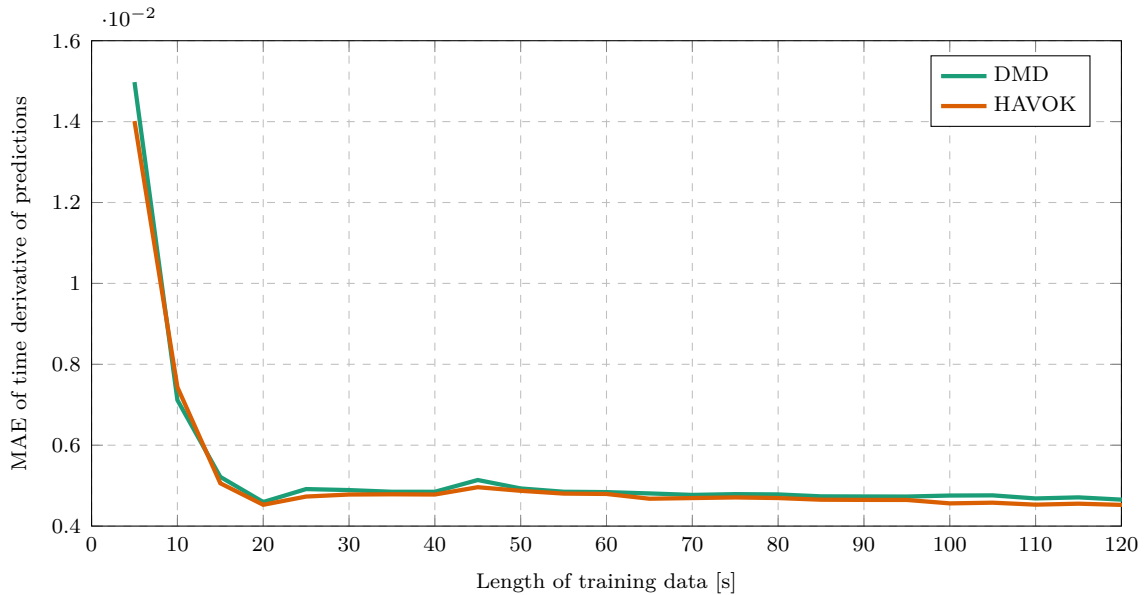
Figuur 3.10: DMD and HAVOK error of time derivative of predictions for different lengths of noisy training data ($m = 0.2 \, \text{kg}$, $l = 0.5 \, \text{m}$, $T_s = 0.03 \, \text{s}$).

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.

3.6.8. 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:

$$0.01 \leq m_p \leq 0.4 \, \text{kg}. \tag{3.23}$$

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:

$$0.5 \leq l \leq 2 \, \text{m}. \tag{3.24}$$

A cable length shorter than $0.5 \, \text{m}$ 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.
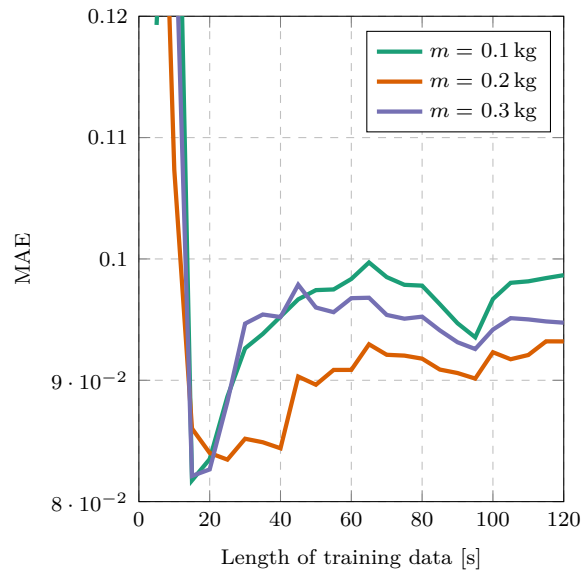
Plot MAE vs l DMD shows the same trend revealed in Figure 3.11.

Plot MAE vs m DMD shows the same trend revealed in Figure 3.12.

Best hyperparameters. Fixed size of data. Fixed sample time.
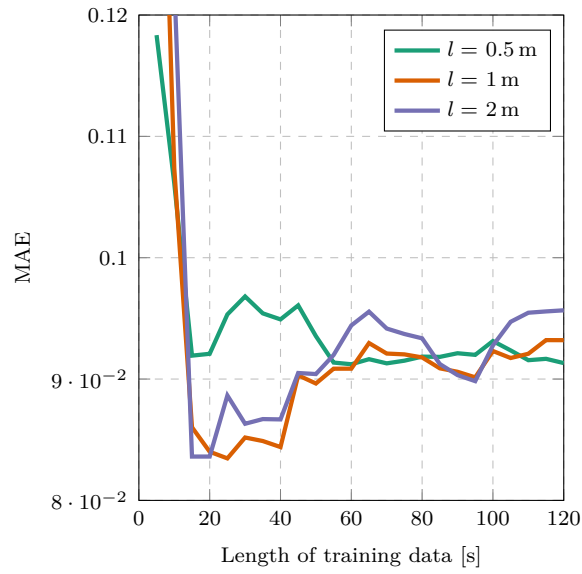
Figuur 3.11: HAVOK prediction error using different cable lengths with different lengths of noisy training data ($m = 0.2\,\text{kg}$, $T_{train} =$ various??.)

