



Budapest University of Technology and Economics  
Faculty of Electrical Engineering and Informatics  
Department of Control Engineering and Information Technology

# Controlling the nonlinear flight dynamics of quadcopters with linear surrogate-modeling-based approaches

**Scientific Students' Association Report**

Author:

Máté Szécsi

Consultant:

*Dr. Roland Tóth*

Senior research fellow  
HUN-REN SZTAKI

Supervisor:

*Dr. István Harmati*

Associate professor  
BME-IIT

Advisor:

*Dr. Tamás Péni*

Senior research fellow  
HUN-REN SZTAKI

# Contents

|          |                                                             |           |
|----------|-------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                         | <b>2</b>  |
| <b>2</b> | <b>The Koopman Framework</b>                                | <b>5</b>  |
| 2.1      | Koopman representation of autonomous systems . . . . .      | 5         |
| 2.2      | Koopman representation of systems with inputs . . . . .     | 7         |
| 2.3      | Deep-learning of Koopman surrogates . . . . .               | 8         |
| 2.3.1    | Data generating system, and model parametrization . . . . . | 8         |
| 2.3.2    | The SUBNET method . . . . .                                 | 10        |
| <b>3</b> | <b>Koopman modeling of quadrotor dynamics</b>               | <b>12</b> |
| 3.1      | Mathematical model of a quadcopter . . . . .                | 12        |
| 3.2      | Data collection . . . . .                                   | 14        |
| 3.2.1    | Experiment design . . . . .                                 | 14        |
| 3.2.2    | Data quality . . . . .                                      | 15        |
| 3.3      | Identification results . . . . .                            | 17        |
| <b>4</b> | <b>Controller design</b>                                    | <b>21</b> |
| 4.1      | LPV controller design for Koopman models . . . . .          | 21        |
| 4.1.1    | Control structure and shaping . . . . .                     | 22        |
| 4.1.2    | Signal shaping . . . . .                                    | 23        |
| 4.1.2.1  | Performance objectives . . . . .                            | 24        |
| 4.1.3    | Gridding the scheduling space . . . . .                     | 24        |
| 4.1.4    | Synthesis of the LPV controller . . . . .                   | 25        |
| 4.1.5    | Scattered interpolation between feedback gains . . . . .    | 27        |
| 4.2      | PI for position tracking . . . . .                          | 28        |
| <b>5</b> | <b>Koopman control of the quadrotor</b>                     | <b>30</b> |
| <b>6</b> | <b>Conclusions</b>                                          | <b>35</b> |
|          | <b>Acknowledgements</b>                                     | <b>37</b> |



# Chapter 1

## Introduction

In modern engineering, the need for higher performance standards, such as reducing environmental impact and improving energy efficiency, has made model-based control methods increasingly dominant in the vehicle industry, as these methods offer superior performance over traditional non-model-based techniques. In these approaches the capability of the model to represent the control relevant dynamics of the system has a significant role in determining the performance of the controllers. As a result we are aiming to create models that capture a high level of detail of the dynamics of our system.

Having a precise representation of the system dynamics is also useful for the construction of a digital twin of the model, which can be used to simulate and monitor the system in real time. By accurately replicating the behavior of the physical system, the digital twin enables the performance analysis of the system. Additionally, it enables the synthesis, and testing of control algorithms, without interfering with the real world system.

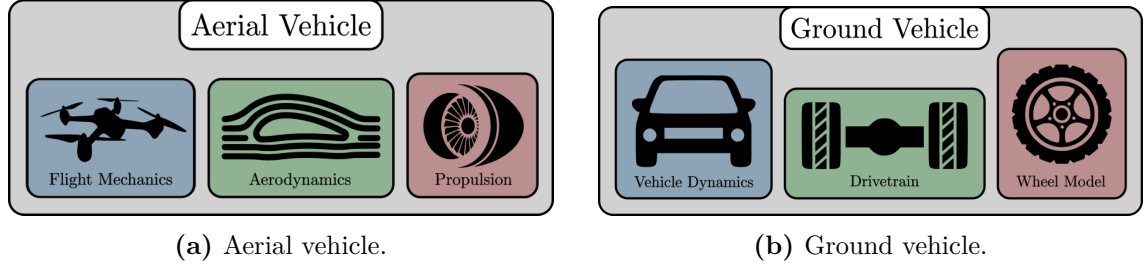
However, development of a high performing controller presents two main challenges.

Firstly, precise modeling is often challenging. Traditionally, models are developed from first principles. However, as we attempt to model finer aspects of the dynamics, it becomes increasingly difficult and time consuming. For complex systems like aerial and ground vehicles, as illustrated in Figure 1.1, accurately modeling specific dynamics, such as tire behavior and aerodynamic effects, is notoriously challenging. Parameter estimation for these models can also be complex, often requiring multiple experiments to estimate different parameters, further complicating the optimization process. Lack of a precise model often necessitates robust controller synthesis, where unknown behavior and model inaccuracies are addressed by characterizing its uncertainties. However, this approach inevitably leads to a deteriorated performance, as the controller must account for a broader range of variations in the dynamics rather than optimizing for a specific, accurate model.

As the dynamics of most systems contain some nonlinearity, an obtained model which precisely represents it also tends to be nonlinear. This introduces the second challenge — controller synthesis for nonlinear systems. Nonlinear control techniques, such as backstepping and control Lyapunov functions [14], are complex and primarily offer stability guarantees, while controller design for linear time-invariant (LTI) systems is well-developed, with established methods like  $H_2$  or  $H_\infty$  control that are easy to use, and offer performance shaping methods such as *mixed-sensitivity* or *loop-shaping*. Such a control design toolchain is largely absent for nonlinear systems.

These two objectives—precise modeling and effective control—are often at odds with each other. Accurate models tend to be nonlinear, limiting the applicability of the well-

established LTI controller framework. On the other hand, using simpler models - such as ones that approximate system dynamics at a particular operating point - sacrifices model accuracy, limiting controller performance.

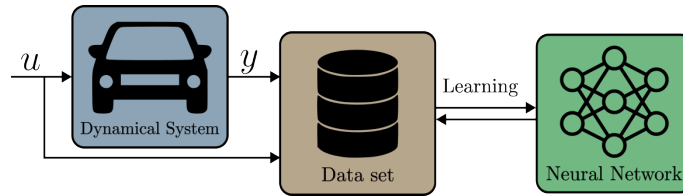


**Figure 1.1:** Dynamics of vehicles.

To address these challenges, we utilized the Koopman framework [17] as our modeling approach. This framework allows us to represent the system model in a linear form governed by the linear Koopman operator. A model obtained in this representation enables the use of the powerful control design toolchain for LTI systems. The central idea of Koopman modeling is to express nonlinear dynamics in a generally infinite-dimensional state space, within which the system evolves in a Koopman-invariant subspace. This linear state space is constructed through "observable functions", which are nonlinear mappings from the physical state space to the latent Koopman space. The main challenge lies in selecting suitable observable functions that accurately capture the nonlinear dynamics in this Koopman-linear state space, which, in practice, is a finite-dimensional approximation.

The modeling challenge can be addressed by using data-driven, black-box, or grey-box methods, which are general, and not system specific, hence simplifying the identification process. By collecting data that capture the dynamics of the system, these methods can produce accurate models.

Neural network-based approaches show much promise, as Deep Neural Networks (DNN) are capable of representing relationships of very high complexity, provided a sufficiently high-quality dataset is available. In this work we chose to use the Koopman-SUBNET method [3] which models the system as a recurrent neural network (RNN) in state-space form. In addition to the estimation of the state transition function, this method takes care of the construction of the state-space representation automatically by estimating an encoder function.



**Figure 1.2:** Data-based identification.

Once a system representation in Koopman-linear form is obtained, we will synthesize a controller using classical  $\ell_2$  gain-optimal LPV control.

While our Koopman-modeling procedure is capable of identifying a high variety of systems, the quadrotor is an ideal choice for testing its capabilities in a simulation environment, the rigid body dynamics of a drone contains high nonlinearities, particularly in its rotational dynamics.

This paper is structured as follows. Chapter 2 introduces the Koopman framework. Chapter 3 explains the LPV controller synthesis method. Chapter 4 presents a simulation study on a quadcopter system, while Chapter 5 discusses real-world implementation. Finally, Chapter 6 provides the conclusions.

## Chapter 2

# The Koopman Framework

An approach to overcome the difficulties that come with nonlinear systems is to embed them into linear models. One approach is the Koopman framework [11], which trades off complexity by representing nonlinear systems using a generally infinite-dimensional, but linear formulation. By applying a nonlinear state transformation through so-called observable functions, the states are projected into a higher dimensional space where their dynamic relation can be expressed as a linear mapping.

One method to construct this higher-dimensional representation involves building time-shifted data matrices to represent the evolution of the system. By selecting a finite set of observables, their parameters can be fitted to the data using a least squares approach [24][26]. However, these methods do not offer a good solution for the handling of systems with inputs. As shown in [13], the approximation error involved in this solution can be substantial.

Although originally developed for autonomous systems, a more recent extension allows the Koopman framework to represent input-affine dynamics in a bilinear form, where the input matrix is state dependent [8][9]. In [12], an *artificial neural network-based state space* (ANN-SS) representation is used to derive the higher-dimensional linear representation. As explained in Section 2.3.2, this approach takes care of the main difficulty of finding the observable functions, by estimating the parameters of a deep neural network to represent them.

The resulting Koopman model structure can be viewed as a *Linear Parameter Varying* (LPV) system, for which numerous control techniques have been developed [22][25].

In this Chapter, the Koopman framework is introduced, and its application for both autonomous and input-affine system representations. The data-driven identification method to estimate Koopman models by learning efficient observable functions, which is called Koopman-SUBNET, is discussed in Section 2.3.

### 2.1 Koopman representation of autonomous systems

Consider a nonlinear autonomous system in the state space form:

$$x_{k+1} = f(x_k) \tag{2.1}$$

where  $x_k : \mathbb{Z} \rightarrow \mathbb{R}^{n_x}$  is the state variable,  $k \in \mathbb{Z}$  is the discrete timestep, and  $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x}$  is a bounded nonlinear function. The behavior of the system can be represented in

Koopman-linear form, using the linear Koopman operator  $\mathcal{K} : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_z}$ , and  $\phi : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_z}$  which is the so-called observable function, satisfying

$$\phi \circ f(x_k) = \mathcal{K}\phi(x_k) \quad (2.2)$$

While often the existence of the Koopman operator requires the elevated state space to be infinite-dimensional, for practical purposes, an  $n_f$ -dimensional linear subspace  $\mathcal{F}_{n_f} \subset \mathcal{F}$  is considered, which can still give a good approximation of the system dynamics, even if it is not exactly Koopman invariant. The main difficulty of the Koopman framework is the choice of the lifting functions such that the resulting observables generate a Koopman invariant subspace. We introduce the finite dimensional lifted state  $z_k = \Phi(x_k)$ , where  $\Phi(x_k) = [\Phi_1(x_k) \ \cdots \ \Phi_n(x_k)]^\top$ . In the lifted form, equation (2.1) takes the form of a linear system:

$$z_{k+1} = Az_k \quad (2.3)$$

where  $z_k \in X_{\text{koop}} = \mathbb{R}^{n_z}$  is the lifted state vector, and  $A \in \mathcal{R}^{n_z \times n_z}$  is a matrix, the finite dimensional approximation of the Koopman operator.

From (2.2) it follows, that the existence condition of a finite-dimensional Koopman invariant subspace is

$$\Phi \circ f \in \text{span}\{\Phi\}. \quad (2.4)$$

The existence of the inverse transformation  $\Phi^\dagger(\Phi(x_k)) = x_k$  can be ensured by requiring the original states to be in the span of the observables. This enables us to reconstruct the original states  $x_k$ .

One difficulty with this representation is choosing the right initial conditions for the lifted state vector, which are given by the nonlinear constraint  $z_0 = \Phi(x_0)$ . Without the right initial conditions, the elevated state vector is not going to stay in the Koopman-invariant subspace, and the linear dynamics is not going to be a good representation of the system.

