NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE
DEPARTMENT OF MATHEMATICS

PIOTR MARCIN ŁASZKIEWICZ
BEng in Biomedical Engineering

DATA-DRIVEN SYSTEM IDENTIFICATION OF RECONFIGURABLE PHYSICAL SYSTEMS

MASTER IN ANALYSIS AND ENGINEERING OF BIG DATA

NOVA University Lisbon
September, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE
DEPARTMENT OF MATHEMATICS

DATA-DRIVEN SYSTEM IDENTIFICATION OF RECONFIGURABLE PHYSICAL SYSTEMS

PIOTR MARCIN ŁASZKIEWICZ

BEng in Biomedical Engineering

Adviser: Cláudia Soares
Assistant Professor, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Co-adviser: Pedro Lourenço
Head of Guidance and Control Section, GMV Portugal

Examination Committee

Chair: João Carlos Gomes Moura Pires
Associate Professor, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Adviser: Cláudia Soares
Assistant Professor, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Member: Zachary Manchester
Assistant Professor, Robotics Institute Carnegie Mellon University

MASTER IN ANALYSIS AND ENGINEERING OF BIG DATA

NOVA University Lisbon
September, 2024

Data-driven system identification of reconfigurable physical systems

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.18) [1].

Acknowledgements

I am extremely grateful to prof. Cláudia Soares, an Assistant Professor at the NOVA University of Lisbon, and dr Pedro Lourenço, a Head of Guidance and Control Section at the GMV Portugal, without whom it would not be possible to develop this thesis. Their patience, willingness to help, and high expertise were not only a crucial factor to complete this work, but also a sufficient one to keep me highly motived and passionate about the research that we conducted.

I am also thankful to the staff of the Nova School of Science and Technology, who either directly or indirectly contributed towards the excellent educational process that I experienced during my studies in Portugal. I am greatly impressed by the passion that the Professors of this School carry towards science and sharing it with the students. I should also remark their high level of hospitality, which made me feel home during my time abroad.

Special thanks should go to GMV Portugal and GMV Poland, who helped me develop this thesis in collaboration with the space industry. This exceptional opportunity enabled me to better understand the importance of the intersection of scientific research with more business-oriented units. Moreover, they enabled me to gain professional experience, as well as focus on the thesis development without worrying about the financial aspects of life.

I would also like to acknowledge my family and friends for their invaluable support. They made a major contribution of not letting me go too high or low during the everyday ups and downs, thus keeping me on the right track. Their good advice has always had a positive impact on my life, and I hope they realize that my achievements, are ours in the end.

Lastly, I would like to thank my dear friend Bartłomiej, without whom this Portuguese journey might have not happened at all.

”

“Each generation stands on the shoulders of those who have gone before them.”

— Stephen Hawking, regarding Open Access to his PhD thesis
(Physicist, cosmologist)

Abstract

This thesis presents a benchmark of Linear Time Varying (LTV) system identification methods with focus on reconfigurable spacecraft. For this purpose we developed simulations of a spring mass damper system in five challenging scenarios including nonlinear damping, strong perturbations, and reconfiguration. To tackle the problem of system identification, we propose two approaches: 1. An integration of nonlinear autoregressive model with exogenous input (NLARX) with different deep learning architectures including multilayer perceptron (MLP), residual network (ResNet), recurrent neural network (RNN), long-short term memory network (LSTM), and regression algorithms such as XGBoost or regression forest. 2. An application of Physics-informed Neural Networks (PINNs) to model the state-space representation of the system. In the benchmark, we enhance this class of machine-learning-based algorithms with fast closed-form identification from large-scale data for LTV systems (COSMIC), and compare them with methods originating from systems theory field, i.e., time-varying eigensystem realization algorithm (TVERA), time-varying observer/Kalman filter identification (TVOKID), and identification of LTV dynamical models with smooth or discontinuous evolution by means of convex optimization (LTVModels). We have found that the machine-learning-based methods dominate the benchmark in terms of state propagation accuracy, however it is not possible to select one best approach for the discussed problem.

Keywords: System identification, benchmark, NLARX, Physics-informed Neural Networks, COSMIC, TVERA, TVOKID, LTVModels, Reconfigurable spacecraft

Resumo

Esta tese apresenta um benchmark de métodos de identificação de sistemas lineares variáveis no tempo com enfoque em veículos espaciais reconfiguráveis. Para este efeito, desenvolvemos simulações de um sistema amortecedor massa mola em cinco cenários desafiantes, incluindo amortecimento não linear, fortes perturbações e reconfiguração. Para resolver o problema da identificação do sistema, propomos duas abordagens: 1. Uma integração de um modelo autoregressivo não linear com entrada exógena com diferentes arquitecturas de aprendizagem profunda, incluindo MLP, ResNet, RNN, LSTM, e algoritmos de regressão como o XGBoost ou regression forest. 2. Uma aplicação de redes neuronais informadas sobre a física para modelar a representação do espaço de estados do sistema. No parâmetro de referência, melhoramos esta classe de algoritmos baseados em aprendizagem automática com COSMIC e comparamo-los com métodos provenientes do domínio da teoria dos sistemas, ou seja, TVERA, TVOKID e LTVModels. Verificámos que os métodos baseados na aprendizagem automática dominam o parâmetro de referência em termos de precisão da propagação de estados. No entanto não é possível selecionar a melhor abordagem para o problema em questão.

Palavras-chave: Identificação de sistemas, referência, NLARX, Redes Neurais Informadas com Física, COSMIC, TVERA, TVOKID, LTVModels, Naves espaciais reconfiguráveis

Contents

List of Figures

List of Tables

Acronyms

ARMAX autoregressive moving average model with exogenous input *(pp. 5, 7)*
ARX autoregressive model with exogenous input *(pp. 5, 7, 8)*

COSMIC fast closed-form identification from large-scale data for LTV systems *(pp. iv, v, 5, 9, 12, 13, 15, 25, 26, 29, 37, 46)*

DMDc Dynamic Mode Decomposition with Control *(pp. 5, 9, 12)*

ECDF empirical cumulative distribution function *(pp. 34, 35, 39, 45, 46)*
ERA eigensystem realization algorithm *(pp. 5, 9, 10)*

FIR Finite Impulse Response *(pp. 3, 5, 8)*

LQR Linear Quadratic Regulator *(p. 53)*
LS Least Squares *(pp. 7, 8, 16)*
LSTM long-short term memory network *(pp. iv, v, 30, 38, 52)*
LTI Linear Time Invariant *(pp. 2, 3, 5–7, 9, 10, 12, 19)*
LTV Linear Time Varying *(pp. iv, 2–4, 6, 10–12, 19, 20, 23, 33, 37, 45, 46, 52)*
LTVModels identification of LTV dynamical models with smooth or discontinuous evolution by means of convex optimization *(pp. iv, v, 5, 9, 11, 26, 29)*

MLE Maximum Likelihood Estimation *(p. 16)*
MLP multi-layer perceptron *(pp. iv, v, 30, 38, 52, 53)*
MPC Model Predictive Control *(p. 53)*
MSE Mean Squared Error *(pp. 16, 30)*

NL nonlinear *(pp. 19, 21, 33, 37, 45)*
NLARX nonlinear autoregressive model with exogenous input *(pp. iv, 5, 8, 29, 30, 36, 38, 39, 41, 45, 52)*

NLD nonlinear perturbed *(pp. 19, 21, 33, 37, 45)*

ODE ordinary differential equation *(pp. 20–22, 53)*
OKID observer/Kalman filter identification *(pp. 5, 9, 10)*

PINNs Physics-informed Neural Networks *(pp. iv, 17, 29, 52)*

ResNet residual network *(pp. iv, v, 30, 38, 52)*
RMSE root mean squared error *(pp. 34, 38, 45)*
RNN recurrent neural network *(pp. iv, v, 30, 38, 52)*

SINDYc Sparse Identification of Nonlinear Dynamics with Control *(pp. 5, 9, 14, 15)*
SISO Single Input Single Output *(pp. 2, 6)*
SVD singular value decomposition *(p. 10)*

TVERA time-varying eigensystem realization algorithm *(pp. iv, v, 5, 9–11, 26, 29)*
TVOKID time-varying observer/Kalman filter identification *(pp. iv, v, 5, 9–11, 26, 29, 36)*

1

Motivation and Problem Statement

1.1 Motivation

One of the main limitations of carrying spacecraft to orbit is their size and weight. Therefore, some extremely large and heavy objects, such as e.g., the International Space Station (ISS) are transported to orbit piece-by-piece, and the assembly of these pieces is carried out in orbit [2]. The construction of large structures represents a major trend in future space exploration, and there is a growing interest in performing these assemblies using robotic manipulators [3].

The construction of spacecraft in orbit makes the control of its attitude a difficult task, as the assembly modifies the dynamics of the system. These changes can be both abrupt and continuous, and they are observed mostly because the inertia and vibrational dynamics of the structure are affected by the assembly process.

Typically, the problem of modelling a system is addressed by applying first-principles of physics. This approach requires measuring all the parameters necessary to completely describe the behavior of a system. For more intricate systems, such as spacecraft during in-orbit assembly, this can fail, as it might be impossible to measure all the required parameters, or because the system can be too complex to be modeled by simple equations. A possible way to overcome these obstacles is to apply a data-driven approach for system identification [4].

Utilizing data driven approaches for system identification can play a pivotal role in substantially decreasing project expenses. It is projected that system modelling accounts for as much as 50% of the total costs of complex engineering projects [5]. A reduction of costs in system modelling is particularly relevant in space missions, which typically entail substantial investments from agencies and national governments. By adopting data driven strategies, these projects can achieve a more efficient resource allocation enhancing the feasibility and sustainability of space exploration.

1.2 Problem statement

System identification is a problem of obtaining a mathematical description of a dynamical system given its behavior as a set of observations [6]. Such mathematical model can then be applied to various tasks like control, analyzing system properties, performing simulation, prediction, filtering, fault diagnosis, and others [7]. System identification is, therefore, an important problem encountered in various areas such as biology, medicine, chemical processes, economics, geology, materials, engineering, flight vehicles, and so on [6].

1.2.1 System representation

To understand system identification, it is crucial to comprehend the basics of systems theory. Therefore, this section is devoted to a brief review of some basic concepts established by this field.

Linear state-space model

A common way to describe a linear system is the linear state-space model defined by

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t),$$

where $t \in \mathbb{R}$ is time, $\mathbf{x}(t) \in \mathbb{R}^n$ is the state of the system, $\mathbf{u}(t) \in \mathbb{R}^k$ is the input to the system, $\mathbf{A}(t) \in \mathbb{R}^{n \times n}$, and $\mathbf{B}(t) \in \mathbb{R}^{n \times k}$. The discrete-time version of this equation is

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k),$$

where $k \in \mathbb{N}$ is the discrete index. In the case where $\mathbf{A}(t) = const.$ and $\mathbf{B}(t) = const.$ or $\mathbf{A}(k) = const.$ and $\mathbf{B}(k) = const.$, we say that the system is Linear Time Invariant (LTI), whereas in the other case, the system is called LTV [8]. In both cases, the matrices $\mathbf{A}$ and $\mathbf{B}$ describe the dynamical system and can be considered its parameters.