Figuur 3.12: HAVOK prediction error using different payload masses with different lengths of noisy training data ($l = 1\,\text{m}$, $T_{train} =$ various??.)

3.6.9. 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:

$$\dot{x} = Ax + Bu \tag{3.25}$$

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,

$$x_{k+1} = Ax_k + Bu_k, \tag{3.26}$$

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
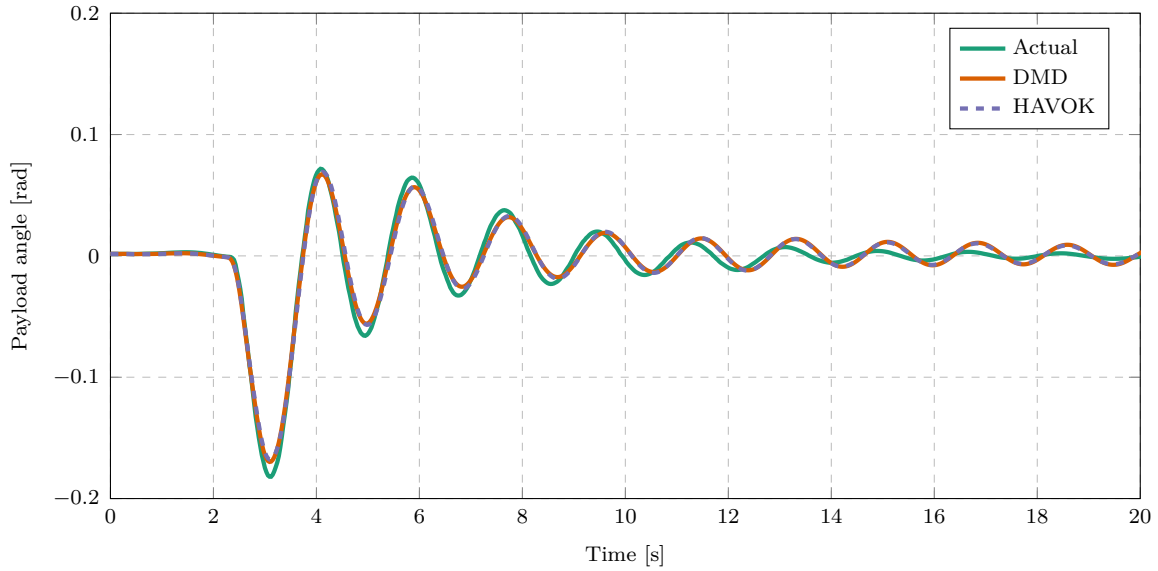

Figuur 3.13: Data-driven model predictions of a single pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).
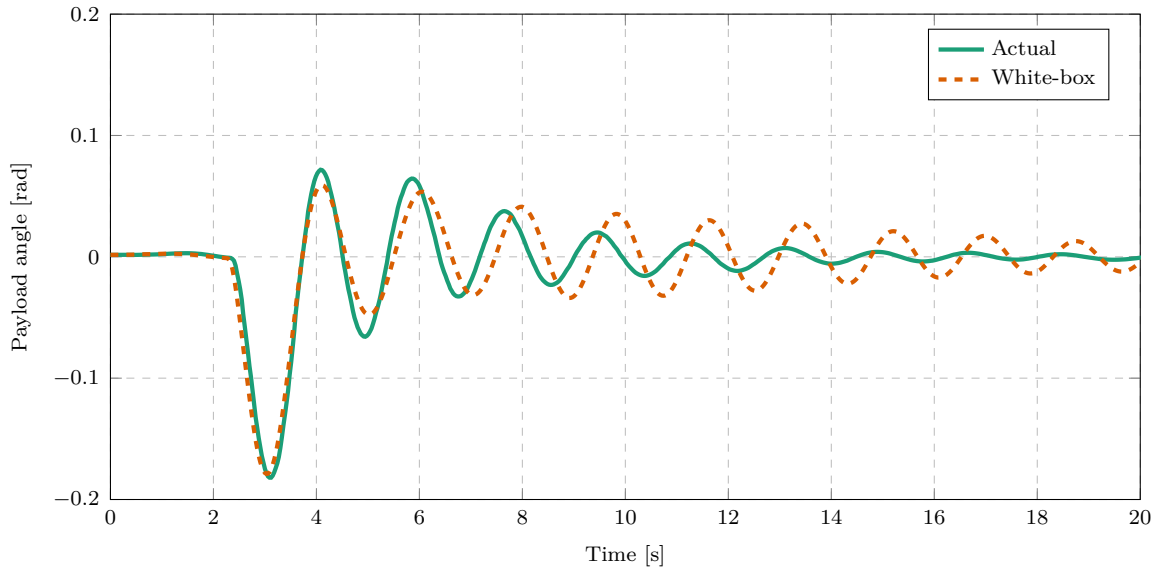
Figuur 3.14: White-box model predictions of a single pendulum for a North velocity step input ($l = 1\,\text{m}$, $T_{train} = $ various??.)