As an example following from [12], consider the following continuous-time nonlinear system:

$$x_{k+1,1} = ax_1 \quad (2.5)$$

$$x_{k+1,2} = bx_{k,2} - cx_{k,1}^2 \quad (2.6)$$

where  $a, b, c$  are constant parameters. The chosen observable functions are the following:

$$\Phi_1 = x_1$$

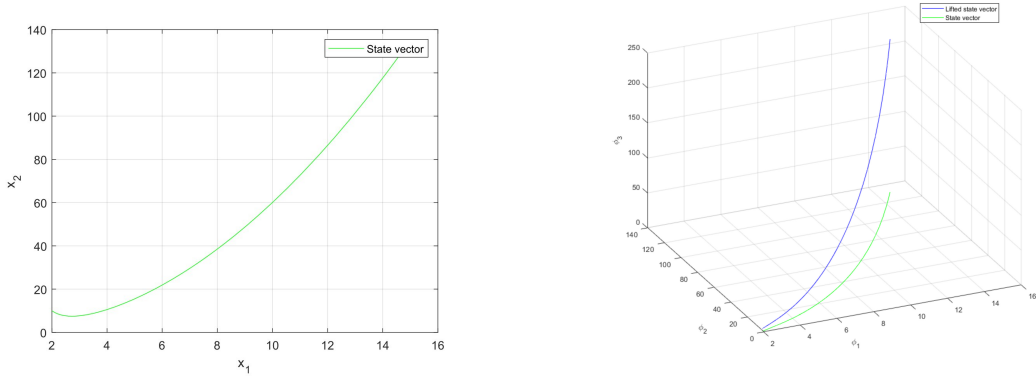
$$\Phi_2 = x_2$$

$$\Phi_3 = x_1^2$$

Substituting the observable functions, the dynamics of the system can be described as:

$$\Phi(x_{k+1}) = \begin{bmatrix} a & 0 & 0 \\ 0 & b & -c \\ 0 & 0 & a^2 \end{bmatrix} \Phi(x_k) = A\Phi(x_k) \quad (2.7)$$

As an example, setting the initial conditions to  $x_0 = [2 \ 10]^\top$ , and calculating the nonlinear constraint on the initial conditions of the lifted state space  $[\Phi_1 \ \Phi_2 \ \Phi_3]^\top = [x_1 \ x_2 \ x_1^2]^\top =$



(a) Trajectory of the state vector in the original state space. (b) Trajectory of the state vector in the original nonlinear, and in the lifted state space.

**Figure 2.1:** Lifting from 2D to 3D state space.

$\begin{bmatrix} 2 & 10 & 4 \end{bmatrix}^\top$ , and simulating the two systems, the evolution of the dynamics in the nonlinear, and Koopman linear representation are shown in Figure a 2.1.

The blue curve shows the evolution of the lifted state vector. In this example, the state variables of the non-linear state space form a subspace of the lifted space, and are calculated by projecting the Koopman linear trajectory onto the  $\Phi_1, \Phi_2$  plane, given by the green curve.

## 2.2 Koopman representation of systems with inputs

To extend the Koopman-representation to the non-autonomous case, we consider the class of input affine nonlinear systems:

$$x_{k+1} = f_d(x_k, u_k), \quad (2.8)$$

which we can decompose into autonomous and input driven parts as:

$$x_{k+1} = f(x_k) + g(x_k)u_k, \quad (2.9)$$

with a potentially nonlinear function  $g : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x \times n_u}$  and input  $u : \mathbb{Z} \rightarrow \mathbb{R}^{n_u}$ . The treatment of the inputs in the lifted form is a topic of debate with many different approaches present in the literature. It can be shown [10], that, in case of finite dimensional embedding or approximate representation, the required formulation to sufficiently capture the behavior of the system is the Koopman form:

$$\Phi(x_{k+1}) = A\Phi(x_k) + B(x_k, u_k)u_k. \quad (2.10)$$

This formulation is not linear in the input  $u$ , due to the matrix function  $B(x_k, u_k)$ , and it is also problematic for controller design, as the input matrix is a function of the input

to the system. However, given that the sampling rate of the discrete-time system is high enough, the following form can still be a good approximation of the dynamics.

$$\Phi(x_{k+1}) = A\Phi(x_k) + B_x(x_k)u_k = Az_k + B(z_k)u_k. \quad (2.11a)$$

It is shown in [5], that, in case number of observables go to infinity, (2.11) can fully capture the nonlinear dynamics.

When looking for the lifting functions, we are going to enforce, that the output function is in the span of the lifted states:

$$y_k = Cz_k, \quad (2.11b)$$

where  $C \in \mathbb{R}^{n_y \times n_z}$  is the constant output matrix.

The main advantage of using the Koopman model structure (2.11a) is that it can be viewed as a linear parameter varying (LPV) model, for which numerous control techniques have been developed[21].

In this work, we consider the specific case where full state measurements of a system in the form 2.9 are accessible, i.e.  $y_k = x_k$ . In this case  $\Theta_x = \mathbf{I}^{n_x \times n_x}$  is the observability map of (2.9). Let  $\Theta_z : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_z}$  be the composition of the lifting function and the observability map. Then  $\Theta_z = \Theta_x \circ \Phi = \Phi$ .

## 2.3 Deep-learning of Koopman surrogates

In this section, the neural network-based identification method is introduced. to learn Koopman surrogate models of the NL dynamics, which are useful for controller design.

### 2.3.1 Data generating system, and model parametrization

We consider the representation of the dynamics of a real physical system in the form a *continuous-time* (CT) nonlinear *state-space* (SS):

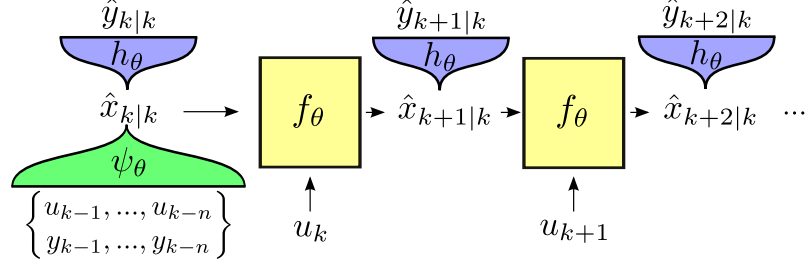
$$\dot{x}(t) = f_c(x(t), u(t)) \quad (2.12a)$$

where  $t \in \mathbb{R}$  is the continuous time,  $x(t) \in \mathbb{R}^{n_x}$  is the collection of the states of the system,  $u(t) \in \mathbb{R}^{n_u}$  is the input of the system, and  $f_c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$  is a nonlinear bounded function. Based on digital actuation and data acquisition, we assume that the input satisfies  $u(t) = u_k$  for all  $kT_s \leq t < (k+1)T_s$ , where  $T_s \in \mathbb{R}_+$  is the sampling time, corresponding to a *zero-order-hold* (ZOH) setting. Furthermore, full state measurements of the states are available in terms of

$$y_k = x_k + e_k \quad (2.12b)$$

where  $y_k \in \mathbb{R}^{n_y}$  is the measured output signal and  $e_k$  an i.i.d. white noise process representing measurement noise.

Under the ZOH setting, the dynamics represented by (2.12) can be also described directly in DT in the form:



**Figure 2.2:** The nonlinear SUBNET structure.

$$x_{k+1} = f(x_k, u_k), \quad (2.13a)$$

$$y_k = x_k + e_k, \quad (2.13b)$$

where  $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$  denotes the state-transition function in DT.

Using a data sequence  $\mathcal{D}_N = \{(y_k, u_k)\}_{k=1}^N$  measured by (2.12), the system can be identified directly in DT form:

$$\hat{x}_{k+1} = f_\theta(\hat{x}_k, u_k), \quad (2.14a)$$

$$\hat{y}_k = h_\theta(\hat{x}_k), \quad (2.14b)$$

where  $\hat{x}_k \in \mathbb{R}^{n_x}$  is the model state,  $\hat{y}_k \in \mathbb{R}^{n_y}$  is the model output, and  $f_\theta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$  with  $h_\theta : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$  are feedforward multi-layer artificial neural network. Each hidden layer is composed of  $m$  activation functions  $\phi : \mathbb{R} \rightarrow \mathbb{R}$  in the form of  $z_{i,j} = \phi(\sum_{l=1}^{m_{i-1}} \theta_{w,i,j,l} z_{i-1,l} + \theta_{b,i,j})$  where  $z_i = \text{col}(z_{i,1}, \dots, z_{i,m_i})$  is the latent variable representing the output of layer  $1 \leq i \leq q$ . Here,  $\text{col}(\cdot)$  denotes the composition of a column vector.

For  $f_\theta$  with  $q$  hidden-layers and linear input and output layers, this means  $f_\theta(\hat{x}_k, u_k) = \theta_{w,q+1} z_q(k) + \theta_{b,q+1}$  and  $z_0(k) = \text{col}(\hat{x}_k, u_k)$ . with  $\theta \in \mathbb{R}^{n_\theta}$  collecting the activation weights as model parameters,  $\tau \in \mathbb{R}_+$  is a normalization parameter. Under these considerations, (2.14) represents recurrent neural networks, also called as ANN-SS models. The nonlinear SUBNET structure can be seen in Figure 2.2

In order to identify models in Koopman-form (2.11), the same data sequence,  $\mathcal{D}_N$  can be used to estimate the parameters of

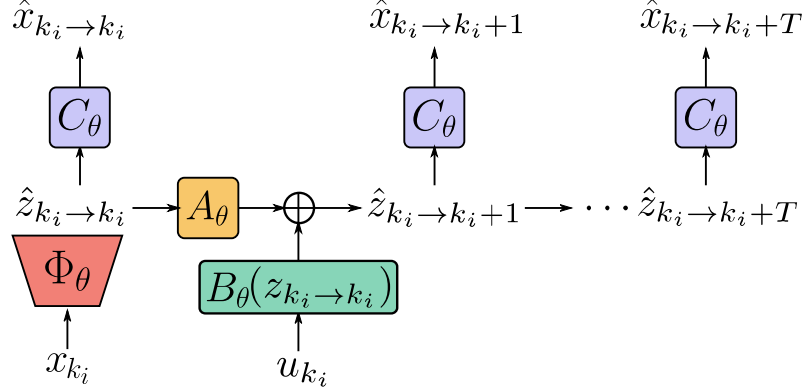
$$z_{k+1} = A_\theta z_k + B_\theta(z_k) u_k, \quad (2.15)$$

$$\hat{x}_k = C_\theta z_k, \quad (2.16)$$

which is a restructured form of (2.14). Here,  $\hat{z}_k \in \mathbb{R}^{n_z}$  is the state of ANN-SS model corresponding to the elevated Koopman state vector,  $\hat{x}_k \in \mathbb{R}^{n_x}$  is the model output, which are the physical states of the modeled system.  $B_\theta(z_k) : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_z \times n_u}$  is a feedforward multi-layer artificial neural network.

The linear Koopman operator is implemented as  $A_\theta : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_z}$  and the constant output matrix as  $C_\theta : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_x}$ , which are both linear layers, with only  $\theta_{w,i,j}$  as parameters.

The Koopman SUBNET structure can be seen in Figure 2.3.



**Figure 2.3:** The Koopman SUBNET structure.

### 2.3.2 The SUBNET method

In the literature, many approaches have been introduced to identify ANN-SS models, e.g., [19, 7]. One recent approach that provides computationally efficient estimation of (2.14) under statistical consistency guarantees is the SUBNET approach [2]. Conventional one-step-ahead prediction error criteria require the simulated responses of (2.13), however, this becomes computationally infeasible for large datasets and is challenging to optimize due to gradient instabilities. The SUBNET method uses a  $T$ -step ahead prediction cost over multiple subsections of the data, which is tractable and numerically attractive to optimize compared to the conventional one-step-ahead prediction error cost. This is formalized for DT as follows:

$$V_{\mathcal{D}_N}^{\text{enc}}(\theta) = \frac{1}{N_{\text{sec}}} \sum_{k \in \mathcal{I}_{N_{\text{sec}}}} \frac{1}{T} \sum_{l=0}^{T-1} \|y_{k+l} - \hat{y}_{k+l|k}\|_2^2, \quad (2.17a)$$

$$\hat{x}_{k+l+1|k} = f_\theta(\hat{x}_{k+l|k}, u_{k+l}), \quad (2.17b)$$

$$\hat{y}_{k+l|k} = h_\theta(\hat{x}_{k+l|k}) \quad (2.17c)$$

where  $\mathcal{I}_{N_{\text{sec}}} \subseteq \{1, \dots, N - T + 1\}$  s.t.  $|\mathcal{I}_{N_{\text{sec}}}| = N_{\text{sec}}$  is the number of sections, and its elements are the starting time of the sections, and the pipe ( $|$ ) notation is used to denote time indexing inside of the section in terms of (current index|start index). As prediction over the sections can be computed in parallel, this enables the use of powerful batch optimization algorithms such as Adam [15] for training.

Furthermore, SUBNET co-estimates an “encoder” function  $\psi_\theta$  together with  $f_\theta$  and  $h_\theta$ , which provides a direct estimate of the initial state  $\hat{x}_{k|k}$  for each section in (2.17) based on past input and output data, i.e.,

$$\hat{x}_{k|k} = \psi_\theta(u_{k-n}^{k-1}, y_{k-n}^{k-1}) \quad (2.17d)$$

where  $u_k^{k+t} = [u_k^\top \dots u_{k+t}^\top]^\top$  for  $t \geq 0$  and  $y_k^{k+t}$  is defined similarly. In fact,  $\psi_\theta$  corresponds to learning a reconstructability map of the state from past IO samples, where  $n$  needs to be chosen at least the lag ( $n_x$  in the SISO case) of the NL system, see [3] for details.

In case of the Koopman-SUBNET, the signals  $y_k$  and  $x_k$  are replaced by  $x_k$  and  $z_k$ , the model Equations (2.17b) and (2.17c) by (2.15), and the state encoder equation (2.17d) by  $z_k = \Phi_\theta(y_k)$ .

## Chapter 3

# Koopman modeling of quadrotor dynamics

This chapter discusses the application of the identification procedure on the Crazyflie 2.0 quadcopter within a simulation environment.