Finite impulse response model

Another way commonly used in the literature [9] to describe a discrete LTI system is the application of a transfer function. For simplicity, we will assume a Single Input Single Output (SISO) system. This results in the system formulation

$$x(k) = G(z, \boldsymbol{\theta})u(k), \tag{1.1}$$

where $G(z, \boldsymbol{\theta})$ is a transfer function from input $u(k) \in \mathbb{R}$ to state $x(k) \in \mathbb{R}$, parametrized by $\boldsymbol{\theta} \in \mathbb{R}^m$, and z denotes a shift operator $zx(k) = x(k+1)$. The right hand side of this equation can be expanded, relating the transfer function to an impulse response of the system $g_n(\boldsymbol{\theta})$

$$G(z, \boldsymbol{\theta})u(k) = \sum_{n=1}^{\infty} g_n(\boldsymbol{\theta}) z^{-n} u(k) = \sum_{n=1}^{\infty} g_n(\boldsymbol{\theta}) u(k-n).$$

By further defining θ_n as the n-th element of $\boldsymbol{\theta}$, and $g_n(\boldsymbol{\theta}) = \theta_n$ for $n = 1, ..., m$ and 0 elsewhere, we adopt a Finite Impulse Response (FIR) model of order m. Pillonetto et al. [10] showed that a discrete LTV system can be described in a similar fashion, by applying a time-varying transfer function

$$G_0(k, z, \boldsymbol{\theta})u(k) = \sum_{n=1}^{\infty} g_n^0(k, \boldsymbol{\theta})z^{-n}u(k) = \sum_{n=1}^{\infty} g_n^0(k, \boldsymbol{\theta})u(k-n). \tag{1.2}$$

Now by defining $g_n^0(k, \boldsymbol{\theta}) = \theta_n(k)$ for $n = 1, ..., m$ and 0 elsewhere, once again we obtain a FIR model of order m. In both presented cases, the dynamics of the system are described by a vector of parameters $\boldsymbol{\theta}$.

Non-linear state-space model

Non-linear systems can be described by the state-space equation

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t),$$

where f is a non-linear function. It turns out that the properties of such systems can be found by analyzing LTI or LTV systems that approximate them. Moreover, non-linear systems can, under some conditions, be controlled with feedback controllers designed for the linearizations of such systems [8].

1.2.2 System identification for reconfigurable spacecraft

The literature [4, 11] identifies the attitude dynamics and kinematics of a spacecraft as the non-linear system

$$\begin{cases} \dot{\mathbf{q}}(t) = \frac{1}{2}\mathbf{q}(t) \otimes \begin{bmatrix} 0 & \boldsymbol{\omega}(t) \end{bmatrix}^T \\ \mathbf{J}(t)\dot{\boldsymbol{\omega}}(t) = \begin{bmatrix} \mathbf{J}(t)\boldsymbol{\omega}(t) \end{bmatrix} \times \boldsymbol{\omega}(t) - \mathbf{u}(t) + \mathbf{T}(t) - \dot{\mathbf{J}}(t)\boldsymbol{\omega}(t) \end{cases},$$

where the attitude with respect to an inertial frame of reference is represented by the quaternion $\mathbf{q} = \begin{bmatrix} q_0 & q_x & q_y & q_z \end{bmatrix}^T$, $\boldsymbol{\omega}(t)$ is the angular velocity of the spacecraft expressed in the body-fixed frame, $\otimes$ represents a quaternion product, $\mathbf{u}(t)$ is torque input, $\mathbf{T}(t)$ are the external and internal disturbances, and $\mathbf{J}(t)$ is the inertia in the spacecraft body frame. Therefore, in order to obtain a mathematical representation of such system, one needs to precisely determine $\dot{\mathbf{J}}(t)$ and model the internal disturbances contributing towards $\mathbf{T}(t)$. In practice, this can be an extremely difficult task, which can be addressed by applying system identification methods.

The task of system identification in this scenario can be approached from two perspectives. We can either try to approximate the non-linear functions representing the system or try to find a linear system which will be a good enough approximation of the non-linear dynamics of the spacecraft.

Approximating the non-linear functions underlying the dynamics of the system is usually done by the means of applying kernel methods, neural networks, Gaussian process estimation, or block-oriented models [9]. In this case, the reconfiguration process will change the functions of interest, requiring one to estimate them again in the new configuration.

Carvalho et al. [4] showed empirically that, under constant inertia, spacecraft dynamics and kinematics can be linearized about a nominal trajectory to obtain a LTV system, which can be identified by applying a particular kind of regularized linear regression. In this case, we expect that the reconfiguration of the system will result in the variation of the parameters of the system, which can be either continuous or discontinuous in time. Therefore, we can look at a reconfiguring system as a special case of an LTV system, where the functions describing parameter values in time have local discontinuities.

Given data describing the state and inputs of the system in time under different trajectories, system identification for reconfigurable spacecraft can be regarded as a regression problem in which we try to either estimate the underlying non-linear functions for each configuration or estimate the parameters of a linear time varying system.

2

State of the art

In this chapter, we explore the existing data-driven solutions to the problem of system identification as well as show the importance of regularization and data conditioning. We aim to establish a better understanding and intuition of the problem. This literature study was helpful for choosing the methods to proceed with, in order to address the problem discussed by this thesis.

To facilitate the understanding of this chapter, we split the discussed system identification methods into two sections: we start by discussing methods based on models described by transfer functions, and continue with methods based on state-space models. Each of the corresponding sections starts with a description of how these models can be extended to real-life scenarios, and then the concrete methods are presented.

The discussed system identification approaches include methods based on autoregressive model with exogenous input (ARX), nonlinear autoregressive model with exogenous input (NLARX) autoregressive moving average model with exogenous input (ARMAX), Finite Impulse Response (FIR) model, as well as eigensystem realization algorithm (ERA), observer/Kalman filter identification (OKID), time-varying eigensystem realization algorithm (TVERA), time-varying observer/Kalman filter identification (TVOKID), identification of LTV dynamical models with smooth or discontinuous evolution by means of convex optimization (LTVModels), Dynamic Mode Decomposition with Control (DMDc), fast closed-form identification from large-scale data for LTV systems (COSMIC), Sparse Identification of Nonlinear Dynamics with Control (SINDYc), and finally methods based on neural networks.

2.1 Transfer function based system identification

In Section 1.2.1 we presented different ways used to represent a dynamical system. In these considerations, however, we did not take into account any disturbances which are crucial for modelling real-world systems. Let's start the analysis by recalling (1.1), describing a transfer function of a discrete Linear Time Invariant (LTI) system. In a real-world system it is reasonable to include an additive disturbance $v(k) \in \mathbb{R}$. For simplicity, in this section, we

will assume that the analyzed system is Single Input Single Output (SISO). The equation describing it becomes

$$x(k) = G(z, \boldsymbol{\theta}_G)u(k) + v(k),$$

where $x(k) \in \mathbb{R}$ is the state of the system, $u(k) \in \mathbb{R}$ is the input to the system, $G(z, \boldsymbol{\theta}_G)$ is a transfer function from u to x parametrized by a vector $\boldsymbol{\theta}_G \in \mathbb{R}^{m_G}$, and z is the shift operator.

This additive noise term $v(k)$ can be considered as white noise $e(k) \in \mathbb{R}$, filtered by a transfer function H parametrized by $\boldsymbol{\theta}_H \in \mathbb{R}^{m_H}$, so that $v(k) = H(z, \boldsymbol{\theta}_H)e(k)$. As a result, the LTI system under disturbance is described by

$$x(k) = G(z, \boldsymbol{\theta}_G)u(k) + H(z, \boldsymbol{\theta}_H)e(k),$$

where $\mathbb{E}\left[e^2(k)\right] = \sigma^2$ and $\mathbb{E}\left[e(k)e(n)\right] = 0$ for $n \neq k$.

Such model representation is used in so-called black-box modelling, which identifies a system purely from collected data. To parametrize transfer functions G and H, they are assumed to be rational in the shift operator, i.e.,

$$G(z, \boldsymbol{\theta}_G) = \frac{B(z)}{F(z)}; \quad H(z, \boldsymbol{\theta}_H) = \frac{C(z)}{D(z)}, \tag{2.1}$$

$$B(z) = b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{n_b} z^{-n_b};$$

$$F(z) = 1 + f_1 z^{-1} + f_2 z^{-2} + \cdots + f_{n_f} z^{-n_f};$$

$$C(z) = 1 + c_1 z^{-1} + c_2 z^{-2} + \cdots + c_{n_c} z^{-n_c};$$

$$D(z) = 1 + d_1 z^{-1} + d_2 z^{-2} + \cdots + d_{n_d} z^{-n_d};$$

where $b_n \in \mathbb{R}$, $f_n \in \mathbb{R}$, $c_n \in \mathbb{R}$, $d_n \in \mathbb{R}$.

We then combine the parameters of these expressions into a vector of coefficients $\boldsymbol{\theta} \in \mathbb{R}^{(n_b+n_f+n_c+n_d)}$, which describes the dynamics of the system [9]

$$\boldsymbol{\theta} = \begin{bmatrix} b_1 & \cdots & f_{n_f} \end{bmatrix}^T.$$

As we presented in (1.2), for a Linear Time Varying (LTV) system we can consider a time-varying transfer function. Therefore for a system of this class we have $B = B(z, k)$, $F = F(z, k)$, $C = C(z, k)$, and $D = D(z, k)$. The vector of parameters becomes time dependent

$$\boldsymbol{\theta}(k) = \begin{bmatrix} b_1(k) & \cdots & f_{n_f}(k) \end{bmatrix}^T. \tag{2.2}$$

It is clear that with this approach, the problem of system identification becomes the problem of finding an optimal $\boldsymbol{\theta}$ or $\boldsymbol{\theta}(k)$.

2.1.1 ARX model

ARX is a very common case in system identification, encountered when $F = D = A$ and $C = 1$ in (2.1). This results in the difference equation for discrete LTI systems

$$x(k) + a_1 x(k-1) + \cdots + a_{n_a} x(k-n_a) = b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) + e(k),$$

where $x(k)$ is the state of the system, $u(k)$ is the input to the system, and $e(k)$ is a zero mean stochastic noise process [12]. This relationship allows us to formulate a predictor of the state of the system

$$\hat{x}(k|\boldsymbol{\theta}) = \boldsymbol{\phi}^T(k)\boldsymbol{\theta}$$
$$\boldsymbol{\phi}^T(k) = \begin{bmatrix} -x(k-1) & -x(k-2) & \cdots & -x(k-n_a) & u(k-1) & \cdots & u(k-n_b) \end{bmatrix}$$
$$\boldsymbol{\theta}^T = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n_a} & b_1 & b_2 & \cdots & b_{n_b} \end{bmatrix}.$$

Given state and inputs of the system in time instants $k = 1, \ldots, N$, we can define the matrices

$$\mathbf{X} = \begin{bmatrix} x(1) \\ x(2) \\ \cdots \\ x(N) \end{bmatrix} \quad \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}^T(1) \\ \boldsymbol{\phi}^T(2) \\ \cdots \\ \boldsymbol{\phi}^T(N) \end{bmatrix} \quad \mathbf{E} = \begin{bmatrix} e(1) \\ e(2) \\ \cdots \\ e(N) \end{bmatrix}, \tag{2.3}$$

which under the assumption that the data was generated by a true model $\boldsymbol{\theta}_0$ leads to the linear regression model

$$\mathbf{X} = \boldsymbol{\Phi}\boldsymbol{\theta}_0 + \mathbf{E}. \tag{2.4}$$

Therefore, system identification for ARX model can be formulated as the Least Squares (LS) optimization problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \|\mathbf{X} - \boldsymbol{\Phi}\boldsymbol{\theta}\|_2^2.$$

The ARX models can approximate any linear system, given that the orders n_a and n_b are sufficiently large. A drawback of this approach, however, is that high-order ARX models are known to suffer from high variance [9].

2.1.2 ARMAX model

ARMAX is another common black-box model. It is obtained when $F = D = A$ in (2.1). This results in the predictor [13]

$$\hat{x}(k|\boldsymbol{\theta}) = \boldsymbol{\phi}^T(k)\boldsymbol{\theta}$$
$$\boldsymbol{\phi}^T(k) = \begin{bmatrix} -x(k-1) & \cdots & -x(k-n_a) & u(k) & \cdots & u(k-n_b) & e(k-1) & \cdots & e(k-n_c) \end{bmatrix}$$
$$\boldsymbol{\theta}^T = \begin{bmatrix} a_1 & \cdots & a_{n_a} & b_1 & \cdots & b_{n_b} & c_1 & \cdots & c_{n_c} \end{bmatrix}.$$

By defining matrices $\mathbf{X}$, $\mathbf{\Phi}$ and $\mathbf{E}$ like in (2.3), a linear regression model is formulated

$$\mathbf{X} = \mathbf{\Phi}\boldsymbol{\theta}_0 + \mathbf{E}.$$

The problem of system identification, therefore, also becomes a LS optimization problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{\Phi}\boldsymbol{\theta}\|_2^2.$$

Similarly to ARX, this model can suffer from high variance when the orders n_a, n_b, and n_c are large.