The FFT of the single pendulum leads to an estimated cable length of $1.19\,\text{m}$

The FFT of the double pendulum leads to an estimated cable length of $1.39\,\text{m}$
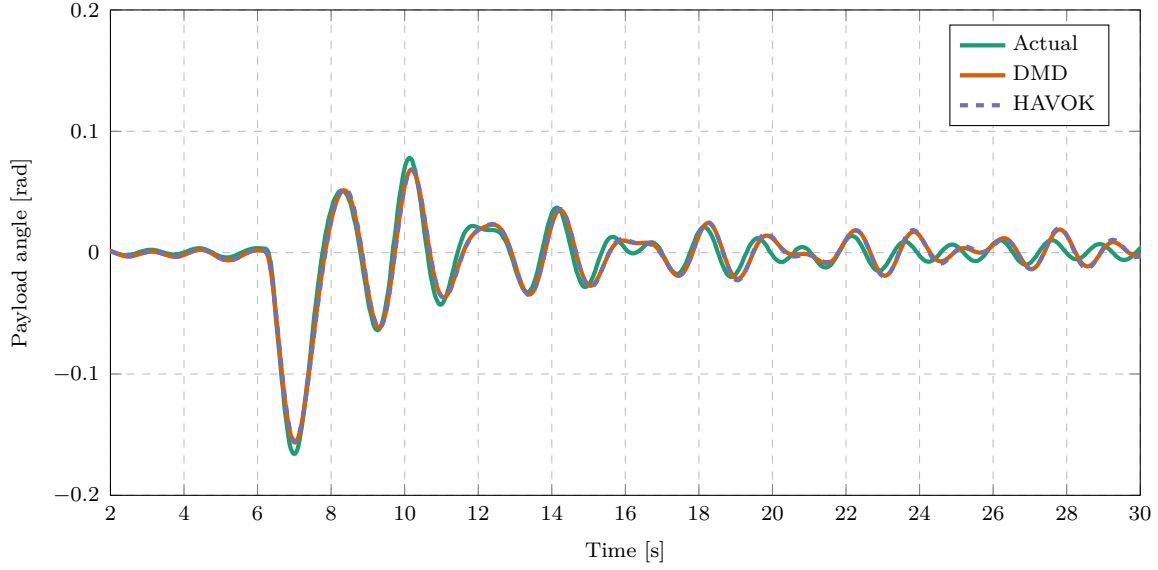
Figuur 3.15: Data-driven model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$)