A major challenge in data-based identification, especially with black-box methods, is the reliance on high-quality datasets, requiring careful planning in data collection. During the optimization of neural networks, their parameters are adjusted to fit the collected data, ensuring accurate local estimations. The interpolation capabilities of the neural network highly depends on good-quality data and proper hyperparameter selection, without them overfitting or underfitting of the network parameters can occur, resulting in bad performing model. As the quadrotor is an unstable system, the use of a nonlinear controller, and careful trajectory planning was necessary to collect data of sufficient quality.

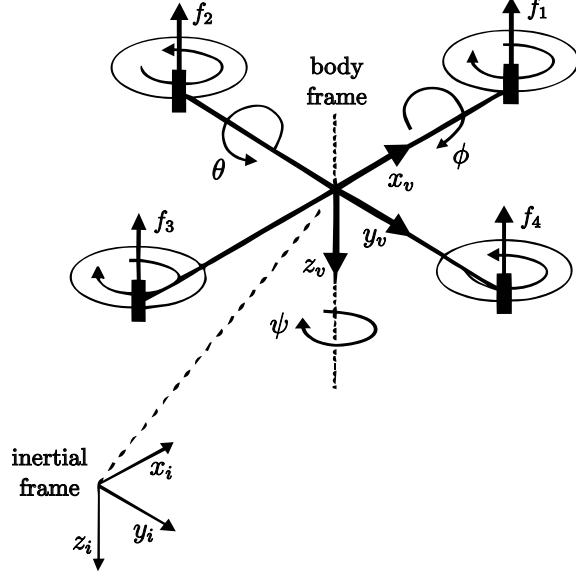
This chapter is structured as follows. The first section described a simple model of a quadcopter. Then the method of collecting data in a closed loop environment is explained, along with the used trajectories for the training and test datasets. In Section 3.3 the described identification procedures is applied to the quadrotor, then the results are presented.

### 3.1 Mathematical model of a quadcopter

This model is based on the following coordinate frames: inertial frame  $F_i$  with *north-east-down* (NED) coordinates, vehicle frame  $F_v$ , and body frame  $F_b$ , which is fixed to the vehicle. The coordinate frames used in the modeling are shown in Figure 3.2. The



**Figure 3.1:** The Crazyflie 2.0 miniature quadrotor.



**Figure 3.2:** The coordinate systems used for modeling the quadcopter. The transformation is a translation between the inertial-vehicle frames, and rotation between the vehicle-body frames.

transformation from  $F_i$  to  $F_v$  is a translation and from  $F_v$  to  $F_b$  a rotation, because the axes of  $F_v$  and  $F_i$  are considered parallel. The translational dynamics of the quadcopter are characterized by

$$m\ddot{\mathbf{r}} = R_b^v \begin{bmatrix} 0 \\ 0 \\ -F \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}, \quad (3.1a)$$

where  $\mathbf{r} = [x \ y \ z]^\top$  is the position corresponding to the center of mass of the drone,  $\dot{\mathbf{r}} = [v_x \ v_y \ v_z]^\top$  is the lateral velocity in the inertial frame,  $m$  is the mass,  $F$  is the collective thrust of the propellers, and  $g$  is the gravitational acceleration.  $R_b^v \in \text{SO}(3)$  is the rotation matrix between the vehicle and body frames, where  $\text{SO}(3)$  denotes the 3D special orthogonal group. The rotational dynamics are described based on Euler's equations as

$$\dot{R}_b^v = R_b^v \hat{\boldsymbol{\omega}}_{bv} \quad (3.1b)$$

$$\hat{\boldsymbol{\omega}}_{bv} = J_b^{-1} (\boldsymbol{\tau} - \boldsymbol{\omega}_{bv} \times J_b \boldsymbol{\omega}_{bv}) \quad (3.1c)$$

where  $\boldsymbol{\omega}_{bv} = [p \ q \ r]^\top$  denotes the angular velocity of the body frame in world frame,  $J_b$  is the inertia matrix, and  $\boldsymbol{\tau} = [\tau_x \ \tau_y \ \tau_z]^\top$  is the torque vector in  $F_b$  produced by the propellers. The notation  $\hat{\cdot}$  stands for the projection:  $\mathbb{R}^3 \rightarrow \text{SO}(3)$  ensuring that  $\hat{x}y = x \times y$  for all  $x, y \in \mathbb{R}^3$ , where  $\times$  corresponds to the vector product.

To achieve a lower dimensional state space representation of (3.1b), the rotation matrix  $R_b^v$  is mapped to Euler angles: roll, pitch, and yaw denoted as  $\phi$ ,  $\theta$  and  $\psi$  respectively.

To reproduce the dynamics of a Crazyflie 2.1, the parameters are set to:  $J_x = J_y = 1.4 \times 10^{-5} \text{ kg} \cdot \text{m}^2$ ,  $J_z = 2.17 \times 10^{-5} \text{ kg} \cdot \text{m}^2$ , corresponding to the diagonal values in the diagonal inertia matrix, and the off-diagonal values are 0,  $m = 0.027 \text{ kg}$ , and  $g = 9.8 \text{ m/s}^2$ .

## 3.2 Data collection

Data generation is conducted using the simulation model (3.1). A simulation-based study was chosen to enable an in-depth analysis of all aspects of model representation, identification process, and control performance. We used a rigid body model of the Crazyflie 2.0 quadrotor, with a sampling frequency of  $f_s = 48$  selected to achieve realistic system behavior.

Data generation has been done using the simulation model (3.1). We used a rigid body model of the Crazyflie 2.0 quadrotor in the environment, and a sampling frequency of  $f_s = 48$  Hz has been selected to achieve a realistic behavior of the system.

As the dynamics of the quadcopter is unstable, it was necessary to collect data in a closed-loop setting. Initial attempts at data collection were made using a simple cascade PID controller, where inner and outer loops stabilized the translational and rotational dynamics. However, this controller only performed well when the quadrotor was operated close to the hovering equilibrium point in the state space, which is linear and which corresponds to slower maneuvers. This was insufficient for us, as the effectiveness of the Koopman framework can only be shown by identifying a nonlinear system. Consequently, the non-linear geometric controller was used to drive the system, described in [20]. This controller is a nonlinear feedback controller, and it also uses feedforward input signals corresponding to a reference trajectory, making the reference tracking more precise. The feedforward input calculation takes advantage of the differential flatness property of the system, and requires that the reference signals be four times differentiable. We designed experiments for data collection by describing reference paths that the controller can follow.

### 3.2.1 Experiment design

The trajectory design needs to be done after careful consideration, as the quality of the training data greatly influences the performance of the obtained neural net model. To discover parts of the state space where the nonlinear behavior of the quadcopter is dominant, we designed planned flight paths that involve large values of roll and pitch angles, as for smaller angles the dynamics are close to linear.

Due to the differential flatness of the system [20], and because the system has four degrees of freedom, it is necessary to define the time function for four data channels in the path planning step. In the following sections, the reference states  $x, y$  and  $z$  for the Training, Validation, and Test data sets are given. In these datasets, for the fourth prescribed channel, the yaw angle  $\psi$  has been kept zero during all reference trajectories.

**Training Data:** For training data, we created reference paths that describe motions in which the quadrotor travels between the origin of a selected inertial frame and points located on the surface of a sphere centered at that origin. These points are parameterized using spherical coordinates, defined by a radius  $r$  and angles  $\phi$  and  $\theta$ . During each flight, the quadrotor performs several rapid back-and-forth movements between the origin and the generated points on the surface of the sphere.

$$\begin{bmatrix} x_{\text{ref}}(\rho) & y_{\text{ref}}(\rho) & z_{\text{ref}}(\rho) \end{bmatrix}^T = R_\theta(\rho)R_\phi(\rho)e_x s(\rho), \quad (3.2a)$$

$$s(\rho) = r_d \sin(2\pi f_r \rho - \frac{\pi}{2}) + r_d, \quad (3.2b)$$

where  $R_\theta(\rho)$  and  $R_\phi(\rho)$  are  $\text{SO}(3)$  matrices corresponding to rotation around the  $e_z$  and  $e_y$  axes respectively. The rotation angles are generated by  $\phi(\rho) = 0.3\pi \sin(2\pi f_\phi t)$ , and

$\theta(\rho) = 2\pi f_\theta \rho$ , while  $\psi_{\text{ref}}(\rho)$  has been kept 0. Here  $r_d$ ,  $\omega_r$ ,  $f_\phi$  and  $f_\theta$  are the trajectory parameters. Multiple flights have been recorded under the choices of  $r_d$ ,  $f_\phi$  and  $f_\theta$  detailed in Table 3.1, with  $r_d$  expressed in meters, and the angular velocity values expressed in Hz.

In order to start the drone from a hovering state, the speed of the movements must gradually increase from zero, which can be achieved by choosing  $\rho$  as:

$$\rho(t) = \begin{cases} t^2 & \text{if } t \leq T \\ t + T^2 - T & \text{if } t > T \end{cases} \quad (3.3)$$

As explained in [20], in order to take advantage of the differential flatness property the signals for the reference channels need to be four times differentiable. Therefore, splines were fitted to the reference, resulting in slight variations in the trajectories, but allowing for the necessary reference state vector and feedforward input calculations.

Furthermore, excitation signals were injected into the system inputs, as

$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} K_T + v_T \\ K_{\tau_x} + v_{\tau} \\ K_{\tau_y} + v_{\tau} \\ K_{\tau_z} + v_{\tau} \end{bmatrix} \quad (3.4)$$

where  $K$  denotes signals computed by the geometric controller, while  $v$  were created by sampling from normal distributions  $\mathcal{N}_T(0, \sigma_T)$  and  $\mathcal{N}_\tau(0, \sigma_\tau)$ , with  $\sigma_F = 0.022 \text{ N}$  for the collective thrust and  $\sigma_\tau = \begin{bmatrix} 2 & 2 & 0.2 \end{bmatrix}^\top \cdot 10^{-4} \text{ N}\cdot\text{m}$  for the torque input. These values were set after some trial and error attempts to excite the system as much as possible without making the closed-loop unstable. As measurement noise, i.i.d. Gaussian white noise has been added to the recorded states of the system. The variance of this noise has been chosen to achieve an aggregated SNR of 25 dB and 15 dB.

**Validation data:** After transforming the collected data into subsections according to Section 2.3.2, 20% of the data was selected to be used as validation data to enable early stopping during optimization.

**Test data:** Flight trajectories with

$$\begin{bmatrix} x_{\text{ref}}(t) \\ y_{\text{ref}}(t) \\ z_{\text{ref}}(t) \end{bmatrix} = \begin{bmatrix} -A \cos(2\pi f_r t) + A - vt \\ -A \cos(2\pi f_r t) + A + vt \\ -B \cos(2\pi f_r t) + B \end{bmatrix}, \quad (3.5)$$

have been generated with  $v = 0.05$ , defining the translational velocity perpendicular to the sinusoidal movement,  $A$  and  $B$  characterizing the amplitude, and  $f_r$  being the frequency of the oscillations, all specified in Table 3.2.

The described paths for the training and test datasets can be viewed in Figure 3.3.

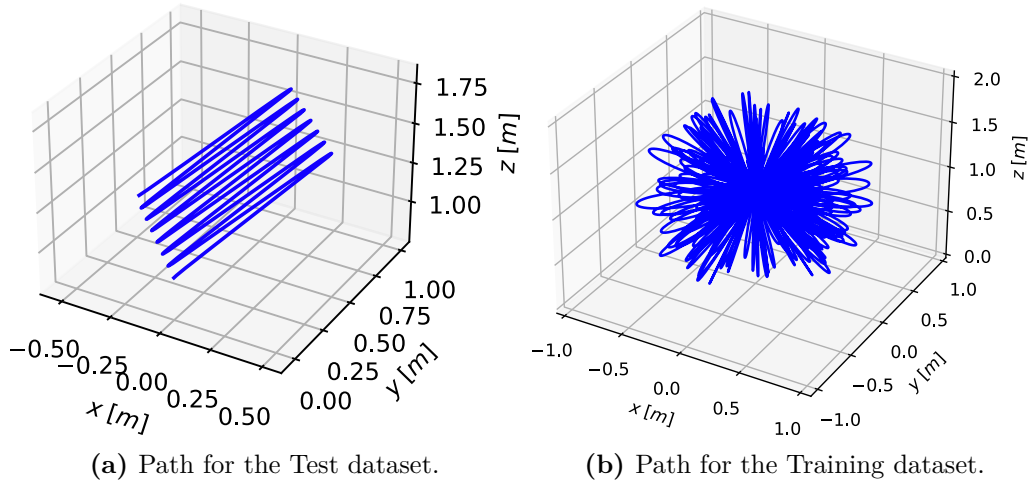
### 3.2.2 Data quality

The datasets generated in the simulation environment are shown in Figure 3.4. It is visible that the system is excited well, as the data covers hyperspherical areas, and the validation set covers the same region as the training set. Note that the quadrotor achieves up to 60deg of tilt during its flights, indicating rapid maneuvers. The test data is chosen to