2.1.3 FIR model

The FIR model is very well studied in the literature. It is encountered when $F = C = D = 1$ in (2.1). This results in the difference equation for LTI systems

$$x(k) = b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) + e(k);$$

So the predictor in this case is

$$\hat{x}(k|\boldsymbol{\theta}) = \boldsymbol{\phi}^T(k)\boldsymbol{\theta}$$
$$\boldsymbol{\phi}^T(k) = \begin{bmatrix} u(k) & \cdots & u(k-n_b) \end{bmatrix}$$
$$\boldsymbol{\theta}^T = \begin{bmatrix} b_1 & \cdots & b_{n_b} \end{bmatrix}.$$

As previously, by defining matrices $\mathbf{X}$, $\mathbf{\Phi}$ and $\mathbf{E}$ like in (2.3), a linear regression model is formulated

$$\mathbf{X} = \mathbf{\Phi}\boldsymbol{\theta}_0 + \mathbf{E}.$$

Once again, the problem of system identification becomes a LS optimization problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{\Phi}\boldsymbol{\theta}\|_2^2.$$

The FIR model can suffer from high variance given that the order n_b is large. Its important property, however, is that smoothness and stability of the desired impulse response can be enforced on the model by applying adequate regularization [9].

2.1.4 NLARX model

NLARX is a nonlinear extension of the ARX model discussed in Section 2.1.1. In this case the difference equation for a discrete system is [14]

$$x(k) = f(x(k-1), \cdots, x(k-n_a), u(k-1), \cdots, u(k-n_b)) + e(k),$$

where f is some nonlinear function. The predictor in this case is

$$\hat{x}(k|\boldsymbol{\theta}) = \psi(\boldsymbol{\phi}^T(k), \boldsymbol{\theta})$$

$$\boldsymbol{\phi}^T(k) = \begin{bmatrix} x(k-1) & x(k-2) & \cdots & x(k-n_a) & u(k-1) & \cdots & u(k-n_b) \end{bmatrix},$$

where ψ is in approximation of function f, parametrized by $\boldsymbol{\theta}$. Once again let's define matrices $\mathbf{X}$, $\boldsymbol{\Phi}$ and $\mathbf{E}$ like in (2.3). This leads to a formulation of a regression model

$$\mathbf{X} = f(\boldsymbol{\Phi}) + \mathbf{E},$$

where f is applied elementwise to $\boldsymbol{\Phi}$. In this case the problem of system identification becomes an optimization problem in which we seek optimal parameters $\boldsymbol{\theta}$, minimizing a loss function $\mathcal{L}$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{X}, \psi(\boldsymbol{\Phi}, \boldsymbol{\theta}), \boldsymbol{\theta}),$$

where ψ is applied elementwise to $\boldsymbol{\Phi}$.

2.2 State-space model based system identification

This section of the thesis starts with methods originating from the field of systems theory i.e., ERA, OKID, TVERA, TVOKID, and LTVModels. Then the discussion focuses on machine learning-based methods such as DMDc, COSMIC, SINDYc, and finally neural networks.

In Section 1.2.1 we presented linear and non-linear discrete state-space models. In the real world the equations describing these systems include an additive noise term. For the linear model we get

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k) + \mathbf{e}(k),$$

where $\mathbf{x}(k) \in \mathbb{R}^n$ is the state of the system, $\mathbf{u}(k) \in \mathbb{R}^k$ is the input to the system, $\mathbf{e}(k) \in \mathbb{R}^n$ is white noise, $\mathbf{A}(k) \in \mathbb{R}^{n \times n}$, and $\mathbf{B}(k) \in \mathbb{R}^{n \times k}$.
The difference equation describing a non-linear model becomes

$$\mathbf{x}(k+1) = f(\mathbf{x}, \mathbf{u}, k) + \mathbf{e}(k), \tag{2.5}$$

where f is a non-linear function.

One can now see that system identification based on state-space model is a problem of either finding optimal matrices $\mathbf{A}(k)$ and $\mathbf{B}(k)$, finding the non-linear function f, or finding a good linear approximation of f.

2.2.1 Eigensystem realization algorithm

ERA [15] is a classical algorithm for identification of LTI systems. It is based on exciting the system of interest with an impulse to obtain an impulse response. Once this is achieved, the impulse response is used to construct a Hankel matrix $\mathbf{H}(0)$ and a shifted Hankel matrix $\mathbf{H}(1)$. It can be showed that these matrices can be decomposed into a product of so called observability and controllability matrices

$$\mathbf{H}(0) = \mathbf{OC},$$

where $\mathbf{O}$ and $\mathbf{C}$ are the observability and controllability matrices respectively. By applying singular value decomposition (SVD) to $\mathbf{H}(0)$, one obtains

$$\mathbf{H}(0) = \mathbf{U}\mathbf{\Gamma}^2\mathbf{V}^T,$$

so equivalently

$$\mathbf{P} = \mathbf{U}\mathbf{\Gamma},$$

$$\mathbf{Q} = \mathbf{\Gamma}\mathbf{V}^T,$$

$$\mathbf{H}(0) = \mathbf{O}\mathbf{C} = \mathbf{P}\mathbf{Q}.$$

This decomposition is further used by ERA to reconstruct the matrices $\mathbf{A}$ and $\mathbf{B}$ of the system.

2.2.2 Observer/Kalman filter identification

OKID [16] is another classical method for LTI system identification. It is applicable for lightly damped systems, for which an impulse response would yield a large Hankel matrix in terms of dimensions, causing numerical errors when applying the ERA algorithm. In OKID, an observer is defined. This observer uses a Kalman filter to estimate the states of the system, excited with an impulse, based on the output measurements. An optimal choice of an observer yields an impulse response that results in Hankel matrices of reasonable size. Then ERA is applied to recover the state-space realizations of the system and corresponding Kalman gains.

2.2.3 Time-varying eigensystem realization algorithm

TVERA [17] is an extension of ERA to LTV systems. This method assumes that the dimension of the state of the system does not change. In TVERA, the state measurements are collected for two sets of experiments — zero initial condition experiments with random inputs to the system sampled from a standard normal distribution, and free response experiments starting from non-zero initial conditions, both for multiple trajectories. Based on the collected data, TVERA builds a Hankel matrix and applies SVD to obtain the matrices $\mathbf{A}(k)$ and $\mathbf{B}(k)$ separately for each set of experiments. The resulting sets of matrices are in different coordinate frames, and the discussed method further takes care of transforming them to a common coordinate frame so that they can be used for state propagation.

2.2.4 Time-varying observer/Kalman filter identification

TVOKID [18] is a time-varying generalization of OKID to the LTV class of systems. Similarly to OKID it helps when a system is lightly damped by introducing an asymptotically stable observer (assumed as a time-varying deadbeat observer). As a result, we obtain smaller Hankel matrices, which eases the computations and numerical errors. The Markov

parameters identified by TVOKID are further used to identify system matrices $\mathbf{A}(k)$ and $\mathbf{B}(k)$ by applying the TVERA algorithm.

2.2.5 Identification of LTV dynamical models with smooth or discontinuous evolution by means of convex optimization

LTVModels [19] is another algorithm for the identification of LTV systems. This method uses only a single trajectory to identify a system. The measurements of the states and inputs to the system for each discrete time index from 0 to $T-1$ are combined into matrices

$$\mathbf{\Phi}(k) = \begin{bmatrix} \mathbf{x}^T(k) & \mathbf{u}^T(k) \end{bmatrix},$$

and further into a block diagonal matrix

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}(0) & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{\Phi}(1) & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \mathbf{\Phi}(T-2) & 0 \\ 0 & 0 & \cdots & 0 & \mathbf{\Phi}(T-1) \end{bmatrix}.$$

The matrices $\mathbf{A}(k)$ and $\mathbf{B}(k)$ are merged for each discrete time index from 0 to $T-1$

$$\mathbf{K}(k) = \begin{bmatrix} \mathbf{A}^T(k) \\ \mathbf{B}^T(k) \end{bmatrix},$$

and further into a single matrix

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}(0) \\ \cdots \\ \mathbf{K}(T-1) \end{bmatrix}.$$

Finally the states of the system for time indices from 1 to T are combined into a matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{x}^T(1) \\ \cdots \\ \mathbf{x}^T(T) \end{bmatrix}.$$

The algorithm poses the problem of system identification as

$$\hat{\mathbf{K}} = \underset{\mathbf{K}}{\operatorname{argmin}} \|\mathbf{\Phi K} - \mathbf{Y}\|_2^2 + \sum_{k=0}^{T-2} f(\mathbf{K}(k+1), \mathbf{K}(k)),$$

where $f(\mathbf{K}(k+1), \mathbf{K}(k))$ is a function which serves the role of regularization, penalizing for the variability of system matrices between consecutive time instants. Different forms of function f are defined depending on the concrete case of the considered system.

2.2.6 Dynamic mode decomposition with control

DMDc [20] is a state-space model based system identification method for discrete LTI systems with control input. To introduce it, let's define the matrices

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}(1) & \mathbf{x}(2) & \cdots & \mathbf{x}(N-1) \\ | & | & & | \end{bmatrix},$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}(2) & \mathbf{x}(3) & \cdots & \mathbf{x}(N) \\ | & | & & | \end{bmatrix},$$

$$\mathbf{\Upsilon} = \begin{bmatrix} | & | & & | \\ \mathbf{u}(1) & \mathbf{u}(2) & \cdots & \mathbf{u}(N-1) \\ | & | & & | \end{bmatrix},$$

$$\mathbf{E} = \begin{bmatrix} | & | & & | \\ \mathbf{e}(1) & \mathbf{e}(2) & \cdots & \mathbf{e}(N-1) \\ | & | & & | \end{bmatrix}.$$

We are considering an LTI system, so $\mathbf{A}(k) = \mathbf{A}$ and $\mathbf{B}(k) = \mathbf{B}$. Therefore the system can be described as

$$\mathbf{X}' = \mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{\Upsilon} + \mathbf{E},$$

resulting in the linear regression model

$$\mathbf{X}' = \mathbf{G}\mathbf{\Omega} + \mathbf{E},$$

where $\mathbf{G} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix}$ and $\mathbf{\Omega} = \begin{bmatrix} \mathbf{X} \\ \mathbf{\Upsilon} \end{bmatrix}$.
The problem of system identification under DMDc, therefore, becomes the optimization problem

$$\hat{\mathbf{G}} = \underset{\mathbf{G}}{\operatorname{argmin}} \|\mathbf{X}' - \mathbf{G}\mathbf{\Omega}\|_F^2,$$

where $\|\cdot\|_F$ is the Frobenius norm.