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.
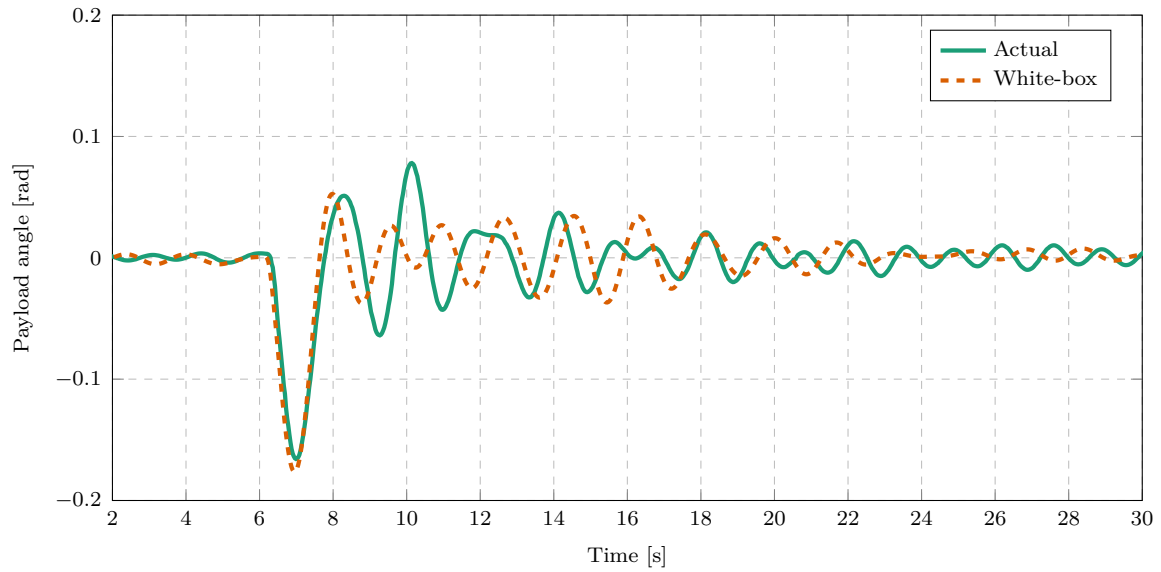
Figuur 3.16: White-box model predictions of a double pendulum for a North velocity step input ($m = 0.3\,\mathrm{kg}$, $l = 1\,\mathrm{m}$)
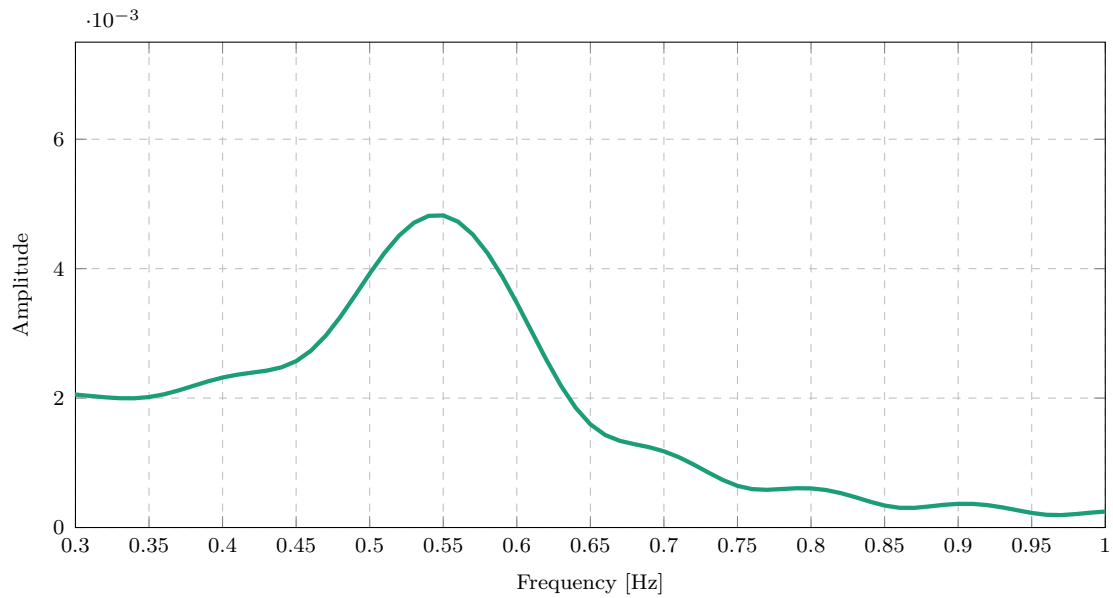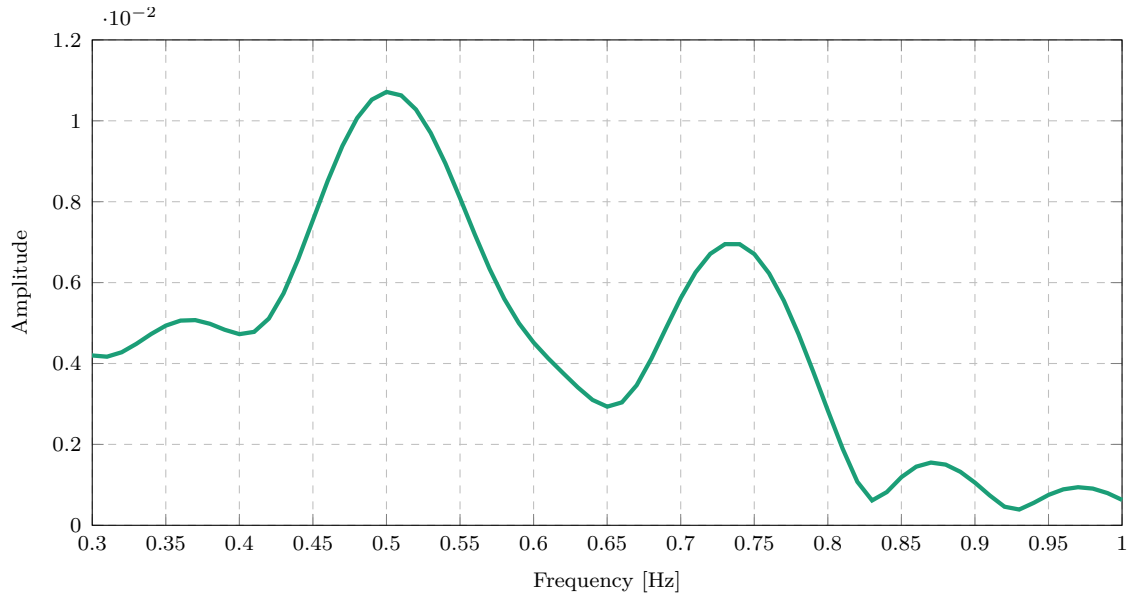

Figuur 3.17: The single-sided amplitude spectrum of the FFT of the single pendulum swing angle ($m = 0.3\,\mathrm{kg}$, $l = 1\,\mathrm{m}$).