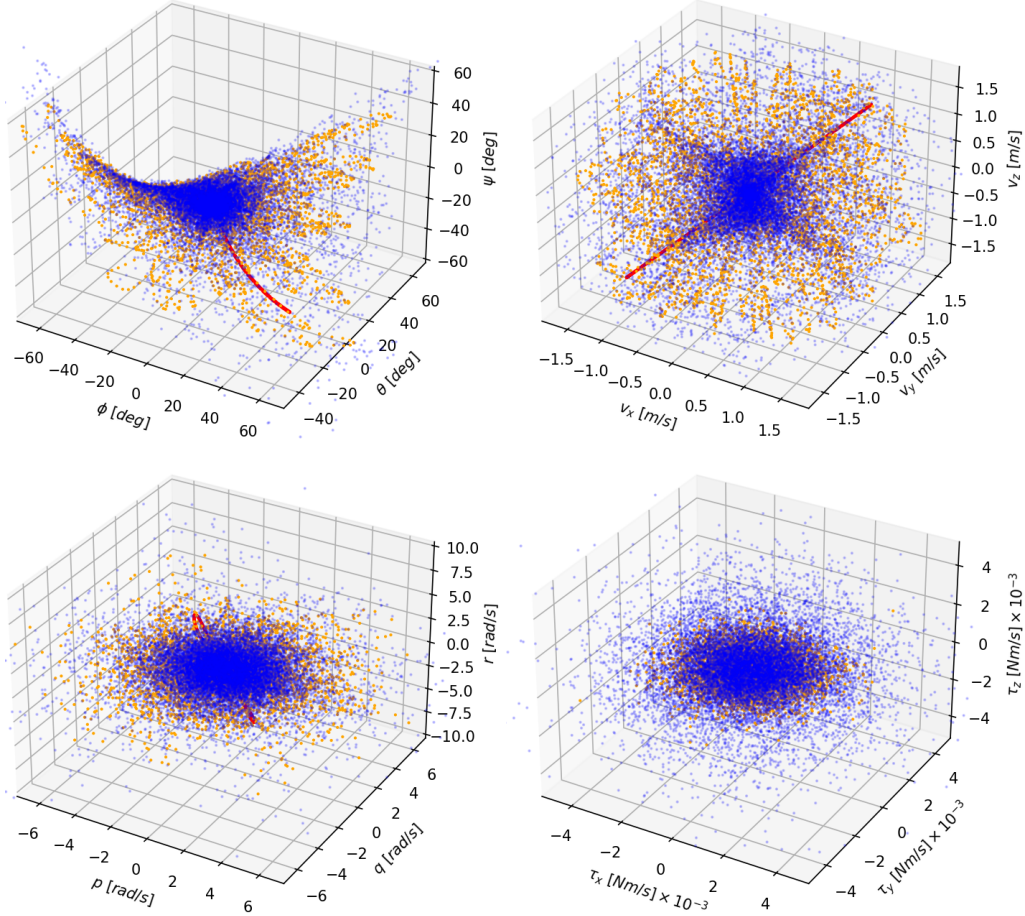
| #  | $f_r$ | $r_d$ | $f_\phi$ | $f_\theta$ |
|----|-------|-------|----------|------------|
| 1  | 0.20  | 0.5   | 0.0092   | 0.12       |
| 2  | 0.25  | 0.45  | 0.11     | -0.95      |
| 3  | 0.33  | 0.45  | 0.012    | -0.095     |
| 4  | 0.35  | 0.5   | 0.092    | 0.12       |
| 5  | 0.45  | 0.45  | 0.010    | -0.095     |
| 6  | 0.45  | 0.5   | 0.0092   | 0.12       |
| 7  | 0.55  | 0.5   | 0.0092   | 0.12       |
| 8  | 0.65  | 0.5   | 0.090    | 0.12       |
| 9  | 0.66  | 0.5   | 0.092    | 0.12       |
| 10 | 0.7   | 0.6   | 0.0087   | 0.12       |

| #  | $f_r$ | $r_d$ | $f_\phi$ | $f_\theta$ |
|----|-------|-------|----------|------------|
| 11 | 0.71  | 0.6   | 0.0088   | 0.12       |
| 12 | 0.74  | 0.55  | 0.096    | 0.12       |
| 13 | 0.75  | 0.55  | 0.0089   | 0.12       |
| 14 | 0.78  | 0.45  | 0.01     | 0.12       |
| 15 | 0.8   | 0.5   | 0.0087   | 0.12       |
| 16 | 0.84  | 0.45  | 0.0085   | 0.12       |
| 17 | 0.841 | 0.45  | 0.010    | 0.085      |
| 18 | 0.842 | 0.45  | 0.011    | 0.085      |
| 19 | 0.844 | 0.45  | 0.011    | -0.095     |

**Table 3.1:** Parameters for the Training paths, each executed for 100 seconds.



**Figure 3.3:** Paths used for data collection.



**Figure 3.4:** Visualization of gathered data. Training, validation, and test sets are shown in blue, orange, and red.

| # | $A$  | $B$ | $f_r$ |
|---|------|-----|-------|
| 1 | 0.25 | 0.3 | 0.7   |

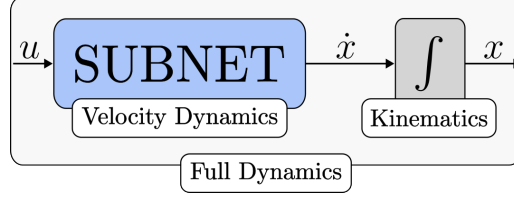
**Table 3.2:** Parameters for the test path, executed for 10 seconds.

evaluate the performance of the model in parts of the state and input space less densely populated with training data points.

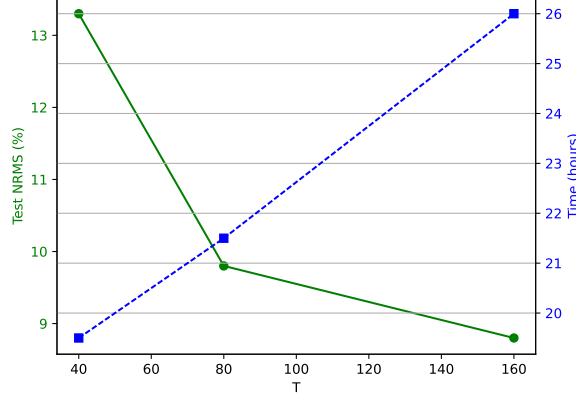
### 3.3 Identification results

For dynamical systems, some components can often be modeled accurately using first principles. In the case of a quadcopter, the kinematics can be precisely described in this way, and by separating the velocity dynamics, we can simplify the data-based modeling process. Applying this concept we only identified the states of the quadcopter corresponding to its translational velocity, orientation and rotational velocity, and integrate the position states from the translational velocities when testing the identified model. We denote the corresponding spaces as  $X_{\text{vel}} \subset \mathbb{R}^9$  and  $X_{\text{full}} \subset \mathbb{R}^{12}$ .

We validated the trained models by comparing their 160-step long open-loop simulation to the recorded data for both the Koopman-SUBNET and general nonlinear SUBNET structures. We evaluated the errors of all possible subsections in the test dataset, and



**Figure 3.5:** Separation of the kinematics relations from the rest.



**Figure 3.6:** Effects of different prediction horizons  $T$  on Test NRMS and training time.

computed the average normed root mean square error (NRMS) of the simulations. NRMS can be calculated as follows:

$$\text{NRMS} = \frac{1}{M} \sum_{m=1}^M \frac{\sqrt{\frac{1}{N} \sum_{k=1}^N (y_k^m - \hat{y}_k^m)^2}}{\sigma_{y_m}}, \quad (3.6)$$

Here  $m$  denotes the data channel, and  $\sigma_{y_m}$  the standard deviation of channel  $m$ .  $N$  is the section length that was set to  $N = 160$  for the calculation of the test errors.

The hyperparameters used for the models and the optimization process can be seen in Tables 3.4 and 3.3. In the table  $hl_{(\cdot)}$  and  $n_{(\cdot)}$  denote the number of hidden layers and neurons used in a given ANN. The hyperparameter selection for the number of hidden layers, neurons per layer and type of activation function come from the original work on SUBNET [2]. We found through trial and error, that these values result in the best performing models. Proper selection of the prediction horizon  $T$  and state dimension  $n_x$  proved to be more challenging. As  $T$  increases, the optimization time increases. However, to maintain precision in open-loop simulations over an extended period, a larger value of  $T$  is required. Figure 3.6 illustrates the trade-off between precision and training time for different values of  $T$ . In our case, a prediction horizon of 80 corresponds to a simulation duration of 0.8 seconds, which is adequate for an agile aerial vehicle. Choosing the right value for  $n_x$  is crucial, as a small state dimension would restrict the Koopman representation too much, and the model could not represent the system well in the linear form. The danger of choosing a high dimension is the easier overfitting of the neural network, as it increases the number of parameters in  $A_\theta$ ,  $B_\theta(\cdot)$ , and  $C_\theta$ . We selected a state dimension of  $n_x = 40$  since the performance of the network was optimal at this value, as shown in Table 3.5.

As discussed in Chapter 2, the dependence of the matrix  $B_\theta(\cdot)$  on the input  $u_k$  is theoretically necessary in the discrete-time (DT) case. Without this dependence, the system

can only be approximated by the Koopman representation. We also tested the use of a constant  $B$  matrix, as this fully linear structure is best for control purposes.

Table 3.6 presents the NRMS errors for the trained models. As expected, the nonlinear SUBNET achieves lower errors due to its more general modeling capability. However, the Koopman SUBNET still achieves good performance, with NRMS errors only slightly exceeding those of the nonlinear SUBNET. We also tested how measurement noise on the training dataset affects the performance of the trained SUBNETs. It can be seen, that the errors increase slightly, but even when added noise reaches 15 dB signal-to-noise ratio, the performance degradation is slight, with only a 1.1% NRMS error increase. Our tests showed that the dependence of  $B$  on the input  $u$  results in only a slight increase in precision, and choosing  $B$  to be constant made it impossible for the network to learn the behavior of the system.

As described in Chapter 2 dependence of the matrix  $B_\theta(\cdot)$  on the input  $u_k$  is needed in the DT case, without the system can only be approximated by the Koopman-representation. Our test showed, that including it only results in slight increase in precision, as can be seen in Table 3.6. Choosing the matrix to be constant however resulted in the network not being able to learn the behaviour of the system.

Figures 3.7 and 3.7 display the model errors relative to the recorded dataset. Once again, it is evident that the Nonlinear SUBNET model more effectively captures the system dynamics. Since any velocity errors accumulate when integrated, both the Nonlinear and Koopman SUBNET show inaccuracy in position by the end of the open-loop simulation, with the Koopman SUBNETs being notably less accurate. However, the errors in the remaining states remain low throughout the simulation.

| Model     | $hl_\psi$ | $n_\psi$ | $hl_f$          | $n_f$          | $n_x$ | $n$ | $\phi$  |
|-----------|-----------|----------|-----------------|----------------|-------|-----|---------|
| Nonlinear | 2         | 64       | 2               | 64             | 15    | 1   | sigmoid |
| Model     | $hl_\Phi$ | $n_\Phi$ | $hl_{B_\theta}$ | $n_{B_\theta}$ | $n_z$ | $n$ | $\phi$  |
| Koopman   | 2         | 64       | 2               | 64             | 40    | 1   | sigmoid |

**Table 3.3:** Hyperparameters of the model structures.

| Model     | $\alpha$  | $N_{\text{batch}}$ | $T$ | $N$     | Epochs | Time [h] |
|-----------|-----------|--------------------|-----|---------|--------|----------|
| Nonlinear | $10^{-4}$ | 256                | 80  | 138,720 | 1500   | 12       |
| Koopman   |           |                    | 80  |         |        | 18       |