2.2.7 COSMIC

COSMIC [4] is a closed-form method for identification of discrete LTV systems based on the state-space model. It can be considered as a regularized extension of DMDc to time-varying systems. For our considerations we will assume that the data was collected from L different trajectories, each composed of N timesteps. Let's define $\mathbf{x}_l(k) \in \mathbb{R}^n$ as the system state at time instant k for trajectory l and $\mathbf{u}_l(k) \in \mathbb{R}^k$ as the system input under the same meaning in terms of time and trajectory. Then we create the matrices

$$\mathbf{X}(k) = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1(k) & \mathbf{x}_2(k) & \cdots & \mathbf{x}_L(k) \\ | & | & & | \end{bmatrix},$$

$$\mathbf{X}'(k) = \mathbf{X}(k+1),$$

$$\mathbf{U}(k) = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1(k) & \mathbf{u}_2(k) & \cdots & \mathbf{u}_L(k) \\ | & | & & | \end{bmatrix},$$

$$\mathbf{C}(k) = \begin{bmatrix} \mathbf{A}^T(k) \\ \mathbf{B}^T(k) \end{bmatrix},$$

$$\mathbf{D}(k) = \begin{bmatrix} \mathbf{X}^T(k) & \mathbf{U}^T(k) \end{bmatrix},$$

and further define

$$\mathbf{C} = \Big[\mathbf{C}(k)\Big]_{k=1}^{k=N-1} = \begin{bmatrix} \mathbf{C}(1) \\ \cdots \\ \mathbf{C}(N-1) \end{bmatrix},$$

$$\mathbf{V} = \Big[\mathbf{D}(k)\Big]_{k=1}^{k=N-1} = \begin{bmatrix} \mathbf{D}(1) & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{D}(2) & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \mathbf{D}(N-2) & 0 \\ 0 & 0 & \cdots & 0 & \mathbf{D}(N-1) \end{bmatrix}.$$

This allows us to formulate the system identification problem as

$$\hat{\mathbf{C}} = \underset{\mathbf{C}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{V}\mathbf{C} - \mathbf{X}'\|_F^2 + \frac{1}{2}\sum_{k=2}^{k=N-1} \|\lambda_k^{\frac{1}{2}}(\mathbf{C}(k) - \mathbf{C}(k-1))\|_F^2,$$

where $\lambda_k \in \mathbb{R}, \lambda_k > 0$ is a regularization constant for time step k, which serves the purpose of limiting the variability of the system between two consecutive time instants.

The authors of COSMIC provide a closed form solution to this problem, which scales linearly with the number of time steps considered. They also define conditions which must be satisfied by the data so that a valid solution is obtained. Moreover, a preconditioning step is proposed to address the problem of ill-conditioned data. All this makes COSMIC a good candidate for applications in the space industry where reliability and robustness are crucial.

2.2.8 SINDYc

SINDYc [21] is a generalization of the original SINDY [22] to include control inputs. This system identification method is based on sparse regression and identifies fully non-linear models from the data. To introduce SINDYc, let's consider the same $\mathbf{X}$, $\mathbf{X}'$, $\mathbf{E}$, and $\boldsymbol{\Upsilon}$ matrices as in Section 2.2.6. Now let $\boldsymbol{\Theta}^T(\mathbf{X}, \boldsymbol{\Upsilon})$ be a library of candidate nonlinear functions, including nonlinear cross terms in $\mathbf{x}$ and $\mathbf{u}$

$$\boldsymbol{\Theta}(\mathbf{X}, \boldsymbol{\Upsilon}) = \begin{bmatrix} \mathbf{1}^T & \mathbf{X}^T & \boldsymbol{\Upsilon}^T & (\mathbf{X} \otimes \mathbf{X})^T & (\mathbf{X} \otimes \boldsymbol{\Upsilon})^T & \cdots & \sin(\mathbf{X})^T & \sin(\boldsymbol{\Upsilon})^T & \sin(\mathbf{X} \otimes \boldsymbol{\Upsilon})^T & \cdots \end{bmatrix}.$$

Under this notation, we can formulate the system as

$$\mathbf{X}' = \boldsymbol{\Xi}\boldsymbol{\Theta}^T(\mathbf{X}, \boldsymbol{\Upsilon}) + \mathbf{E},$$

where $\boldsymbol{\Xi}$ is a matrix of coefficients to be learned.

In SINDYc we assume that $\boldsymbol{\Xi}$ is sparse. The resulting sparse regression problem is solved for each row $\boldsymbol{\xi}_m$ of $\boldsymbol{\Xi}$, using a Lasso L1 penalty on the parameters $\boldsymbol{\Xi}$

$$\hat{\boldsymbol{\xi}}_m = \underset{\boldsymbol{\xi}_m}{\operatorname{argmin}} \|\mathbf{X}'_m - \boldsymbol{\xi}_m \boldsymbol{\Theta}^T(\mathbf{X}, \boldsymbol{\Upsilon})\|_2 + \alpha \|\boldsymbol{\xi}_m\|_1,$$

where $\mathbf{X}'_m$ is the m-th row of $\mathbf{X}'$, and $\alpha \in \mathbb{R}$ is a regularization parameter balancing model accuracy and complexity.

2.2.9 Neural networks

Under the universal approximation theorem [23], neural networks can represent a wide variety of different functions. Therefore, they have a potential of modelling the f function in the nonlinear state-space (2.5). There is a vast list of neural network architectures applied to the problem of system identification, including feedforward neural networks [24], long-short term memory neural networks [25], physics-informed neural networks [26], and many more.

For an application to system identification, we define the input-output training pairs as

$$\mathbf{X}_{in}^k = \begin{bmatrix} | \\ \mathbf{x}(k) \\ | \\ | \\ \mathbf{u}(k) \\ | \\ k \end{bmatrix},$$

$$\mathbf{X}_{out}^k = \begin{bmatrix} | \\ \mathbf{x}(k+1) \\ | \end{bmatrix}.$$

The system is formulated as

$$\mathbf{X}^k_{out} = \psi(\mathbf{X}^k_{in}, \boldsymbol{\theta}) + \mathbf{e}(k) \quad \text{for} \quad k = 1, 2, ..., N-1,$$

where ψ is the function approximated by a neural network, parametrized by weights and biases $\boldsymbol{\theta}$. The system identification problem, therefore, becomes an optimization problem, in which we seek optimal parameters $\boldsymbol{\theta}$, minimizing a loss function $\mathcal{L}$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{k=1}^{k=N-1} \mathcal{L}(\mathbf{X}^k_{out}, \psi(\mathbf{X}^k_{in}, \boldsymbol{\theta}), \boldsymbol{\theta}).$$

Such approach results in a model which learns to predict the state of the system one step ahead. The predicted state can be fed into the network recursively to simulate for more timesteps into the future.

One of the biggest issues related to training neural networks are their low sample-efficiency and high variance, limiting their generalization possibilities in the low-data limit.

2.3 Regularization

The majority of data driven methods for system identification described in Sections 2.1 and 2.2 need to balance between model order, which enables a model to approximate a wider range of functions, and model variance which causes a model to overfit. This problem of bias-variance tradeoff is well-known and studied in statistics and learning theory. A common way to address this issue is to use regularization. Some of the described methods such as COSMIC and SINDYc already come with problem-specific regularization. In COSMIC we have a term which penalizes for large model variability between time instants, whereas in SINDYc the regularization favors sparse solutions. For other models we can apply well-known regularization methods, which we will review in this section.

All the system identification methods that we described in this chapter can be divided into two classes, i.e., the ones based on linear regression, and the ones based on neural networks. Therefore we split the discussion of regularization into two sections. The first one focuses on linear regression models, and the other one is specific to neural networks.

2.3.1 Regularization in linear regression

For convenience, let's consider linear regression for system identification in matrix form as we did in 2.1.1. For this purpose we define

$$\mathbf{X} = \begin{bmatrix} x(1) \\ x(2) \\ \cdots \\ x(N) \end{bmatrix} \quad \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}^T(1) \\ \boldsymbol{\phi}^T(2) \\ \cdots \\ \boldsymbol{\phi}^T(N) \end{bmatrix} \quad \mathbf{E} = \begin{bmatrix} e(1) \\ e(2) \\ \cdots \\ e(N) \end{bmatrix}.$$

Given that the observed data was generated by true model $\boldsymbol{\theta}_0$, the system can be reformulated as

$$\mathbf{X} = \boldsymbol{\Phi}\boldsymbol{\theta}_0 + \mathbf{E}.$$

Under the assumption of normally distributed and independent residuals $\mathbf{E}$, the solution for $\boldsymbol{\theta}$ under the Maximum Likelihood Estimation (MLE) coincides with the solution to the LS problem of minimizing the Mean Squared Error (MSE) and can be calculated analytically

$$\hat{\boldsymbol{\theta}}^{MLE} = \hat{\boldsymbol{\theta}}^{LS} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \|\mathbf{X} - \boldsymbol{\Phi}\boldsymbol{\theta}\|_2^2 = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{X}.$$

The idea of regularization with quadratic penalties is achieved by penalizing large absolute parameter values of the estimator, so that the regularized linear regression can be expressed as the following optimization problem with an analytical solution

$$\hat{\boldsymbol{\theta}}^{R} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}(\|\mathbf{X} - \boldsymbol{\Phi}\boldsymbol{\theta}\|_2^2 + \gamma\boldsymbol{\theta}^T\mathbf{P}^{-1}\boldsymbol{\theta}) = (\mathbf{P}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \gamma\mathbf{I}_n)^{-1}\mathbf{P}\boldsymbol{\Phi}^T\mathbf{X},$$

where $\gamma \geq 0$ is a regularization parameter, $\mathbf{I}_n$ is an n-dimensional identity matrix, and $\mathbf{P}$ is a positive definite matrix, thus invertible, and often called regularization matrix.

When the regularization matrix $\mathbf{P}$, and the regularization constant γ are suitably chosen, one can achieve a decrease in variance of $\hat{\boldsymbol{\theta}}^{R}$. If the resulting increase in bias is moderate, the resulting MSE can be smaller than the one achieved for $\hat{\boldsymbol{\theta}}^{MLE}$ [9].

It can be showed that the optimal regularization matrix is $\mathbf{P} = \boldsymbol{\theta}_0\boldsymbol{\theta}_0^T$, and the regularization parameter $\gamma = \sigma^2$, where σ^2 is the noise variance in data. This cannot be used in practice but it suggests that the regularization matrix should be chosen so that it mimics the behaviour of $\boldsymbol{\theta}_0\boldsymbol{\theta}_0^T$. This way prior knowledge about the system can be incorporated to the problem [9].

2.3.2 Regularization in neural networks

There is a vast amount of regularization techniques used to balance the bias-variance tradeoff in neural networks. In this section we will assume that the reader has a basic knowledge of neural networks and review the most common regularization methods.

L1 and L2 regularization

Neural networks are parametrized by a set of weights W and biases B. The idea of L1 or L2 regularization is to add a loss term which penalizes for high L1 or L2 norms, respectively, of the weights of the network. Therefore, given $\mathcal{L}$ as a loss function without regularization, the loss with L1 regularization becomes

$$\mathcal{L}_{L1} = \mathcal{L} + \gamma \sum_{w \in W} |w|,$$

where γ is the regularization coefficient. Similarly, the loss with L2 regularization becomes

$$\mathcal{L}_{L2} = \mathcal{L} + \gamma \sum_{w \in W} w^2.$$

Dropout

The idea of dropout is to randomly switch off certain neurons during the training of a neural network. This is achieved by setting the activation of these neurons to 0. The activation of the remaining neurons is adjusted to compensate for the ones which were off. Dropout is parametrized by a probability p of switching off a single neuron. With dropout, the neuron activation h becomes

$$h' = \begin{cases} 0 & \text{with probability } p \\ \frac{h}{1-p} & \text{otherwise} \end{cases}.$$

This way, the resulting model is more robust to perturbations, reducing its variance [27].

Batch normalization

Although designed to accelerate the convergence of deep neural networks, regularization is a secondary benefit of batch normalization. Given an input $\mathbf{x}$ to a batch normalization layer, the following operation is performed

$$\mathbf{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}_B}{\hat{\sigma}_B} + \beta,$$

where the parameters γ and β are learned during training, $\odot$ represents elementwise multiplication, $\hat{\mu}_B$ and $\hat{\sigma}_B$ are the mean and standard deviation calculated for the minibatch B during training. During inference, these are replaced by dataset statistics [27].

Encoding physical knowledge

Another way to regularize neural networks and increase their sample-efficiency is to encode physical knowledge of the modeled system. This is especially useful for system identification, where it is common to know certain differential equations related to physical laws which must be obeyed by the system.