Figuur 3.18: The single-sided amplitude spectrum of the FFT of the double pendulum swing angle ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).
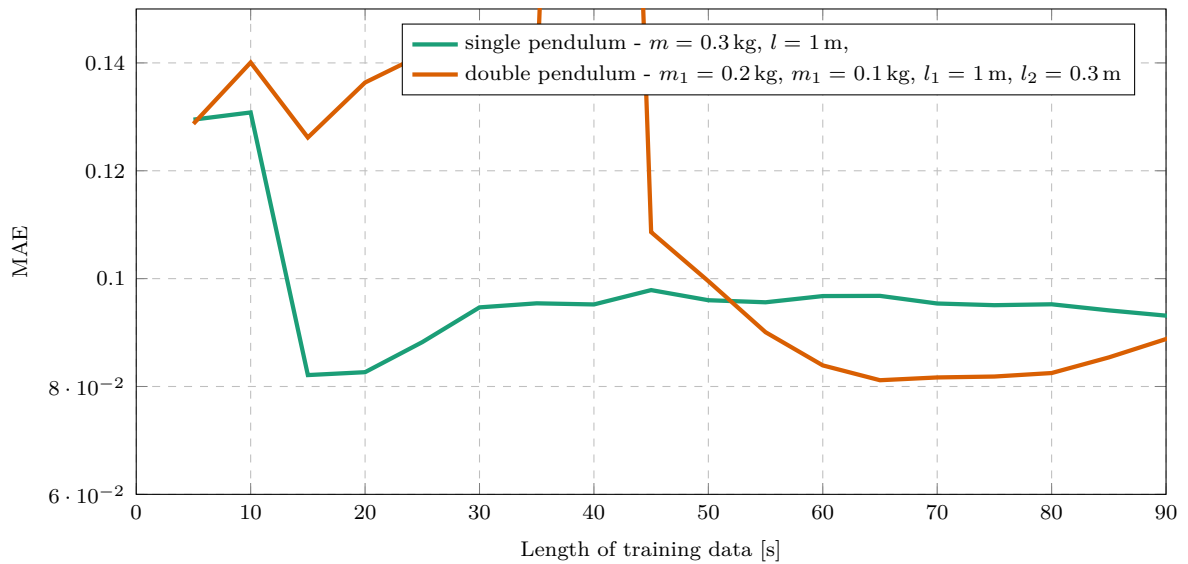
Figuur 3.19: HAVOK prediction error using different cable lengths with a range of different lengths of noisy training data ($l = 1\,\text{m}$, $T_{train} = $ various??.)
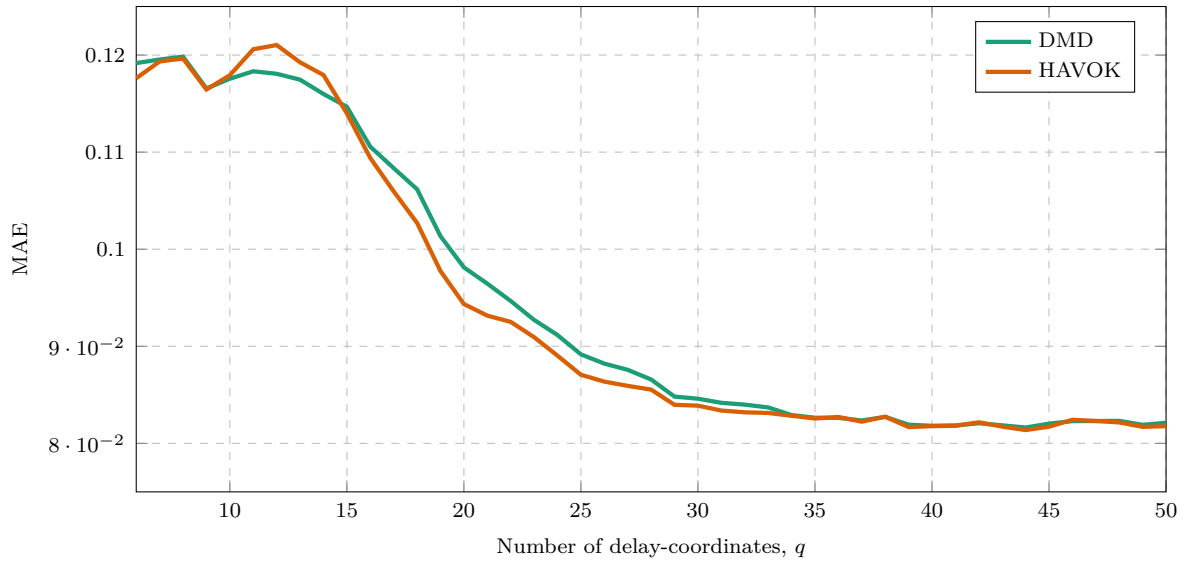
Figuur 3.20: DMD and HAVOK predictions error of double pendulum for different numbers of delay-coordinates ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$ $T_{train} = 70\,\text{s.}$)