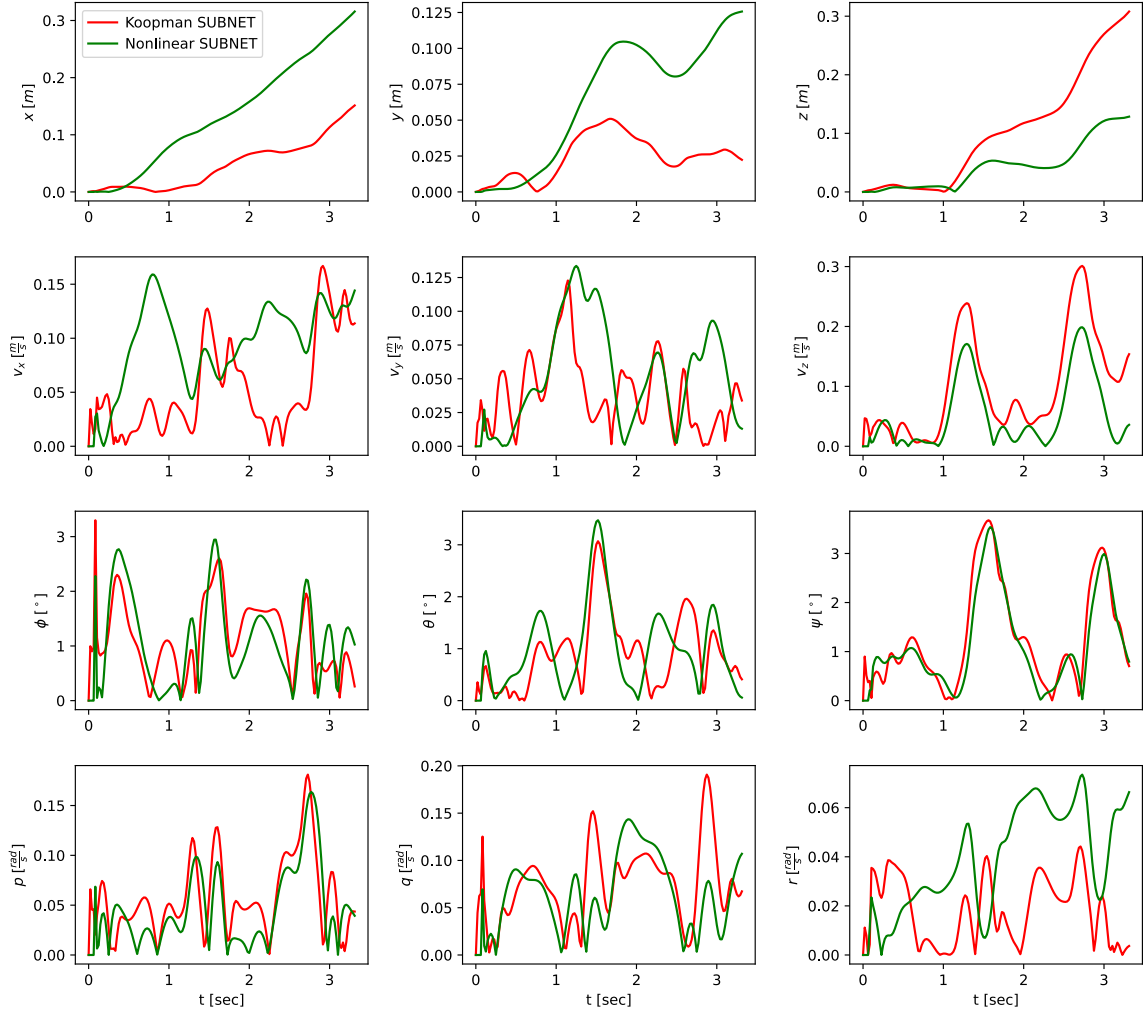
**Table 3.4:** Hyper parameters for the training of neural networks.

| $n_x$     | 20    | 40   | 60   |
|-----------|-------|------|------|
| Test NRMS | 11.8% | 7.8% | 8.0% |

**Table 3.5:** Koopman-SUBNET accuracy for different choices of  $n_x$ .

| Model                                   | Test NRMS |
|-----------------------------------------|-----------|
| Koopman (noise free)                    | 7.8%      |
| Koopman (25 dB noise)                   | 8.0%      |
| Koopman (15 dB noise)                   | 9.0%      |
| Koopman constant $B$ (noise free)       | 81%       |
| Koopman $B$ depends on $u$ (noise free) | 7.4%      |
| Nonlinear SUBNET (noise free)           | 5.5%      |

**Table 3.6:** NRMS test errors of different models.



**Figure 3.7:** NRMS errors of the 160-step open-loop simulations using the test dataset.

## Chapter 4

# Controller design

During the identification we decaupled the kinematics part of the dynamics, complicating position reference tracking. This can be managed by including the position states in the state-space representation. However, this is made difficult by the nonlinear observable function  $\Phi$ , which maps the physical state space into the Koopman-linear representation. As a solution we implemented a cascade controller structure, where the inner loop stabilizes the identified velocity dynamics, while the outer loop the kinematics.

In the model, we acquired the input enters the system affinely, through a parameter-varying matrix, resulting in a Linear Parameter Varying (LPV) system, which motivates the design of an LPV controller for the inner loop.

LPV controller design is a powerful tool, which is widely used to develop high performance controllers for nonlinear systems. [21] The idea behind this method, is to develop a parameter-varying stabilizing feedback gain, which satisfies the given performance specifications. The varying parameters of the controller are functions of measured/observable variables of the system as scheduling variables denoted as  $p \in \mathcal{P}$ , which indicate the locally linear region where the system operates. This LPV representation allows us to use established LTI control methods, like  $\mathcal{H}_2/\mathcal{H}_\infty$  optimal control and mixed sensitivity shaping, which provide strong performance guarantees.

Different LPV synthesis methods exist depending on scheduling dependency of the system. For affine or rational dependencies where  $\mathcal{P}$  forms a convex set, global controllers can be designed, offering stability and performance over the entire scheduling space [16], [23]. For more complex dependencies, or when defined only at specific points, gridded LPV synthesis can be applied [6]. This approach, while guaranteeing stability only at grid points, can manage intricate dependencies, such as those described by neural networks in our case, if the grid is sufficiently dense.

In the first section of this chapter the LPV-framework is introduced, along with details specific to our identified Koopman-SUBNET model. Section 4.2 explains the integration of the outer loop PI controller within the full controller structure.

### 4.1 LPV controller design for Koopman models

For the design of LPV controllers, a few theoretical concepts are needed, which will be introduced in this section. In order to systematically shape the performance of our various controllers, we introduce a general framework with the generalized plant concept and

formulate the required conditions for stability, with performance tuning handled through sensitivity shaping.

Given the complexity of the dependency of the dynamics on the scheduling variable, a grid-based synthesis approach will be applied. However, gridding the space of the scheduling variable is challenging due to its high dimensionality, therefore, we will use an irregular grid. A convex controller synthesis method is going to be used to satisfy performance specifications and ensure closed-loop stability at each grid point. Additionally, an interpolation method will be introduced to evaluate feedback gains between grid points for a globally functional controller.

#### 4.1.1 Control structure and shaping

In order to introduce the generalized plant concept we are going to categorize the signals in a general way, which can be done for most dynamical systems. The four categories that can be established include:

- measurement output vector  $y \in \mathbb{R}^{n_y}$
- control input vector  $u \in \mathbb{R}^{n_u}$ ,
- controlled output vector  $\tilde{z} \in \mathbb{R}^{n_{\tilde{z}}}$ , This usually includes the control input and the reference signal error, denoted as  $u$ ,  $e$  respectively.
- generalized disturbance vector  $w \in \mathbb{R}^{n_w}$ . This usually includes reference signal, system disturbances, and measurement noise, denoted as  $r$ ,  $d$  and  $n$  respectively.

As previously mentioned, this work assumes full state measurements  $y_k = x_k$ . This is a reasonable assumption given that the onboard microcontrollers often come equipped with integrated IMUs. These IMUs typically measure rotational velocities, translational acceleration, and may be complemented by magnetometers and GPS for position and heading angle measurements. By using these measurements and their integrals, we can estimate the state corresponding to the model in (3.1). The outputs from the high dimensional Koopman-model - which are the physical states - are available through the matrix  $C_\theta$ . For the sake of brevity, the subscript  $(\cdot)_\theta$ , which corresponds to the optimized parameters of the neural network, will be omitted from now on.

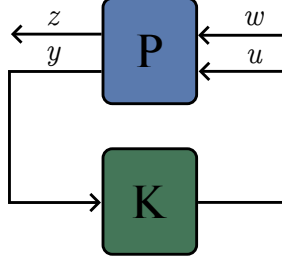
Expanding the Koopman-linear state-space description, our plant  $P$  takes the form:

$$\begin{aligned} z_{k+1} &= Az_k + B_w(p_k)w_k + B_u(p_k)u_k \\ \tilde{z}_k &= C_{\tilde{z}}z_k + D_{\tilde{z}w}w_k + D_{\tilde{z}u}u_k \\ x_k &= C_xz_k \end{aligned} \tag{4.1}$$

In matrix form, this can be written as:

$$\begin{bmatrix} z_{k+1} \\ \tilde{z}_k \\ x_k \end{bmatrix} = \begin{bmatrix} A & B_w(p_k) & B_u(p_k) \\ C_{\tilde{z}} & D_{\tilde{z}w} & D_{\tilde{z}u} \\ C_x & 0 & 0 \end{bmatrix} \begin{bmatrix} z_k \\ w_k \\ u_k \end{bmatrix}.$$

where  $p \in \mathcal{P}$  is the scheduling variable, which in our case is equal to the lifted state vector  $p_k = z_k$ .  $A \in \mathbb{R}^{n_x \times n_x}$  is the state transition matrix,  $B_w : \mathbb{P} \rightarrow \mathbb{R}^{n_x \times n_w}$  and  $B_u : \mathcal{P} \rightarrow \mathbb{R}^{n_x \times n_u}$  are the input matrices for generalized disturbance and control inputs,  $C_{\tilde{z}} \in \mathbb{R}^{n_{\tilde{z}} \times n_x}$



**Figure 4.1:** PK form

and  $C_x \in \mathbb{R}^{n_x \times n_z}$  are the output matrices for controlled output and measurement signals, respectively, and  $D_{\tilde{z}w} \in \mathbb{R}^{n_{\tilde{z}} \times n_w}$ ,  $D_{\tilde{z}u} \in \mathbb{R}^{n_{\tilde{z}} \times n_u}$  are the feedthrough matrices. Given that the pair of  $A$  and  $B_u$  is stabilizable and the pair of  $A$  and  $C_x$  is detectable (i.e. there exists a controller  $K(p_k)$ , which stabilizes the system)  $P$  is a generalized plant.

In the following subsections, a performance shaping method, and the  $\ell_2$  gain performance measure is going to be introduced, which will be used to formulate the controller synthesis problem.

#### 4.1.2 Signal shaping

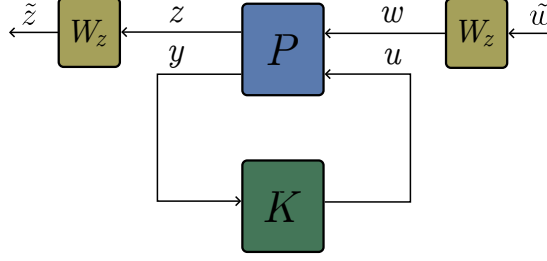
Our objective is to "minimize" the transfer from the exogenous inputs  $w$  to the control output signal  $\tilde{z}$ . The exogenous inputs consist of the disturbance, noise and reference signals  $w = [d \ n \ r]^\top$ , and the control outputs consist of the error and control input signals  $\tilde{z} = [e \ u]^\top$ .

The optimization algorithm, which synthesizes the controller (detailed later in Section 4.1.4) minimizes the chosen induced norm of the transfer matrix from  $w$  to  $\tilde{z}$ , and so a natural way of influencing the performance is by altering the shape of the spectrum of the various transfers according to our needs with the use of filters.

- In terms of the output channels, our aim is the suppression of signal magnitude in the lower frequency range, where the reference signal is active and we also intend the controller output channels to have high amplitude in this frequency range. This can be achieved by introducing filters that implement the inverse of the desired frequency response shapes.
- In terms of the input channels, apriori information about the spectrum of the disturbance and noise signals helps in improving the performance of the controller. The frequency range, where high-amplitude signals are expected needs more restrictive treatment by filters - as they need to be suppressed -, elsewhere looser specification is sufficient. This process can be thought of as the normalization of the signals on the basis of their expected spectrum.

The availability of no apriori information was assumed in this work, consequently,  $W_w$  is set to be an identity matrix, the choices for the output filters will be provided in Chapter 5. The use of low-order filters are recommended, as their states are incorporated into the full state-space representation; higher-order filters would expand the state space more, complicating the synthesis process.

Figure 4.2 shows how the filters are integrated into the the PK-form.



**Figure 4.2:** Sensitivity shaping using filters.

#### 4.1.2.1 Performance objectives

The  $\ell_2$  norm is often used in robust controller synthesis, as it is a measure of the worst case (i.e. biggest) energy gain of the system.

For an LPV system  $N(p)$ , that is proper and stable for all  $p \in \mathcal{P}_{\text{sched}}$ , the  $\ell_2$  induced gain is defined as:

$$\|N(p)\|_2 = \sup_{\substack{0 < \|w\|_2 < \infty, \\ p \in \mathcal{P}_{\text{sched}}}} \frac{\|N(p)w\|_2}{\|w\|_2} = \sup_{\|w\|_2=1} \|N(p)w\|_2, \quad (4.2)$$

where  $\|\cdot\|_2$  denotes the  $\ell_2$  norm.

A minimization objective of this performance measure pushes down the value of the biggest singular value of the transfer matrix along the imaginary frequency axis, which corresponds to the worst-case frequency and input direction.

#### 4.1.3 Gridding the scheduling space

The dependency of the system dynamics on the scheduling variables is highly complex, as  $B_\theta$  is realized by a deep neural network. This makes global LPV synthesis infeasible, so we adopted a grid-based approach.

For the synthesis of the LPV controller, the grid of scheduling variables  $p_1, \dots, p_n \in \mathcal{P}_{\text{grid}} \subset \mathcal{Z}_{\text{koop}}$ , and corresponding Koopman-linear system matrices  $A, B(z_k), C, D$  are used. However, as the mapping between the space of the velocity dynamics of the physical system  $X_{\text{vel}}$  and the Koopman-invariant subspace of the space  $X_{\text{koop}}$  is a bijection, the physical states can be used as scheduling variables. This approach greatly reduces the dimension of the scheduling space.