Physics-informed Neural Networks (PINNs), proposed by Lagaris et al. [28] are one way of encoding physical knowledge in a neural network. To introduce it, let's assume that we are interested in predicting variable $x(t, u)$. Let the differential equation that we want to impose on the solution be

$$\dot{x} + \mathcal{N}[x; u; \lambda] = 0, \tag{2.6}$$

where $\mathcal{N}$ is a nonlinear differential operator parametrized by λ. Based on this equation we define the function

$$f := \dot{x} + \mathcal{N}[x; u; \lambda]. \tag{2.7}$$

And train a neural network for approximating $x(t, u)$. The values of function f are calculated by automatic differentiation. The unknown parameter values of $\mathcal{N}$ are included

as parameters of the network. Then a loss function $\mathcal{L}$ is defined as

$$\mathcal{L} = \frac{1}{N_x}\sum_{i=1}^{N_x} \|x(t_x^i, u_x^i) - x^i\|^2 + \eta\left(\frac{1}{N_f}\sum_{j=1}^{N_f} \|f(t_f^j, x_f^j, u_f^j)\|^2 + \frac{1}{N_{ic}}\sum_{k=1}^{N_{ic}} \|x(t_{ic}^k, u_{ic}^k) - x_{ic}^k\|^2\right), \tag{2.8}$$

where $x(t_x^i, u_x^i)$ is the i-th network prediction, x^i is the i-th ground truth value for training data with $i = 1, ..., N_x$. $f(t_f^j, x_f^j, u_f^j)$ is the j-th value of f for the points of evaluation of the differential equation, such that $j = 1, ..., N_f$. $x(t_{ic}^k, u_{ic}^k)$ is the k-th network prediction for initial conditions data, with ground truth value x_{ic}^k, such that $k = 1, ..., N_{ic}$. η is a parameter determining the importance of following the differential equation(s).

The loss defined this way has a data fidelity term: $\frac{1}{N_x}\sum_{i=1}^{N_x} \|x(t_x^i, u_x^i) - x^i\|^2$, and a regularization term $\eta\left(\frac{1}{N_f}\sum_{j=1}^{N_f} \|f(t_f^j, x_f^j, u_f^j)\|^2 + \frac{1}{N_{ic}}\sum_{k=1}^{N_{ic}} \|x(t_{ic}^k, u_{ic}^k) - x_{ic}^k\|^2\right)$. The resulting unconstrained optimization problem can be interpreted as a penalty method equivalent to the constrained problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N_x} \|x(t_x^i, u_x^i) - x^i\|^2 \\ \text{subject to} \quad & \dot{x} + \mathcal{N}[x; u; \lambda] = 0. \end{aligned}$$

2.4 Ill-conditioned problems

In this chapter, we reviewed multiple system identification methods based on linear regression. In such problems, if the condition number of the data matrix $\mathbf{\Phi}$ as in (2.4) is large, the estimate of the parameters becomes very sensitive to perturbations in data. To identify a linear system via linear regression, it is therefore crucial to address this issue. One of the possible solutions is to apply various preconditioning methods to the data matrix [29]. Another possibility is to apply regularization with quadratic penalties, which under proper choice of the regularization parameters, can make the problem well-conditioned [9].

3

Modelled system

In this chapter we present a spring mass damper system, which we simulated with Simulink™ in order to obtain data further used for system identification with methods selected from Chapter 2.

We propose five different scenarios for the system: Linear Time Varying (LTV), nonlinear (NL), nonlinear perturbed (NLD), LTV instantaneous reconfiguration, and LTV reconfiguration. The described approach enabled us to initially model the complexities related to system identification of reconfigurable spacecraft using a relatively simple and fast simulation.

3.1 Spring mass damper system

In this section we present the five spring mass damper system scenarios that were used to initially simulate reconfiguring spacecraft. We start with an introduction of the simplest case, where the parameters of the system are constant, yielding a Linear Time Invariant (LTI) system. Later on, the consecutive subsections introduce different changes to this basic case, explaining each of the simulated scenarios.

The considered spring mass damper system (see Figure 3.1) is characterized by three physical quantities: mass m, spring constant C_s, and damping constant C_d. We decided to not model gravity acting on the mass, as it would only introduce a bias added to the resulting forces acting on the system. Therefore, in a general case we can distinguish two internal forces acting on the system: a spring force, which according to the Hooke's law can be expressed as

$$F_s(t) = C_s x(t),$$

where $x(t)$ is the mass position in relation to the resting point at $x = 0$; and a damping force, in this simplest case proportional to the velocity of the mass

$$F_d(t) = C_d v(t) = C_d \dot{x}(t),$$

where $v(t) = \dot{x}(t)$ is the velocity of the mass, equal to the first order time derivative of $x(t)$. Moreover, we model the effects of external forces acting on the system as $ext(u(t)) = u(t)$,

where $u(t)$ is the resultant external force. By applying Newton's second law of motion we obtain an ordinary differential equation (ODE)

$$m\ddot{x}(t) = -C_d\dot{x}(t) - C_s x(t) + u(t), \tag{3.1}$$

where $\ddot{x}(t)$ is the acceleration of the mass equal to the second order time derivative of $x(t)$. To represent this system with a continuous-time state space model, we denote

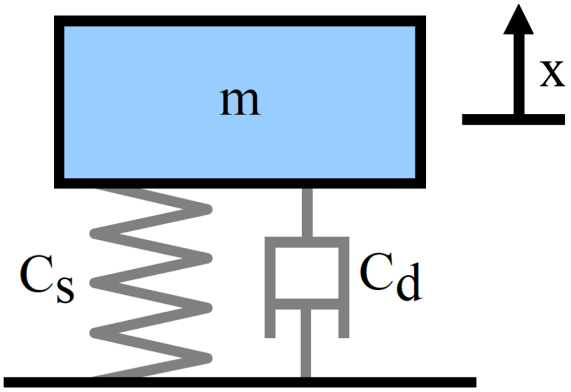

Figure 3.1: Schematic of a typical spring mass damper system.

$$\mathbf{x}(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

as the state. This way we obtain the desired representation

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{C_s}{m} & -\frac{C_d}{m} \end{bmatrix}}_{\mathbf{A}_c} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{\mathbf{B}_c} u(t). \tag{3.2}$$

Finally, we can represent the system using a discrete-time state space model

$$\mathbf{x}(k+1) = \underbrace{e^{\mathbf{A}_c \Delta t}}_{\mathbf{A}} \mathbf{x}(k) + \underbrace{\mathbf{A}_c^{-1}(e^{\mathbf{A}_c \Delta t} - \mathbf{I})\mathbf{B}_c}_{\mathbf{B}} u(k), \tag{3.3}$$

where Δt is the sampling interval, and $\mathbf{x}(k)$, $u(k)$ represent the state and inputs to the system at time $k\Delta t$.

3.1.1 LTV scenario

The simplest case of the spring mass damper system that we simulated is the LTV scenario. In this simulation the spring and damping constants (C_s and C_d) became time dependent parameters of the system. For this purpose we defined

$$\begin{cases} C_s(t) = \cos\left(1.5\omega_0 t + \frac{\pi}{4}\right)^2 C_s \\ C_d(t) = \left[1.5 + \cos\left(\omega_0 t\right)\right] C_d \end{cases}. \tag{3.4}$$

This modification introduced a modification to the ODE defined in (3.1)

$$m\ddot{x}(t) = -C_d(t)\dot{x}(t) - C_s(t)x(t) + u(t), \tag{3.5}$$

changing the state-space representations of the system ((3.2), (3.3)) to

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{C_s(t)}{m} & -\frac{C_d(t)}{m} \end{bmatrix}}_{\mathbf{A}_c(t)} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{\mathbf{B}_c} u(t), \tag{3.6}$$

$$\mathbf{x}(k+1) = \underbrace{e^{\mathbf{A}_c(k)\Delta t}}_{\mathbf{A}(k)} \mathbf{x}(k) + \underbrace{\mathbf{A}_c^{-1}(k)(e^{\mathbf{A}_c(k)\Delta t} - \mathbf{I})\mathbf{B}_c}_{\mathbf{B}(k)} u(k). \tag{3.7}$$

3.1.2 NL scenario

To pose a more challenging system identification problem, a NL scenario was defined, introducing a nonlinear damping force to the setup described in Subsection 3.1.1

$$F_d(t) = C_d(t)\dot{x}(t) + C_{d3}\dot{x}^3(t),$$

and a saturation of the input

$$ext(u(t)) = sat_U(u(t)).$$

This resulted in another modification of the ODE presented in (3.5)

$$m\ddot{x}(t) = -C_d(t)\dot{x}(t) - C_{d3}\dot{x}^3(t) - C_s(t)x(t) + sat_U(u(t)), \tag{3.8}$$

changing the continuous-time state space representations of the system (3.6) to

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{C_s(t)}{m} & -\frac{C_d(t)-C_{d3}x_2^2(t)}{m} \end{bmatrix}}_{\mathbf{A}_c(t)} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{\mathbf{B}_c} sat_U(u(t)),$$

resulting in a linearization and discretization equivalent to (3.7).

3.1.3 NLD scenario

To further enhance the NL scenario, an impulsive stochastic disturbance $d(x_1(t))$ dependent on the distance of the mass to a particular point ($x_1 = 2$) is added to $x_2(t)$ resulting in a NLD scenario. In this case the ODE of the system becomes

$$m\ddot{x}(t) = -C_d(t)\dot{x}(t) - C_{d3}\dot{x}^3(t) - C_s(t)x(t) + sat_U(u(t)) + md(x_1(t)). \tag{3.9}$$

The continuous-time state space representation of this system is now

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{C_s(t)}{m} & -\frac{C_d(t)-C_{d3}x_2^2(t)}{m} \end{bmatrix}}_{\mathbf{A}_c(t)} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{\mathbf{B}_c} sat_U(u(t)) + \begin{bmatrix} 0 \\ d(x_1(t)) \end{bmatrix},$$

resulting in a linearization and discretization equivalent to (3.7).

3.1.4 LTV instantaneous reconfiguration scenario

To simulate a scenario of instantaneous reconfiguration, we go back in the considerations to the simplest spring mass damper system as described in Section 3.1. We introduce the following modification to the physical parameters of the system: C_d, C_s, and m are kept constant within predefined two-second-long time intervals and change instantaneously between them. Therefore we can state that these parameters are dependent on the index of current two-second-long time interval, i.e., we have $C_d(i)$, $C_s(i)$, and $m(i)$, where $i = \left\lfloor \frac{t}{2} \right\rfloor$. Moreover, each change of the values of these parameters is accompanied by a stochastic disturbance of the system's velocity $d_v(t)$, which attains non-zero values only when transitioning to a new two-second-long time frame. Now the ODE of the system becomes

$$m(i)\ddot{x}(t) = -C_d(i)\dot{x}(t) - C_s(i)x(t) + u(t) + m(i)d_v(t). \tag{3.10}$$

The continuous-time state space representation of the system is

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{C_s(i)}{m(i)} & -\frac{C_d(i)}{m(i)} \end{bmatrix}}_{\mathbf{A}_c(t)} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m(i)} \end{bmatrix}}_{\mathbf{B}_c(t)} u(t) + \begin{bmatrix} 0 \\ d_v(t) \end{bmatrix},$$

and its linear discrete version is given by

$$\mathbf{x}(k+1) = \underbrace{e^{\mathbf{A}_c(k)\Delta t}}_{\mathbf{A}(k)} \mathbf{x}(k) + \underbrace{\mathbf{A}_c^{-1}(k)(e^{\mathbf{A}_c(k)\Delta t} - \mathbf{I})\mathbf{B}_c(k)}_{\mathbf{B}(k)} u(k). \tag{3.11}$$

3.1.5 LTV reconfiguration scenario

In this final scenario, we simulate mixed reconfiguration, i.e., the one in which the properties of the system change both continuously and abruptly. To achieve this we combine the approaches described in Sections 3.1.1 and 3.1.4. The parameters C_s and C_d follow the time-dependent variations as defined in (3.4) inside every two-second-long time interval. Between these time frames, m, C_s, and C_d change instantaneously. Once again each instantaneous parameter change is accompanied by a stochastic disturbance of the system's velocity $d_v(t)$, which attains non-zero values only when transitioning to a new two-second-long time frame. The ODE of the system is now

$$m(i)\ddot{x}(t) = -C_d(t)\dot{x}(t) - C_s(t)x(t) + u(t) + m(i)d_v(t). \tag{3.12}$$

The continuous-time state space representation of the system is

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{C_s(t)}{m(i)} & -\frac{C_d(t)}{m(i)} \end{bmatrix}}_{\mathbf{A}_c(t)} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m(i)} \end{bmatrix}}_{\mathbf{B}_c(t)} u(t) + \begin{bmatrix} 0 \\ d_v(t) \end{bmatrix},$$

where

$$\begin{cases} C_s(t) = \cos\left(1.5\omega_0 t + \frac{\pi}{4}\right)^2 C_s(i) \\ C_d(t) = \left[1.5 + \cos\left(\omega_0 t\right)\right] C_d(i) \end{cases} . \tag{3.13}$$

Its linear discrete-time version is given by (3.11).