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.
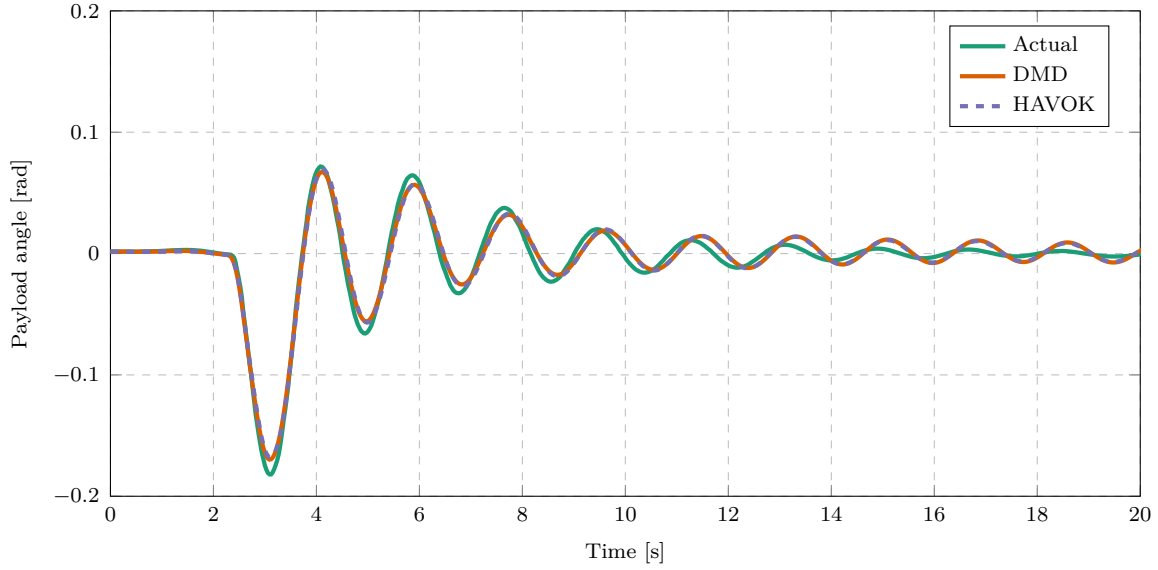
Figuur 3.21: Data-driven model predictions of a single pendulum for a North velocity step input ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).
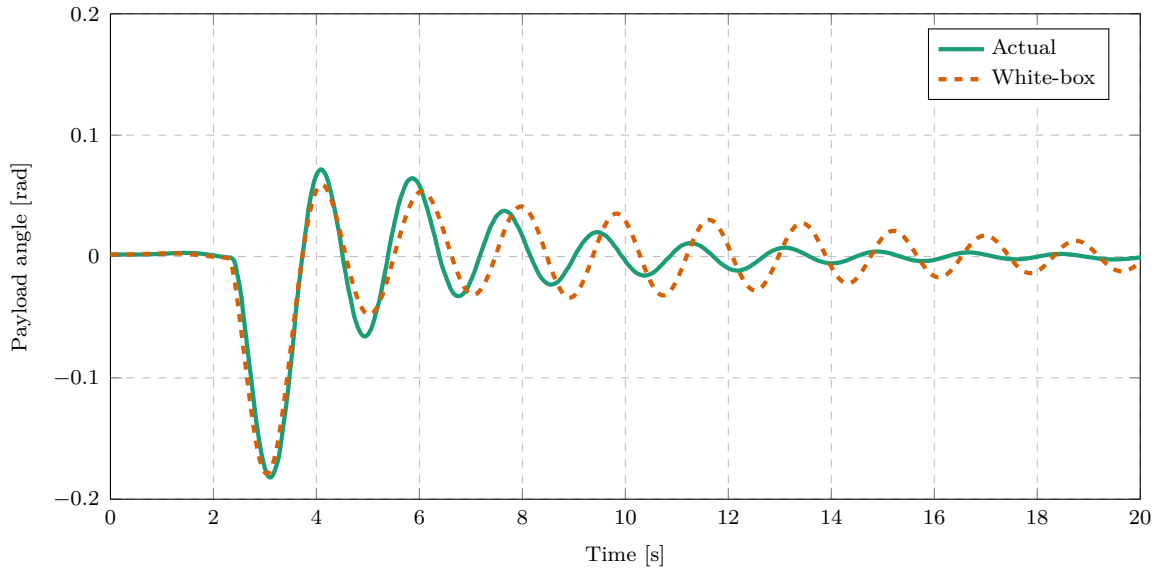
Figuur 3.22: White-box model predictions of a single pendulum for a North velocity step input ($l = 1\,\text{m}$, T_{train} = various??.)
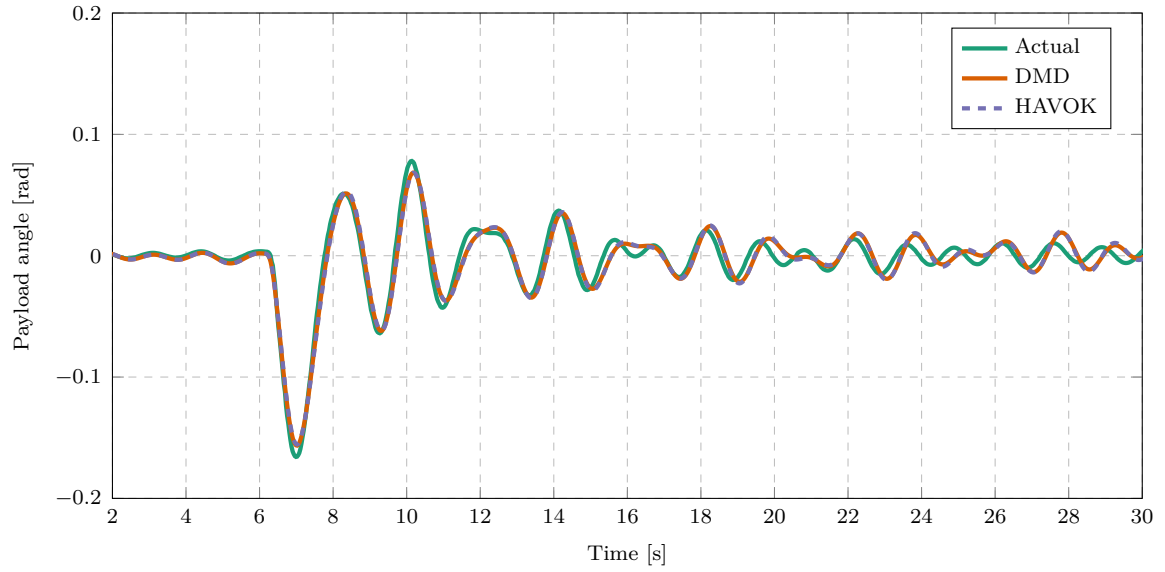