However, with a high-dimensional scheduling variable, such as in our case, the formation of a structured grid is problematic. Even with a low-resolution grid, discretizing each of the 9 dimensions into 3 points results in  $3^9 = 19,683$  gridpoints, and the optimization procedure becomes intractable. Additionally, the precision of the trained neural network is highly dependent on the location of the data points. Due to its weak extrapolation and interpolation capabilities, in the state space regions that are scarcely populated by datapoints, its performance is insufficient.

Consequently, we decided to use an unstructured grid, with the gridpoints formed by a selection of the training data points. Given that the neural network is trained on 140,160 data points, a subset was needed to maintain computational tractability for synthesis. We first reduced the dataset by discarding 80% of the points and further refined it by selecting uniformly distributed points. This was achieved using Algorithm 1 to identify and retain

the points with the greatest distance from each other, ensuring better performance for the chosen interpolation method.

Algorithm 1 works as follows: Starting from an initial point, it iteratively selects points that are farthest from the already chosen subset. For each new selection, the algorithm calculates the minimum distance between each remaining point and the points already in the subset. The point with the maximum minimum distance is then added to the subset. This approach maximizes the spacing between selected points, resulting in a more uniform distribution across the data space.

---

**Algorithm 1:** Distance based data set reduction

---

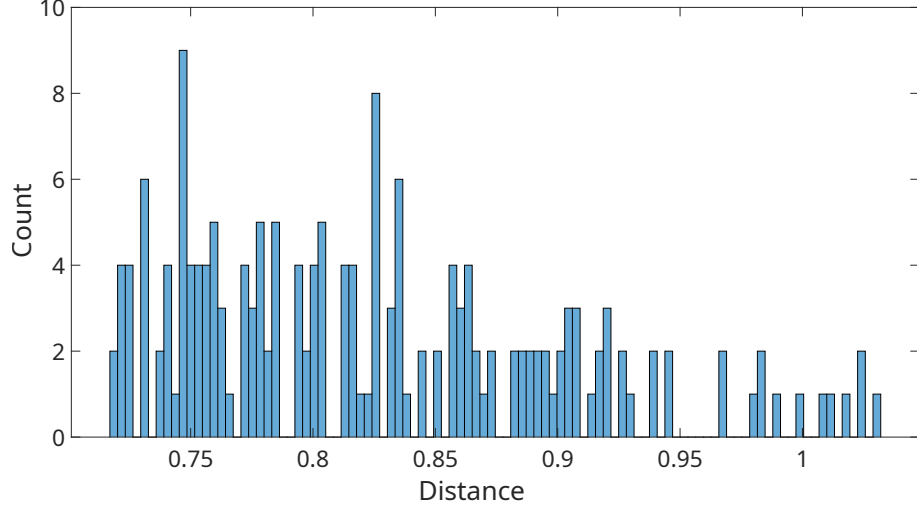
**Input:**  $p, n_{des}, n$   
**Output:**  $\bar{p}$   
 $idxs_1 \leftarrow 1$  ;  
 $\bar{p}_1 \leftarrow p_1$  ;  
**for**  $i \leftarrow 2$  **to**  $n_{des}$  **do**  
     $d_{max} \leftarrow 0$  ;  
     $idx_{max} \leftarrow 0$  ;  
    **for**  $j \leftarrow 1$  **to**  $n$  **do**  
         $d_{min} \leftarrow \infty$  ;  
        **for**  $k \leftarrow 1$  **to**  $i - 1$  **do**  
             $d \leftarrow \|p_j, p_{idxs_k}\|_2$  ;  
            **if**  $d < d_{min}$  **then**  
                 $d_{min} \leftarrow d$  ;  
            **end**  
        **end**  
        **if**  $d_{min} > d_{max}$  **then**  
             $d_{max} \leftarrow d_{min}$  ;  
             $idx_{max} \leftarrow j$  ;  
        **end**  
    **end**  
     $idxs_i \leftarrow idx_{max}$  ;  
     $\bar{p}_i \leftarrow p_{idxs_i}$  ;  
**end**

---

Using this algorithm we selected 200 datapoints, after which 10% of the points with the largest distances were discarded to remove outliers.

#### 4.1.4 Synthesis of the LPV controller

The controller synthesis is based on *linear matrix inequalities* (LMIs), and the problem formulation is in the class of *semidefinite programming* (SDP), which is efficiently solved by solvers such as YALMIP or CVX. Stability of the closed loop is ensured by a quadratic storage function that qualifies as a Lyapunov function, which is synthesized during optimization if the problem is feasible. Our objective of minimization of the  $\mathcal{L}_2$  signal norm between the generalized disturbance and the control output vectors is assured by minimization of the variable  $\gamma$  in the problem formulation. The resulting  $\gamma$  of the optimization process shows how low the worst case signal norm could be kept. For detailed derivation of the synthesis method see [6]. A value  $\gamma \leq 1$  shows, that the closed loop behavior satisfies the specifications given by the filters. If this value is very low, the performance of the



**Figure 4.3:** Histogram of distances of the data points from each other after the distance based data set reduction.

controller can be increased by tightening the filters, hence requiring higher tracking error and noise suppression, or less aggressive control inputs.  $\gamma$  could be pushed lower by relaxing the filters.

$$\min \gamma, \quad (4.3)$$

$$\text{s.t.} \quad \begin{pmatrix} X & A_{cl}(p_i) & B_{cl}(p_i) & 0 \\ A_{cl}(p_i)^\top & G + G^\top - X & 0 & C_{cl}^\top \\ B_{cl}(p_i)^\top & 0 & \gamma I & D_{cl}^\top \\ 0 & C_{cl} & D_{cl} & \gamma I \end{pmatrix} \prec 0, \quad (4.4)$$

for all  $p_i \in \mathcal{P}_{\text{grid}}$ , where

$$\begin{aligned} A_{cl}(p_i) &= AG - B_u(p_i)F_i \\ B_{cl}(p_i) &= B_w(p_i) \\ C_{cl} &= C_z G - D_{zw} F_i \\ D_{cl} &= D_{zw} \end{aligned}$$

Here, the matrices  $X \in \mathbb{R}^{n_z \times n_z}$ ,  $G \in \mathbb{R}^{n_z \times n_z}$  and  $F_i \in \mathbb{R}^{n_u \times n_z}$  for  $i = 1, 2, \dots, n$  are decision variables, with  $X$  being a symmetric matrix.

The controller  $K$  can be obtained by:

$$K_i = F_i^* G^{*-1} \quad (4.5)$$

For the controller synthesis we used the LPVcore MATLAB toolbox, which uses YALMIP as the solver for the constructed optimization problems.

#### 4.1.5 Scattered interpolation between feedback gains

Our grid-based LPV controller synthesis procedure produces feedback gains for the corresponding grid points in the scheduling space. In order to use the controller at points  $p \notin \mathbb{P}_{\text{grid}}$ , an interpolation method must be used.

Due to the high dimensionality of the interpolation space, our decision fell on a radial basis function-based algorithm, implemented in the LPVcore toolbox [6]. This algorithm also has the additional advantage of being  $C^\infty$  continuous. This removes transients often seen in gain scheduled controllers, during switching between feedback gains.

**Data:** After controller synthesis, we obtained  $N$  feedback matrices corresponding to scheduling data points:

$$P = \{p_1, p_2, \dots, p_N\},$$

$$K = \{K_1, K_2, \dots, K_N\}.$$

**Interpolation Weight Computation:** To perform the scattered interpolation, we base our method on the Radial Basis Functions (RBF) method described in [1, Section 4]. First, we compute a "distance" matrix, containing the distances (defined by RBFs), given by:

$$M_{i,j} = d(p_i, p_j),$$

for  $i, j = 1, \dots, N$  where

$$d(x, y) = \sqrt{\|x - y\|_2^2 + c^2},$$

i.e., the quadric RBF, where  $c \in \mathbb{R}$  is a tuning parameter. Note that the matrix  $M$  is symmetric. Next, we construct the matrix  $\bar{K}_v \in \mathbb{R}^{(n_u \cdot n_x) \times N}$ , given by:

$$\bar{K}_v = \begin{bmatrix} \text{vec}(K_1) & \text{vec}(K_2) & \dots & \text{vec}(K_N) \end{bmatrix},$$

where  $\text{vec}(\cdot)$  denotes the (column-wise) vectorization of a matrix. Denote by  $\text{mat}(\cdot)$  the inverse operation of  $\text{vec}$ , such that  $\text{mat}(\text{vec}(K_i)) = K_i$  for  $i = 1, \dots, N$ . Then, we want to construct a weight matrix  $W \in \mathbb{R}^{N \times (n_u \cdot n_x)}$  such that

$$\bar{K}_v^\top = MW.$$

Therefore, we can compute  $W$  as follows:

$$W = M^{-1} \bar{K}_v^\top.$$

Moreover, let us denote:

$$W = \begin{bmatrix} W_1 \\ \vdots \\ W_N \end{bmatrix},$$

where  $W_i \in \mathbb{R}^{1 \times (n_u \cdot n_x)}$  for  $i = 1, \dots, N$ .

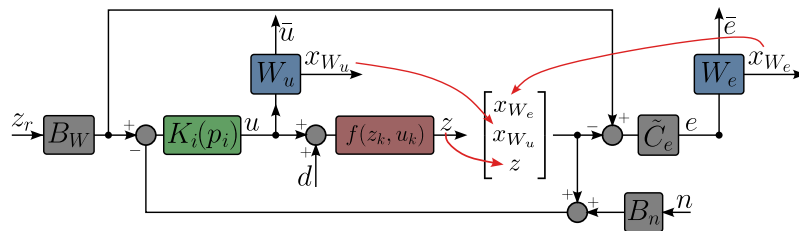
The condition number of the distance matrix  $D$  can be high, i.e. it can be close to singular if the scheduling points are too close to each other. In our experience, using the distance based data reduction procedure described in Section 4.1.3, ensures that  $D$  can be inverted.

**Interpolation:** Based on our calculated weight matrix  $W$ , we have that:

$$\text{vec}(K_i)^\top = \sum_{j=1}^N d(p_i, p_j) \cdot W_j,$$

for  $i = 1, \dots, N$ . Let us define the matrix function  $K$  of the interpolated controller by  $\hat{K}$ . Therefore, at a scheduling point  $p^* \in \mathbb{R}^{n_p}$  that is not part of our data set, the interpolated values of  $K$  can be computed as:

$$\text{vec}(\hat{K}(p^*))^\top = \sum_{j=1}^N d(p^*, p[j]) \cdot W_j.$$



**Figure 4.4:** Controller design.

The structure of the designed controller is shown in Figure 4.4. Weighting filters are included within the closed loop, as in the LPV controller synthesis method (and its LTI equivalent in  $H_\infty$  synthesis), these filters expand the state space of the system. We denote the expanded state vector as  $\tilde{z} \in \tilde{X}_{\text{koop}}$ . During controller operation, the filter states are updated based on their state-space representation and the controlled output signal.

The noise and reference signals belong to the Koopman state-space:  $n, z_r \in X_{koop}$ . The matrices  $B_W, B_n \in \mathbb{R}^{n_z \times n_z}$  project these signals into the expanded state space  $\tilde{X}_{koop}$ .

## 4.2 PI for position tracking

As only the velocity dynamics of the system is described by the trained Koopman-SUBNET, the LPV controller is designed to follow a velocity reference. The position states cannot be included in the Koopman-linear representation, as the mapping from the physical state-space is only through the nonlinear observable function  $\Phi$ .

In this work position tracking is solved by designing a PI controller for the outer loop:

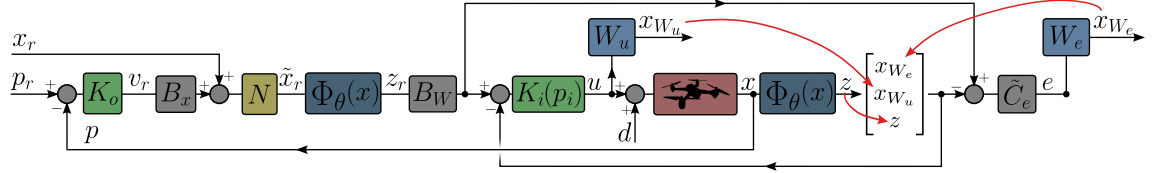
$$\dot{p} = v, \quad (4.6)$$

where  $p$  is the position of the quadrotor, and  $v$ , its velocity is the control input. The controller was manually tuned through trial and error to optimize reference tracking for position channels.

The full controller structure is shown in Figure 4.5, where the inner- and outer-loop controllers are labeled  $K_i$  and  $K_o$ , respectively. The outer-loop PI controller receives the difference between position of the quadcopter and the position reference. Its output, serving as the velocity reference for the inner loop, is then projected onto the velocity state space of the physical system,  $X_{\text{full}}$ , using the projection matrix  $B_x$ . This vector is aug-

mented by adding a feedforward signal, the velocity state reference which corresponds to the reference trajectory. The resulting vector is then transformed into the space  $\tilde{X}_{\text{koop}}$ .

Since the neural network was trained on a normalized dataset, the first step is to normalize the vector using an affine transformation  $N$ . The state vector is then encoded through the observable function  $\Phi_\theta$  and finally the Koopman-state vector is projected onto the space  $\tilde{X}_{\text{koop}}$  through the matrix  $B_W$ .