3.2 Relation to reconfigurable spacecraft

In this section we elaborate further on the claim that the systems described in Section 3.1 model the complexities related to system identification of reconfigurable spacecraft.

As previously stated in Chapter 1, the rotational dynamics and kinematics of spacecraft yield a non-linear system, which can be linearized about a reference trajectory to obtain an LTV system. This draws a conclusion that the LTV scenario described in Section 3.1.1 is a good baseline to model spacecraft not undergoing reconfiguration.

Moreover the nonlinear effects and disturbances introduced in Sections 3.1.2 and 3.1.3 can be interpreted as nonlinear terms that arise in real-world systems, as well as disturbances that unavoidably act on spacecraft during operation. Therefore, these simulations introduce important features that increase the similarity between our simple simulation and real spacecraft in operation.

Finally the abrupt and continuous changes of the properties of system introduced in Sections 3.1.4 and 3.1.5 introduce the scenario of reconfiguration, corresponding to a situation where, for example, a robotic manipulator is moving along the spacecraft and performing an assembly. In such case the inertia of the system changes gradually during the movement of the manipulator, and abruptly when a new piece is attached to the spacecraft.

4

System identification experiments

In Chapter 3 we introduced a spring mass damper system that we simulated under five different scenarios representing challenges related to the modelling of reconfigurable spacecraft. In this chapter we further present the experimental setup related to data collection, system identification, and finally validation of the obtained models.

4.1 Data collection

In this section we explain the process in which the training, validation, and testing datasets were obtained for each scenario using the developed Simulink™ simulation. A general setup for system identification is presented in Figure 4.1. It consists of three parts, i.e., excitation, a ground truth system, and state measurements.

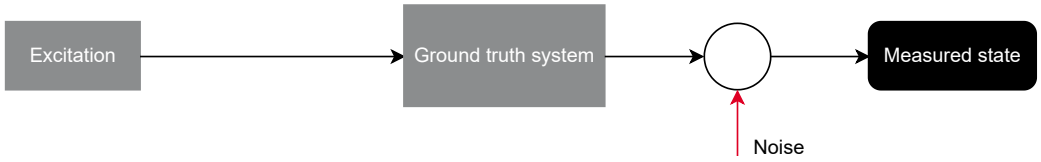

Figure 4.1: Schematic of the experimental setup for data collection.

4.1.1 Excitation

A general property of data-driven supervised methods is the need for usage of sufficiently varied and informative datasets in the training process of the algorithm. If the collected data contains mistakes or features that are not informative enough, the trained model will not be able to generalize well. This becomes especially true for the case of modern deep learning, where apart from the quality of datasets, it is also their size that influences the final accuracy of the model [27].

Without surprise, the informativeness of a training dataset is also crucial for data-driven system identification methods. Development of an informative dataset in this

case is often based on applying the concept of persistent excitation, i.e., supplying the investigated open loop system with a persistently exciting input signal [30].

Given the above considerations, for the process of data collection, our open loop spring mass damper system was excited with a disturbed chirp input signal

$$u(t) = A \sin\left(\frac{\omega_1 - \omega_0}{T} t^2 + \omega_0 t + \phi\right) + d_c(t), \tag{4.1}$$

where ω_1, ω_0 are the maximal and minimal frequencies of the input signal, T is the duration of the whole simulation, ϕ is phase, $d_c(t)$ is a Gaussian stochastic signal with mean 0 and variance σ^2, and t is time.

4.1.2 Ground truth system

For each of the scenarios described in Chapter 3 we simulated the spring mass damper system using a corresponding continuous-time state space representation. We varied the initial conditions, i.e., the state vector $\mathbf{x}(0)$ and phase ϕ of the chirp signal (4.1) between different experiments for a given system, producing various trajectories.

4.1.3 State measurements

The last building block of the data collection process were the state measurements. To simulate a real-world scenario in which measurement noise is inevitable, we introduced an optional additive signal sampled from a Gaussian normal distribution to slightly corrupt the measurements. Note that this noise was optional and included only in the development of the training dataset.

4.2 Training, validation, and testing datasets

In Section 4.1 we explained the process of data collection. We will now focus on how the concepts introduced there were applied to the creation of the training, validation, and testing datasets. Moreover, we will explain the purpose of each dataset. In general we can state that each dataset contained data regarding the position x_1, velocity x_2, input signal u to the system, and time t .

4.2.1 Training datasets

The training dataset was used for the purpose of system identification. Depending on the applied method, however, different requirements had to be imposed.

COSMIC training datasets

In the most general case, used for system identification with fast closed-form identification from large-scale data for LTV systems (COSMIC), the training dataset consisted of 100 trajectories. In each simulation the initial conditions were sampled from the uniform

distributions $U(0,5)[m]$ for position $x_1(0)$ and $U(-1,1)[\frac{m}{s}]$ for velocity $x_2(0)$. The phase ϕ of the input signal was sampled from another uniform distribution $U(0,2\pi)[rad]$. The variance of the noise injected to the chirp signal was $\sigma^2 = 3$.

Single trajectory training datasets

For system identification with method identification of LTV dynamical models with smooth or discontinuous evolution by means of convex optimization (LTVModels) and single trajectory version of COSMIC, we could only use a single simulation. Therefore, we reduced the COSMIC training dataset to only include the first recorded trajectory.

TVERA/TVOKID training datasets

Methods such as time-varying eigensystem realization algorithm (TVERA) and time-varying observer/Kalman filter identification (TVOKID), as mentioned in the corresponding sections of Chapter 2, require a training dataset consisting of free response experiments with non-zero initial conditions and experiments with random input signal and zero initial conditions. Therefore for their purpose we produced a training dataset consisting of 50 trajectories with $u(t) = 0$ and initial conditions sampled from the same uniform distributions as in the case of COSMIC; and another 50 trajectories with $x_1(0) = x_2(0) = 0$ and inputs sampled from a standard normal distribution.

NN training datasets

Finally for the training of neural network-based methods we merged the COSMIC and TVERA/TVOKID datasets to produce the most varied one, as these algorithms require the greatest amounts of data. For these datasets, the positions and velocities of each system are visualized in Figure 4.2. The inputs contained in this dataset are presented in Figure 4.3.

4.2.2 Validation datasets

Given that the applied system identification methods are characterized by different hyperparameters, it becomes crucial to properly tune them. For the purpose of hyperparameter tuning, for each scenario of a spring mass damper system, we created a validation dataset. These validation datasets contained 15 trajectories each: 5 trajectories for free response experiments from non-zero initial conditions sampled as in 4.2.1, 5 trajectories for experiments with zero initial conditions and disturbed chirp input signal, and finally 5 trajectories for experiments with non-zero initial conditions sampled as in 4.2.1 and disturbed chirp input signal.

The free response trajectories allowed us to verify if the estimated matrices $\mathbf{A}(k)$ were correctly identified, the zero initial condition trajectories verified the correctness of the estimated $\mathbf{B}(k)$ matrices, as for the initial steps of these experiments they had the most

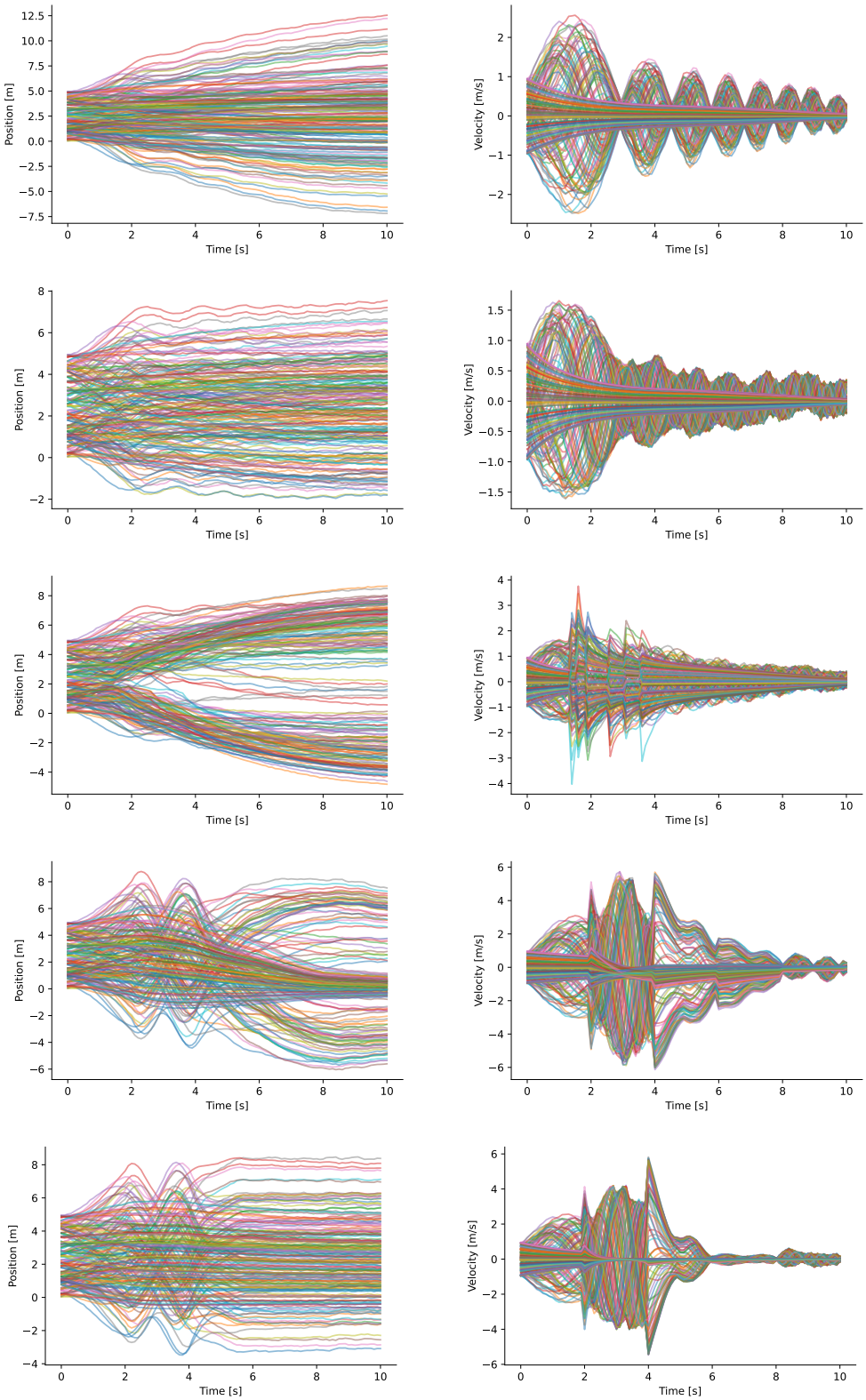

Figure 4.2: NN training dataset position and velocity visualizations for spring mass damper system in the discussed scenarios. Colors represent different trajectories. Left to right: position and velocity. Top to bottom: LTV, NL, NLD, LTV instantaneous reconfiguration, LTV reconfiguration scenarios.