Figuur 3.23: Data-driven model predictions of a double pendulum for a North velocity step input ($m_1 = 0.2$ kg, $l_1 = 1$ m, $m_2 = 0.1$ kg, $l_2 = 0.3$ m)
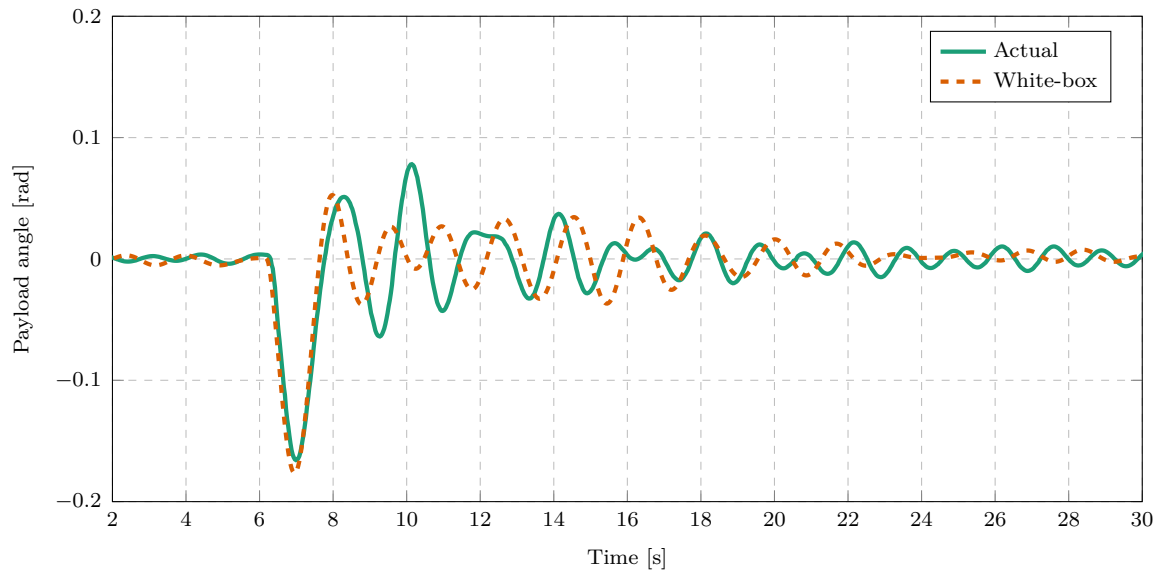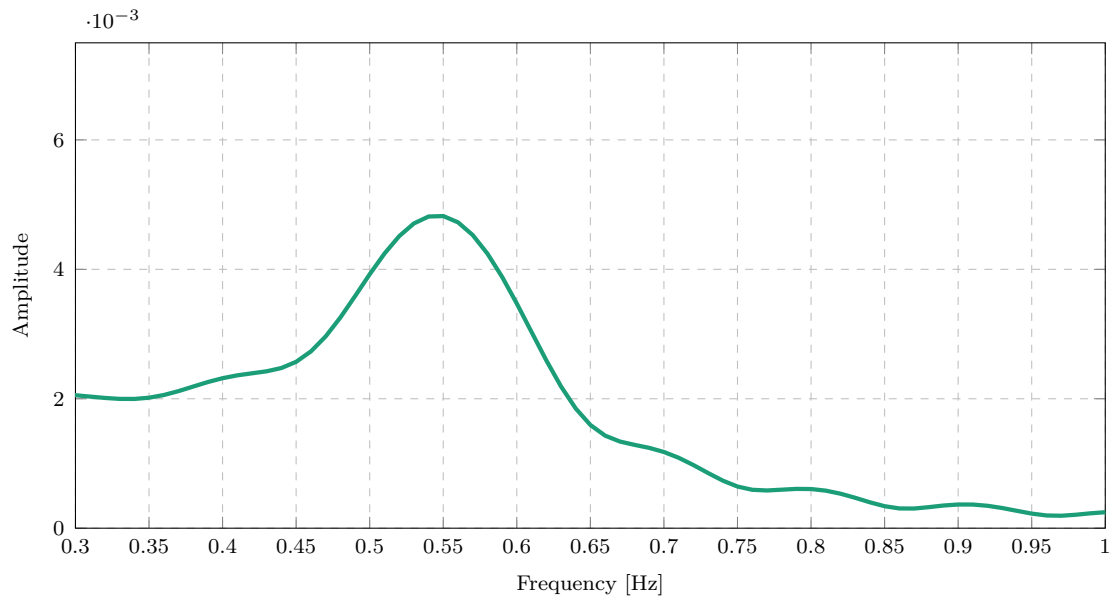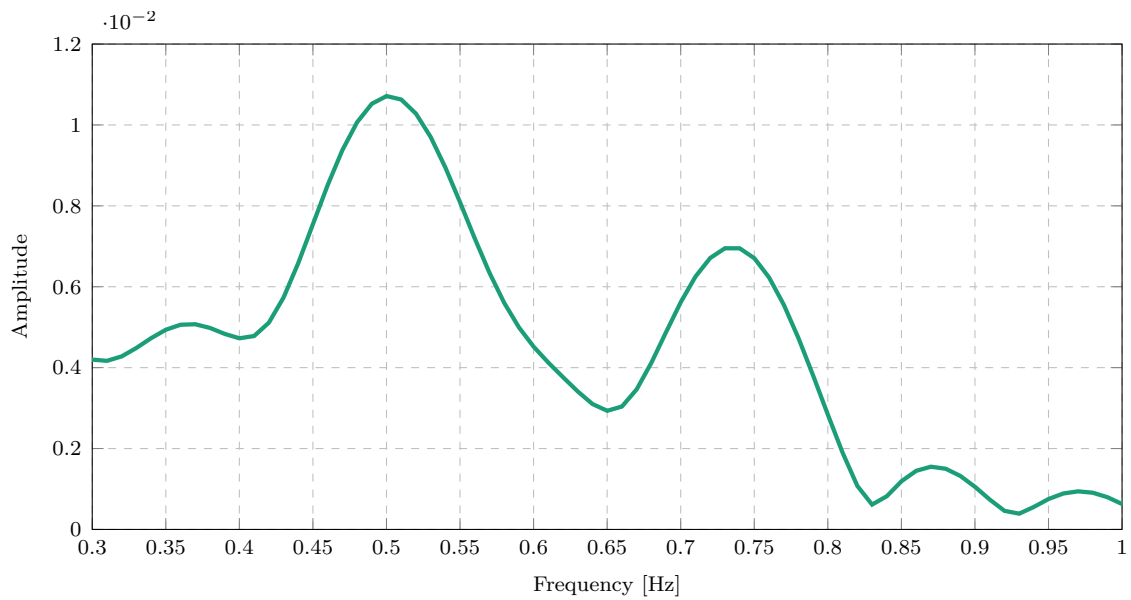

Figuur 3.24: White-box model predictions of a double pendulum for a North velocity step input ($m = 0.3$ kg, $l = 1$ m)

Figuur 3.25: The single-sided amplitude spectrum of the FFT of the single pendulum swing angle ($m = 0.3\,\text{kg}$, $l = 1\,\text{m}$).

Figuur 3.26: The single-sided amplitude spectrum of the FFT of the double pendulum swing angle ($m_1 = 0.2\,\text{kg}$, $l_1 = 1\,\text{m}$, $m_2 = 0.1\,\text{kg}$, $l_2 = 0.3\,\text{m}$).

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] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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

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

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

Bylae A

PID gains

This is an appendix about PID gains used for Honeybee.

Bylae B

Project Planning Schedule

This is an appendix.