**Figure 4.5:** Cascade controller setup for position reference tracking.

## Chapter 5

# Koopman control of the quadrotor

Synthesis of the controller was done in MATLAB using the numerical optimization toolbox YALMIP. Selecting the proper hyperparameters for the synthesis method, and running the optimization algorithm was a time consuming process due to the high dimension.

The important hyperparameters to be chosen were the number of grid points to be used, and the characteristics of the performance shaping filters. As performance and stability is only guaranteed at the gridpoints with a fixed scheduling variable, the more densely populated the scheduling variable space, and the tighter the filter specifications are, the better the interpolated feedback gain performs. However, this often resulted in a high  $\gamma$ , indicating that the performance requirements could not be met. The value of  $\gamma$  can be reduced by relaxing the filters, but the controller performance was often found to be insufficient, with poor reference following capabilities when the filters were relaxed too much. The number of gridpoints also had a significant influence on the optimization time and even the convergence of the optimization process, as choosing it too high resulted in the infeasibility of the optimization problem.

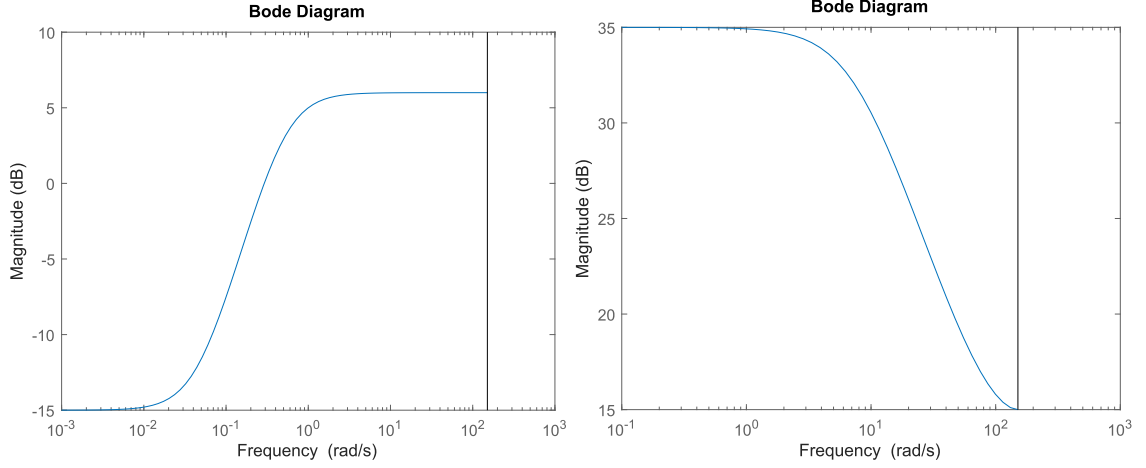
For the controller design 200 datapoints were selected, and the filters used can be seen in Figure 5.1. A controller with these design decisions required approximately 40 minutes for the optimization algorithm to complete with  $\gamma = 1.49$ . This violates the required  $\gamma \leq 1$  value, but only slightly, and analyzing the time-domain behavior of the closed-loop system showed that the performance was best at this value.

To test the controller structure, the Test path described in Section 3.2.1 was used as the nominal path. As a reference, the PI controller was provided with the position states, which generate translational velocities for the inner loop of the controller to follow. The target velocity and orientation states were added to the output of the PI controller as feedforward signals.

The closed loop simulation setup can be seen in Figure 4.5.

The Koopman SUBNET model based controller is compared to a nominal model based LPV controller with the same filter specifications. This controller was created using the same grid based synthesis method. The grid points were obtained by linearizing the NL quadcopter model (3.1) around a set of 3-dimensional regular grid points which was constructed using the following discrete values:  $\phi \in \{-45^\circ, 0^\circ, 45^\circ\}$ ,  $\theta \in \{-45^\circ, 0^\circ, 45^\circ\}$ ,  $\psi \in \{-25^\circ, 0^\circ, 25^\circ\}$ .

Figures 5.2, 5.3 show the performance of the inner-loop LPV-controller, while Figures 5.4 and 5.5 depict the performance of the whole cascade controller structure. As expected, the nominal model-based controller performs slightly better, as the Koopman controller relies



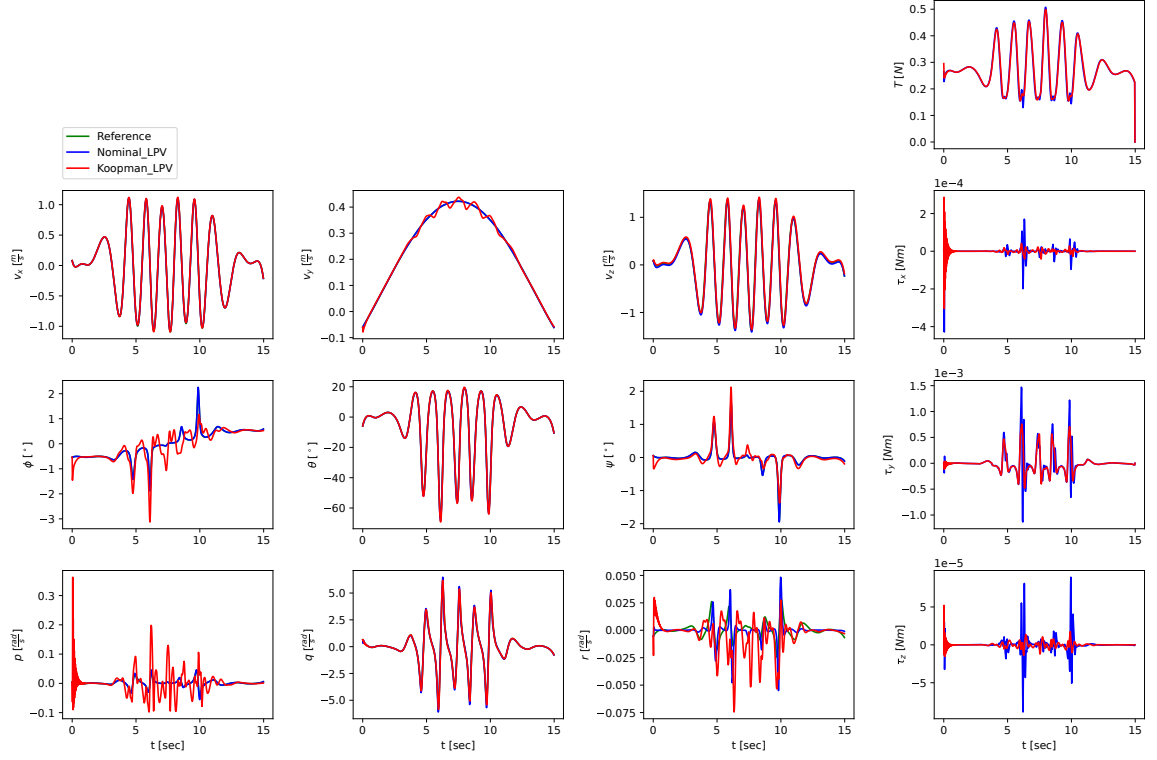
(a) Inverse of the chosen filters for  $W_e$ . (b) Inverse of the chosen filters for  $W_u$ .

**Figure 5.1:** The used filter for the performance channels.

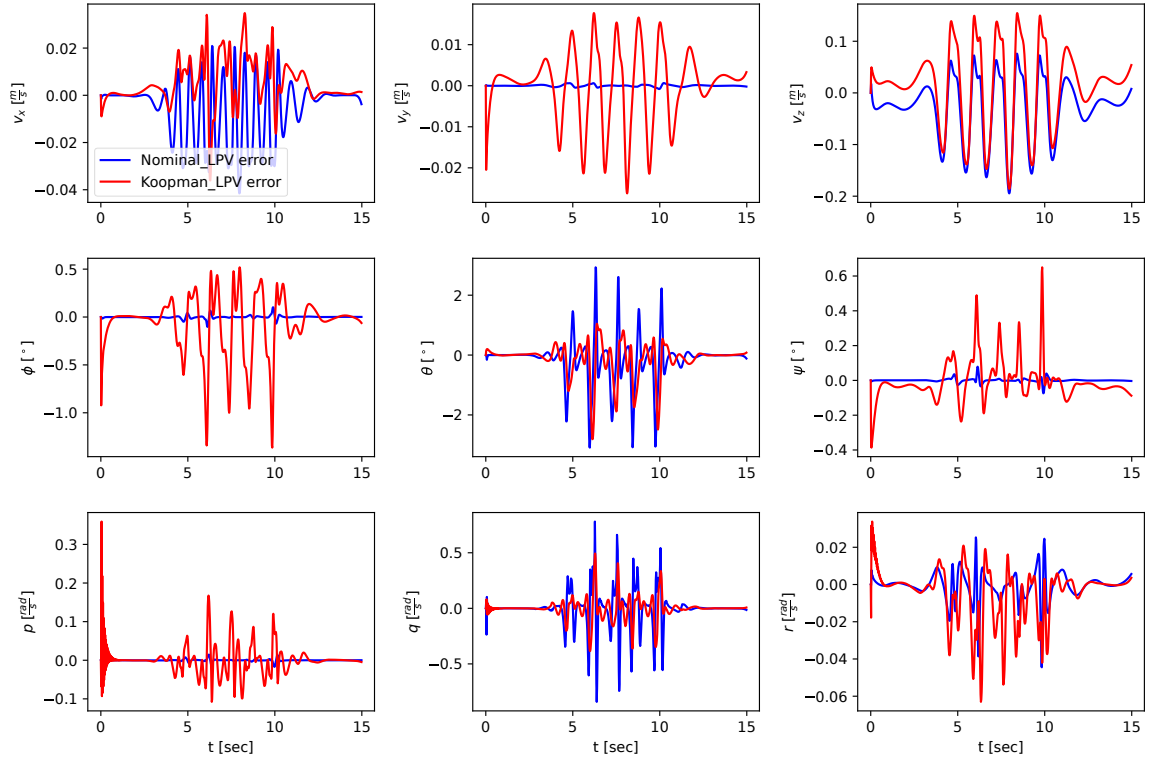
on a model identified from a dataset, which introduces estimation error from the neural network optimization and losses due to truncation of the infinite-dimensional Koopman state space. Nonetheless, the Koopman-based controller performed well, with an NRMS reference tracking error of 3%, while the nominal LPV controller outperformed it only slightly with a 2% error.

Examining Figure 5.2 it is conspicuous, that the velocity reference  $v_y$  is followed with a much higher error by the Koopman-controller, and oscillations occurs around the nominal velocity values. However, inspecting the error plot in Figure 5.3, it can be seen that while the Koopman model-based LPV controller has a larger error in this channel, it outperforms the nominal model-based controller in  $v_x$ , where faster changes in the velocities occur.

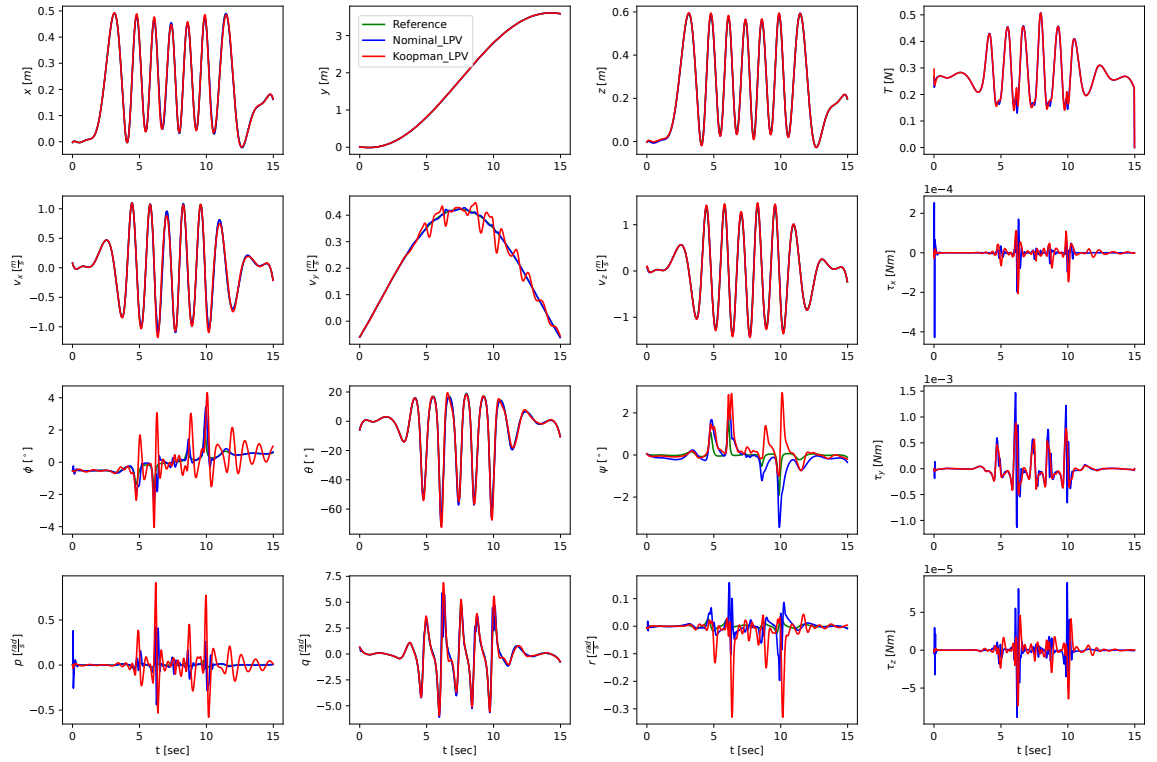
In the comparison of the full controller structure with the nominal model-based LPV controller, the performance decrease of the Koopman model-based controller becomes more apparent. However, it still performs well, with deviations of no more than 2 cm from the nominal fast-paced trajectory in the position channels.