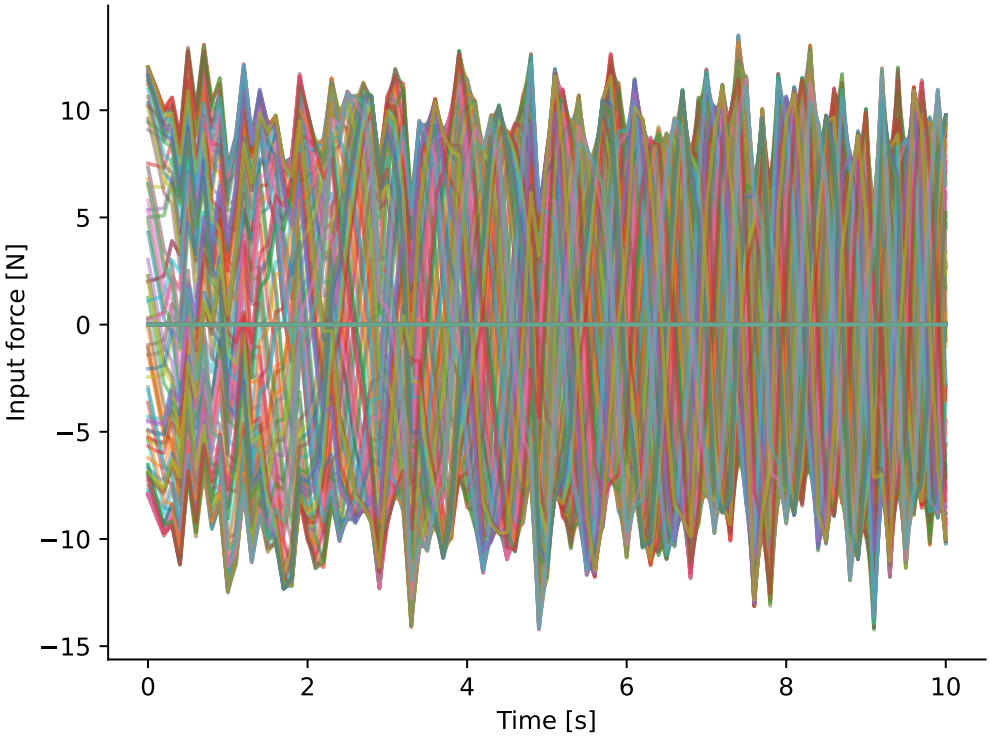

Figure 4.3: NN training dataset input force visualizations for spring mass damper systems in all the scenarios. Colors represent different trajectories.

influence on state propagation. The other 5 trajectories verified the overall correctness of the obtained models of the systems.

It is important to note that for these experiments, the measurements were not corrupted by noise to enable better validation of models. Moreover, we used different base frequencies ω_1, ω_0 of the chirp signal than in the training datasets.

4.2.3 Testing datasets

In order to obtain unbiased estimates of the performance of the identifies systems, for each scenario we developed a testing dataset. These testing datasets were generated in the same way as the validation datasets 4.2.2, with one modification - we used different base frequencies ω_1, ω_0 of the chirp signal than in the training and validation datasets.

4.3 System identification

To verify the effectiveness of various system identification approaches, we decided to select methods based on both transfer function (Section 2.1) and state space representation (Section 2.2), in the latter selecting algorithms originating from systems theory as well as

machine learning-based approaches. In the end the tested approaches include TVERA, TVOKID, LTVModels, COSMIC, nonlinear autoregressive model with exogenous input (NLARX), and Physics-informed Neural Networks (PINNs).

4.3.1 TVERA

For system identification with TVERA we used a special training dataset, prepared as described in Section 4.2.1. This method is characterized by two hyperparameters: p and q. In order to select them properly, a grid search was run for each spring mass damper scenario. In this process, the models of the systems were estimated using a corresponding training dataset for various values of the hyperparameters and then evaluated in the task of predicting validation trajectories from initial conditions with known inputs. Finally, the best models selected based on validation trajectories were tested on the testing dataset.

4.3.2 TVOKID

For system identification with TVOKID we used the same training datasets as in the TVERA experiments. This method is characterized by four hyperparameters: p, q, number of parameters, and observer order. In order to select them properly we followed the grid search approach described in Section 4.3.1 was followed. Finally we validated the resulting models using the testing dataset.

4.3.3 LTVModels

In this series of experiments, we used the single trajectory datasets, according to the definition of this method. Moreover, we used the sub-method C, i.e., *Piecewise constant time evolution* as we found it the most applicable to our cases. This system identification algorithm is characterized by a hyperparameter λ. Similarly to previous experiments, a grid search was run to find the optimal values of λ for each scenario, and the final models were validated using the testing dataset.

4.3.4 COSMIC

We split this series of experiments into two parts — system identification based on single trajectories, and based on multiple trajectories. Therefore we used both the COSMIC training datasets and their reduced versions as in Section 4.3.3. The investigated algorithm is characterized by a hyperparameter λ, therefore once again, a grid search was run for each scenario and series of experiments to determine the optimal values of λ. The final models were validated with respect to the testing trajectories.

4.3.5 NLARX

As stated in Section 2.1.4, given past states and inputs to the system, the future states can be estimated by applying a suitable nonlinear function f to these quantities. Therefore,

inspired by [31], we decided to take advantage of NLARX by modelling the function f using different deep learning regression approaches, i.e., multi-layer perceptron (MLP), residual network (ResNet), long-short term memory network (LSTM), recurrent neural network (RNN) as well as some classical machine-learning regression methods, i.e., random forest and XGBoost. The training datasets used in these experiments, were the NN training datasets, as described in Section 4.2.1.

NLARX-MLP

In these experiments, the input to the neural network were the past states and inputs to the system. The number of these past quantities considered was named the order of the network.

The MLP architecture that we used consisted of an input layer, which took as an input the previous states and inputs to the system. In some experiments, this input layer included an extra neuron representing the time at which the prediction was performed, making its presence a hyperparameter of this architecture. A leaky ReLU activation was applied to the output of this layer. The input layer was followed by a few hidden fully connected layers with a certain number of neurons. The numbers of hidden layers and neurons were another hyperparameters of this MLP. These layers also included a leaky ReLU activation followed by dropout with probability left as another hyperparameter of the architecture. Finally, the last part of the model was an output layer with two units and linear activation.

For the loss function of NLARX-MLP we chose the Mean Squared Error (MSE) of the predicted and ground truth values of the future states. To minimize this loss, the Adam optimizer was used, with learning rate and weight decay left as hyperparameters of the experiments. The training data was fed in batches, where the batch size was yet another tunable hyperparameter. The models were trained for 4000 epochs with early stopping with patience of 500 epochs. Every ten training epochs the model was evaluated on the validation trajectories in the task of trajectory prediction from initial conditions with known inputs. This way, for each training, the best model checkpoint was selected. The process of hyperparameter selection was performed in Optuna using Bayesian optimization. The best models for each spring mass damper scenario were then evaluated using the testing datasets.

NLARX-ResNet

In these experiments, the input to the neural network were the past states and inputs to the system. The number of these past quantities considered was named the order of the network, as in the case of NLARX-MLP. Similarly the inputs could include time, and the presence of this input was a tunable hyperparameter.

The architecture of NLARX-ResNet was an adaptation of the 2D ResNet architecture used widely in computer vision to 1D inputs. Each state and input recorded for the

dynamical system was treated as a separate channel in the data. The data grouped into channels was fed into an input convolutional block composed of a 1D convolutional layer, a 1D batch normalization layer, and finally a 1D max pooling layer. This convolutional layer was decided to have a fixed kernel size of 5, stride of 1 and padding *same*. The number of output channels of this layer was left as a hyperparameter of this neural network and is further referred to as the *first number of output channels*. The kernel of the max pooling layer had a fixed size of 3, stride 1 and padding 1. The input block was then followed by a few residual modules, each composed of residual blocks. The numbers of these residual modules and residual blocks building each module were left as hyperparameters of the architecture. Every residual block was built of the following sequence of layers: 1D convolution, 1D batch normalization, ReLU, 1D convolution, and 1D batch normalization. The output of the last batch normalization layer was added to either pure input of the residual block or to the input of the residual block transformed by a 1D convolutional layer with kernel size 1 (skip connection). Finally, a ReLU activation was applied, and the output of a residual block was obtained. This convolution with kernel size 1 was applied only for the second and following residual modules and only in the first residual block building a given residual module. The convolutional layers building a given residual block had the same number of output channels, fixed kernel size of 3, stride of 1, and padding *valid*. The first residual module had the same number of output channels as the first convolutional block, and each further residual module had double the number of output channels of the previous module. Finally, the output of the network was obtained by applying a 1D average pooling layer and a linear layer with two units.

The loss, optimizer, training procedure, hyperparameter optimization, and validation were the same as in the case of NLARX-MLP.

NLARX-RNN

In these experiments, the input to the neural network were the past states and inputs to the system. The number of these past quantities considered was named the order of the network, as in the case of the other NLARX experiments with neural networks. Similarly the inputs could include time, and the presence of this input was a tunable hyperparameter.

The architecture that we used was composed of a few sequentially stacked recurrent layers that process the input to the network. This series of recurrent layers treated each state and input to the dynamical system as a separate feature. Every recurrent layer was composed of a certain number of features in a hidden state. The number of recurrent layers and features in hidden states were left as the hyperparameters of the considered architecture. The activations of these recurrent layers were fixed to Tanh. Moreover, we applied dropout in the recurrent layers, with dropout strength left as another hyperparameter to be tuned. After the series of recurrent layers, the architecture was composed of a series of fully connected layers. The number of these layers was left as a

hyperparameter of the architecture. Each fully connected layer except for the last one had the same number of neurons as the number of features produced by the recurrent layers and a GELU activation. The last layer had two neurons, corresponding to the number of states of the dynamical system, and a linear activation. This series of fully connected layers was applied only to the last hidden state produced by the recurrent layers.

The loss, optimizer, training procedure, hyperparameter optimization, and validation were the same as in the case of NLARX-MLP and NLARX-ResNet.

NLARX-LSTM

In these experiments, the input to the neural network were the past states and inputs to the system. The number of these past quantities considered was named the order of the network, as in the case of the other NLARX experiments with neural networks. Similarly the inputs could include time, and the presence of this input was a tunable hyperparameter.

The architecture that we used was composed of a few sequentially stacked LSTM layers that processed the input to the network. This series of LSTM layers treated each state and input to the dynamical system as a separate feature. Every LSTM layer was composed of a certain number of features in a hidden state. The number of LSTM layers and features in hidden states were left as the hyperparameters of the considered architecture. Moreover, we applied dropout in the LSTM layers, with dropout strength left as another hyperparameter to be tuned. After the series of recurrent layers, the architecture was composed of a series of fully connected layers. The number of these layers was left as a hyperparameter of the architecture. Each fully connected layer except for the last one had the same number of neurons as the number of features produced by the LSTM layers and a GELU activation. The last layer had two neurons and a linear activation. This series of fully connected layers was applied only to the last hidden state produced by the LSTM layers.

The loss, optimizer, training procedure, hyperparameter optimization, and validation were the same as in the case of NLARX-MLP, NLARX-ResNet, and NLARX-RNN.

NLARX random forest

In these experiments, we used a fixed number of ten past states and inputs to the system, where the inputs were augmented with time. These past values were the input to the random forest regressor. The hyperparameter *number of estimators* was set to 10, and *criterion* to *squared error*, with no specification of maximal depth. The obtained state predictor was then validated using the testing trajectories.

NLARX XGBoost

In these experiments, we used a fixed number of ten past states and inputs to the system, where the inputs were augmented with time. These values were an input to the XGBoost

regressor. After training, the obtained state predictor was validated using the testing trajectories.

4.3.6 PINNs

The training datasets used in these experiments were, once again, the NN datasets. For each spring mass damper scenario, maximum and minimum values of each state x_1, x_2, and input u were obtained based on the training dataset. We increased the maximum values by 20% and decreased the minimum ones by 20%, obtaining their lower and upper bounds: $x_{1,min}$, $x_{1,max}$, $x_{2,min}$, $x_{2,max}$, u_{min}, u_{max}. Based on these values we sampled 100000 points for the evaluation of the ODE of the system as defined in (2.8) from the uniform distribution $U[(x_{1,min}, x_{1,max}), (x_{2,min}, x_{2,max}), (u_{min}, u_{max}), (0, T), (0, \Delta t)]$, where T is the total duration of the simulation and Δt is the sampling interval. Further we sampled 1000 initial condition points as in (2.8) from $U[(x_{1,min}, x_{1,max}), (x_{2,min}, x_{2,max}), (u_{min}, u_{max}), (0, T)]$ and concatenated $\Delta t = 0$.