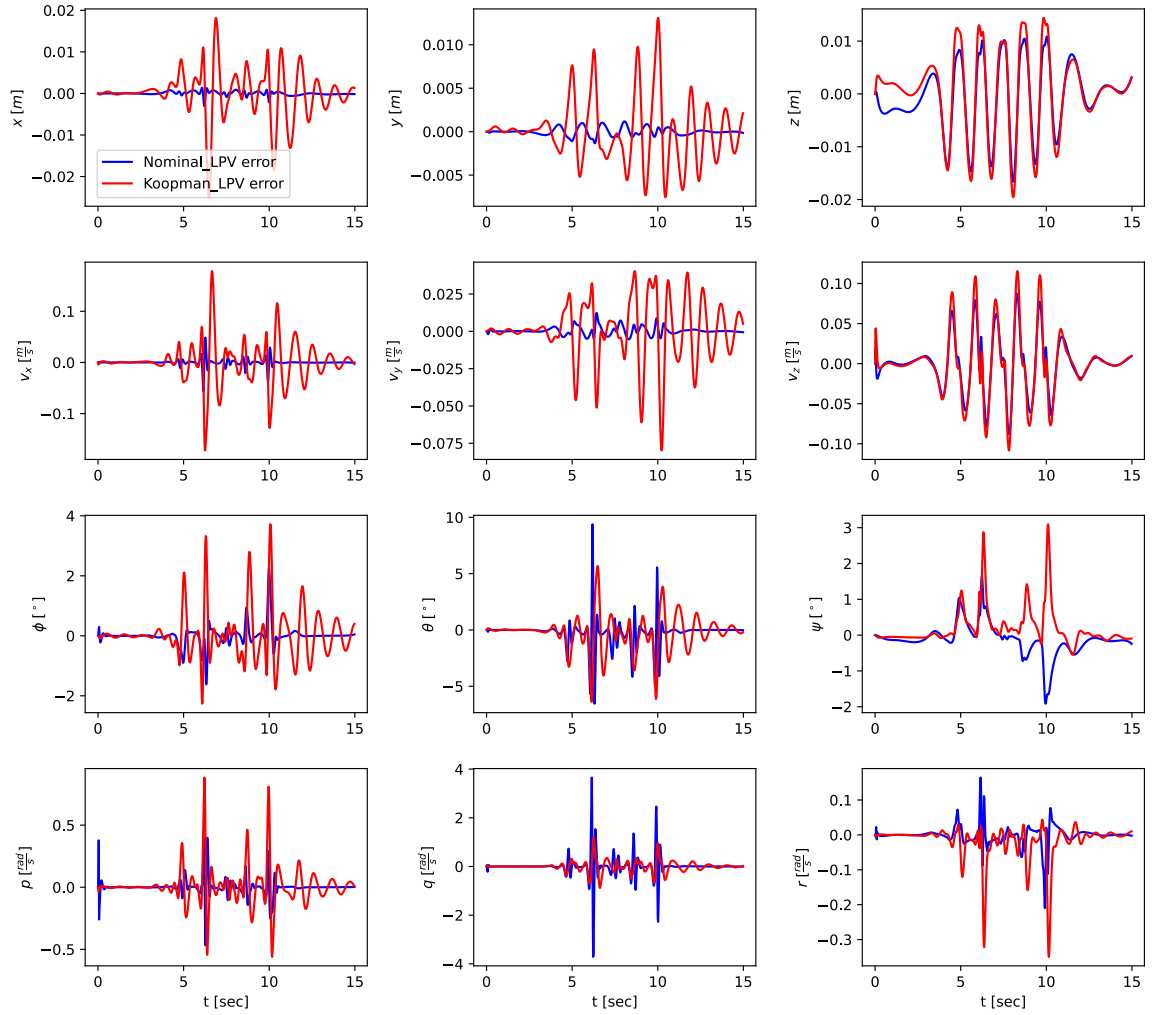
**Figure 5.2:** Comparison between the nominal model based, and Koopman SUB-NET based LPV controllers in the inner loop. The plots show the states and inputs during test flight.



**Figure 5.3:** Comparison between the nominal model based, and Koopman SUB-NET based LPV controllers in the inner loop. The plots show the state errors during test flight.



**Figure 5.4:** Comparison between the nominal model based, and Koopman SUB-NET based full controllers. The plots show the states and inputs during test flight.



**Figure 5.5:** Comparison between the nominal model based, and Koopman SUB-NET based full controllers. The plots show the state errors during test flight.

## Chapter 6

# Conclusions

In this work, we successfully identified the dynamics of a drone using the Koopman operator framework, enabling us to represent the nonlinear system in a linear form. This transformation enabled the design of a reference tracking controller. The developed controller demonstrated good performance in following reference trajectories, confirming the effectiveness of the Koopman-based approach in modeling complex systems.

## Possible Improvements and Future Work

While the results are promising, there are several areas where improvements can be made:

- **Affine dependence of the dynamics on scheduling variable:** By ensuring an affine dependence of the control input matrix  $B$  on the scheduling variable  $p$ , global LPV controllers can be synthesized, and the need for interpolation between feedback gains can be eliminated. This approach would guarantee performance and stability across the entire operating range.
- **Reduction of scheduling variables and state space:** Simplifying the model by reducing the number of scheduling variables and decreasing the state space dimension can simplify the controller design process. This can be done with the help of the [6] toolbox, which has various built-in tools to achieve dimensionality reduction.
- **Implementation on real systems:** Transitioning the controller design procedure from simulations to a real quadrotor is an important next step. Real-world implementation will allow us to assess the performance of the controller under practical conditions.
- **More efficient optimization algorithms for neural network optimization:** The gradient descent-based Adam optimizer can be slow when optimizing neural networks that represent high-dimensional systems. Improvements can be made by using more efficient optimization methods, such as quasi-Newton methods like variants of the BFGS algorithm [18], such as [4], that is particularly suitable for ANN-SS models.

## Important Considerations

Some important findings from this project are worth noting:

- **Trade-off between analytical modeling and data-driven approaches:** While the traditional method of describing dynamics using first principles is time-consuming, the data-driven approach introduces its own time-consuming challenges. These include the task of data collection, optimal hyperparameter selection for neural networks, and long optimization of the network parameters.
- **Performance limitations in high-dimensional systems:** In systems with high-dimensional state spaces, achieving optimal performance may be confined to regions that are densely populated by collected datapoints. This limitation emphasizes the need for comprehensive data gathering across all relevant operating conditions to ensure model accuracy and effectiveness; however, collecting high-quality data that covers the entire state space may be impractical or even impossible. A promising approach to address this issue involves taking advantage of already developed first principle models, and enhancing their precision by augmenting them with neural networks.
- **Suitability for systems with repetitive references:** The used black-box identification approach is particularly well-suited for systems where the reference signals are repetitive, such as for robots working in production lines. In these scenarios, the operating region is narrow, and straightforward to explore during the data collection phase.

# Acknowledgements

I would like to thank Roland Tóth and Tamás Péni at SZTAKI for their help and support throughout this work.

The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems. (RRF-2.3.1-21-2022-00002)

# Bibliography

- [1] Isaac Amidror. Scattered data interpolation methods for electronic imaging systems: A survey. *Journal of Electronic Imaging*, 11, 04 2002. DOI: 10.1117/1.1455013.
- [2] Gerben Beintema, Maarten Schoukens, and Roland Tóth. Deep subspace encoders for nonlinear system identification. *Automatica*, 10 2022. DOI: 10.48550/arXiv.2210.14816.
- [3] Gerben I. Beintema, Maarten Schoukens, and Roland Tóth. Deep subspace encoders for nonlinear system identification. *Automatica*, 156:111210, 10 2023.
- [4] Alberto Bemporad. An l-bfgs-b approach for linear and nonlinear system identification under  $\ell_1$ - and group-lasso regularization, 2024. URL <https://arxiv.org/abs/2403.03827>.
- [5] Petar Bevanda, Bas Driessen, Lucian Cristian Iacob, Roland Toth, Stefan Sosnowski, and Sandra Hirche. Nonparametric control-koopman operator learning: Flexible and scalable models for prediction and control, 2024. URL <https://arxiv.org/abs/2405.07312>.
- [6] Pascal den Boef, Pepijn B. Cox, and Roland Tóth. Lpvc core: Matlab toolbox for lpv modelling, identification and control. *IFAC-PapersOnLine*, 54(7):385–390, 2021. ISSN 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2021.08.390>. URL <https://www.sciencedirect.com/science/article/pii/S2405896321011642>. 19th IFAC Symposium on System Identification SYSID 2021.
- [7] Marco Forgone and Dario Piga. Continuous-time system identification with neural networks: Model structures and fitting criteria. *Eu. Jour. of Cont.*, 59:69–81, 2021.
- [8] Debdipta Goswami and Derek A. Paley. Global bilinearization and controllability of control-affine nonlinear systems: A koopman spectral approach. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 6107–6112, 2017. DOI: 10.1109/CDC.2017.8264582.
- [9] Bowen Huang, Xu Ma, and Umesh Vaidya. Feedback stabilization using koopman operator. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6434–6439, 2018. DOI: 10.1109/CDC.2018.8619727.
- [10] Lucian Iacob, Roland Tóth, and Maarten Schoukens. Koopman form of nonlinear systems with inputs, 07 2022.
- [11] Lucian Cristian Iacob, Gerben Izaak Beintema, Maarten Schoukens, and Roland Tóth. Deep identification of nonlinear systems in koopman form. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2288–2293, 2021. DOI: 10.1109/CDC45484.2021.9682946.

- [12] Lucian Cristian Iacob, Roland Tóth, and Maarten Schoukens. Koopman Form of Nonlinear Systems with Inputs, December 2023. URL <http://arxiv.org/abs/2207.12132>. arXiv:2207.12132 [cs, eess].
- [13] Lucian Cristian Iacob, Roland Tóth, and Maarten Schoukens. Koopman form of nonlinear systems with inputs. *Automatica*, 162:111525, 2024. ISSN 0005-1098. DOI: <https://doi.org/10.1016/j.automatica.2024.111525>. URL <https://www.sciencedirect.com/science/article/pii/S0005109824000177>.
- [14] Alberto Isidori, M. Thoma, E. D. Sontag, B. W. Dickinson, A. Fettweis, J. L. Massey, and J. W. Modestino. *Nonlinear Control Systems*. Springer-Verlag, Berlin, Heidelberg, 3rd edition, 1995. ISBN 3540199160.
- [15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.
- [16] P. J. W. Koelewijn. *Analysis and Control of Nonlinear Systems with Stability and Performance Guarantees - A Linear Parameter-Varying Approach*. PhD thesis, Eindhoven University of Technology, 2023.
- [17] B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931. DOI: 10.1073/pnas.17.5.315. URL <https://www.pnas.org/doi/abs/10.1073/pnas.17.5.315>.
- [18] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989. URL <https://api.semanticscholar.org/CorpusID:5681609>.
- [19] Daniele Masti and Alberto Bemporad. Learning nonlinear state–space models using autoencoders. *Automatica*, 129:109666, 2021.
- [20] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proc. of the IEEE Int. Conf. on Rob. and Aut.*, pages 2520–2525, 2011.
- [21] J. Scherer Mohammadpour and C.W. *Control of Linear Parameter Varying Systems with Applications*. Springer-Verlag New York, 2012.
- [22] Javad Mohammadpour Velni and Carsten Scherer. *Control of Linear Parameter Varying Systems with Applications*. Springer, 01 2012. ISBN 9781461418337. DOI: 10.1007/978-1-4614-1833-7.
- [23] C.W. Scherer. Lpv control and full block multipliers. *Automatica*, 37(3):361–375, 2001. ISSN 0005-1098. DOI: [https://doi.org/10.1016/S0005-1098\(00\)00176-X](https://doi.org/10.1016/S0005-1098(00)00176-X). URL <https://www.sciencedirect.com/science/article/pii/S000510980000176X>.
- [24] Peter Schmid and Jörn Sesterhenn. Dynamic Mode Decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656, November 2008. DOI: 10.1017/S0022112010001217.
- [25] Roland Tóth. *Modeling and Identification of Linear Parameter-Varying Systems*, volume 403, chapter 3. Springer Germany, 01 2010. DOI: 10.1007/978-3-642-13812-6.

- [26] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, June 2015. ISSN 1432-1467. DOI: 10.1007/s00332-015-9258-5. URL <http://dx.doi.org/10.1007/s00332-015-9258-5>.