The architecture used in these experiments was the modified MLP as described in [32], with 5 inputs and 4 hidden layers, each with 64 neurons and Tanh activation. The five inputs corresponded to the current values of $x1$, $x2$, u, t, and a time delay Δt for which the next state prediction was performed. In such setup, our PINN differed from its classical formulation in the sense that the initial conditions of the problem, i.e., $x1$, $x2$, and t were not fixed, but used as an input to the network.

Given the fact that the ODE parameters characterizing the spring mass damper systems were unknown, we defined a second network for each scenario to find their values. In all the cases, this network was composed of 4 hidden layers with Tanh activations and its input was the sum $t + \Delta t$. For the **Linear Time Varying (LTV)** scenario the layers of this network had 3 neurons, to predict $m(t)$, $C_s(t)$, and $C_d(t)$. For the **nonlinear (NL)** scenario these layers had 4 neurons, to predict $m(t)$, $C_s(t)$, $C_d(t)$, and $C_{d3}(t)$. For the **nonlinear perturbed (NLD)** scenario we had 5 neurons, to predict $m(t)$, $C_s(t)$, $C_d(t)$, $C_{d3}(t)$, and $d(x_1(t))$. Finally for the **LTV instantaneous reconfiguration** and **LTV reconfiguration** scenarios we had 4 neurons, to predict $m(t)$, $C_s(t)$, $C_d(t)$, and $d_v(t)$.

To train these neural networks, we followed the instructions published in a paper by Wang et al. [32]. For the first 5000 iterations, we only minimized the data fidelity term by predicting the next state of the system. Then for the next 8000 steps we kept increasing the regularization weight η linearly from 0 to $\frac{8}{7}$. Initially, we were training the networks using the Adam optimizer setup with a learning rate $5e-3$. After the first 10000 training steps, the optimizer was changed to LBFGS. Every 5000 steps, the 100000 points of evaluation of the ODE were resampled, and so were the data points (in batches of 8192). For each scenario we adjusted the regularization part of the loss according to the ODEs defined in Chapter 3. Finally we validated the resulting models on the testing dataset.

4.4 Model validation

For the purpose of validation, the obtained models were evaluated in two tasks: trajectory prediction from initial conditions with known inputs, and trajectory prediction one step ahead. A schematic of the model validation procedure is presented in Figure 4.4. In general, the trajectory prediction loss (4.2) computed for the validation trajectories was used as a metric to be minimized in hyperparameter tuning. All the validation methods described in Sections 4.4.1 - 4.4.3, applied to the testing trajectories, were used for the final validation of models.

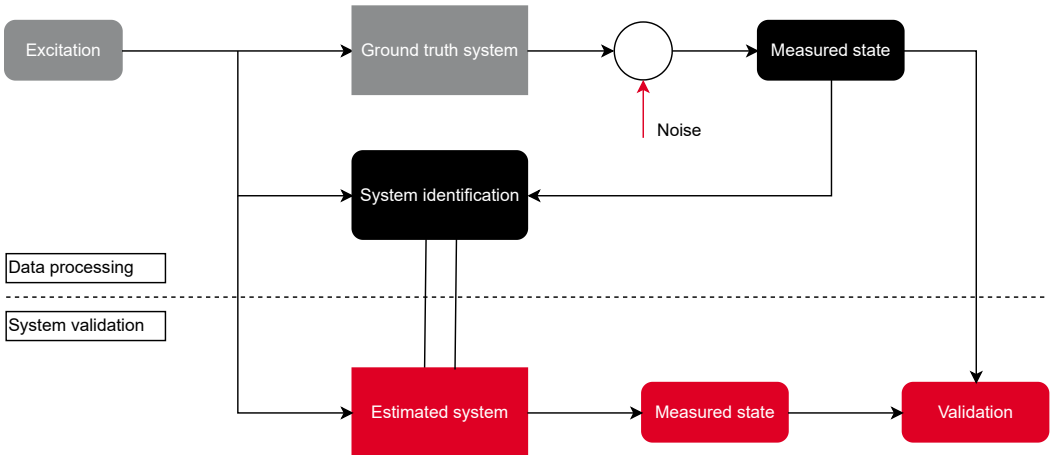

Figure 4.4: Schematic of the experimental setup for model validation.

4.4.1 Trajectory prediction loss

Based on the predictions from initial conditions, root mean squared error (RMSE) was computed on either the set of validation or testing trajectories. The RMSEs obtained in this manner were averaged to obtain trajectory prediction loss:

$$L(\boldsymbol{\beta}) = \frac{1}{|\mathbf{S}|} \sum_{l \in \mathbf{S}} \sqrt{\frac{1}{N} \sum_{k=1}^{N} \sum_{m=1}^{p} \left(\hat{x}_{k,m}^{(l)}(\boldsymbol{\beta}) - x_{k,m}^{(l)} \right)^2}, \tag{4.2}$$

where $\boldsymbol{\beta}$ corresponds to hyperparameters of a system identification method, $\hat{x}_{k,m}^{(l)}(\boldsymbol{\beta})$ is the m-th element of the p-dimensional state vector predicted from initial conditions by the identified system at time step k for trajectory l, $x_{k,m}^{(l)}$ is a corresponding true state of the system, and $\mathbf{S}$ represents a set of trajectories. Moreover, a standard deviation of these RMSEs was computed.

4.4.2 ECDFs of scaled absolute position residuals

The full horizon predictions were also used to produce empirical cumulative distribution function (ECDF) plots of scaled absolute position residuals. For these plots, for each

testing trajectory, absolute values of position residuals were computed and scaled by a mean absolute value of position of this true trajectory.

4.4.3 Analysis of residual correlations

Based on the predictions one step ahead, we evaluated the autocorrelation of residuals and their cross correlation with inputs to the system. We computed the standard deviations, means, as well as maximum and minimum absolute values of these correlations. They were also plotted as ECDFs.

5

Results

This chapter presents the results of system identification experiments described in Chapter 4. We start with providing the optimal hyperparameter values for each method, if applicable. In the discussion of NLARX hyperparameters, we also select the best NLARX approach for each spring mass damper scenario. We then present a comparison of test results for all the experiments from Chapter 4, where the NLARX models are represented by the best approach for a given system scenario.

5.1 TVERA hyperparameters

In this series of experiments, the hyperparameter values were selected by running a grid search and computing the validation trajectory prediction losses, as described in Section 4.3.1. The optimal hyperparameters are presented in Table 5.1. Note that we obtained a stable value of $q = 50$, which was also the maximum value of this parameter investigated in the grid search. The magnitude of this hyperparameter indicates that the Hankel matrices used in the computations were large in terms of number of columns, which can be attributed to a long impulse response of the simulated systems.

Table 5.1: Results of hyperparameter tuning for TVERA models.

Hyperparameter	**LTV**	**NL**	**NLD**	**LTV inst reconfig**	**LTV reconfig**
p	2	5	50	2	5
q	50	50	50	50	50

5.2 TVOKID hyperparameters

Similarly to Section 5.1, we run a grid search to select the best TVOKID hyperparameters based on the validation trajectory prediction loss. The results of this process are presented in Table 5.2. The introduction of an asymptotically stable observer by this algorithm resulted in smaller values of q, compared to the results presented in Section 5.1. Thus, the problems related to long impulse response were eased.

Table 5.2: Results of hyperparameter tuning for TVOKID models.

Hyperparameter	LTV	NL	NLD	LTV inst reconfig	LTV reconfig
p	2	2	17	2	2
q	44	20	17	42	31
Number of parameters	44	81	83	66	44
Observer order	3	43	2	5	7

5.3 LTVModels hyperparameters

As in the previous sections, the optimal values of hyperparameter λ for this method were obtained by an application of a grid search and a computation of validation trajectory prediction losses. The results of this process are presented in Table 5.3. These values did not reach the minimal nor maximal considered magnitudes. As one can see, high system variability was favored in the LTV, NLD, and LTV instantaneous reconfiguration scenarios, contrary to NL and LTV reconfiguration systems, where low system variability provided better results.

Table 5.3: Results of hyperparameter tuning for LTVModels.

Hyperparameter	LTV	NL	NLD	LTV inst reconfig	LTV reconfig
λ	1.0	50.0	1.0	1.0	50.0

5.4 COSMIC hyperparameters

As previously, for both the single and multi-trajectory versions of COSMIC, a grid search was run to find the optimal values of λ based on the validation trajectory prediction losses. The results of this process are presented in Table 5.4. As one can see, in the single trajectory case, higher system variability was favored in the LTV and NLD scenarios, whereas in the rest of the investigated systems, low system variability resulted in better state propagation. As for the multi-trajectory experiments, high system variability was preferred in the LTV, LTV instantaneous reconfiguration, and LTV reconfiguration scenarios, contrary to NL and NLD. This shows that for LTV system identification it is important to consider multiple trajectories in order to avoid drawing wrong conclusions.

Table 5.4: Results of hyperparameter tuning for COSMIC models.

Hyperparameter	LTV	NL	NLD	LTV inst reconfig	LTV reconfig
λ (single trajectory)	1	1e10	1	1e10	1e8
λ (multi-trajectory)	1e-1	1e3	1e1	1e-4	1e-5

5.5 NLARX hyperparameters and model selection

5.5.1 Hyperparameters

In this series of experiments, we investigated six different approaches to modelling the f function in the NLARX model, i.e., MLP, ResNet, RNN, LSTM, XGBoost, and random forest. The first four listed methods included a process of Bayesian hyperparameter optimization in Optuna. The results of this process are presented in Tables 5.5 - 5.8.

Table 5.5: Results of hyperparameter tuning in Optuna for NLARX-MLP models.

Hyperparameter	**LTV**	**NL**	**NLD**	**LTV inst reconfig**	**LTV reconfig**
Model order	5	11	11	8	6
Include time	True	True	False	True	False
Number of hidden layers	24	39	34	11	19
Number of neurons in hidden layers	45	25	32	47	34
Dropout strength	0.0034	0.2618	0.1783	0.3033	0.0693
Learning rate	2.78e-5	9.61e-4	4.66e-4	5.92e-5	8.36e-4
Weight decay	0.0013	0.009	0.0613	0.234	0.302
Batch size	512	128	32	512	256

Table 5.6: Results of hyperparameter tuning in Optuna for NLARX-ResNet models.

Hyperparameter	**LTV**	**NL**	**NLD**	**LTV inst reconfig**	**LTV reconfig**
Model order	11	11	8	7	10
Include time	False	True	False	True	True
Number of residual modules	2	5	2	4	4
Number of residual blocks in each residual module	3	2	2	3	2
Number of output channels of the first convolutional block	32	16	32	32	16
Learning rate	1.42e-5	2.37e-4	4.27e-5	4.15e-5	1.17e-4
Weight decay	0.0003	0.0127	0.061	0.0344	0.2189
Batch size	256	1024	512	256	512

5.5.2 Model selection

After selecting the optimal hyperparameters, the final models for each approach were then validated on the testing dataset following the methods described in Section 4.4.

Trajectory prediction from initial conditions

Each model was evaluated in the task of trajectory prediction from initial conditions. Based on these predictions, we computed the trajectory prediction loss, i.e., the mean value of RMSE. Moreover, we calculated the standard deviations of RMSEs across different

Table 5.7: Results of hyperparameter tuning in Optuna for NLARX-RNN models.

Hyperparameter	**LTV**	**NL**	**NLD**	**LTV inst reconfig**	**LTV reconfig**
Model order	11	7	10	1	11
Include time	False	True	False	True	True
Number of recurrent layers	2	4	4	4	4
Number of features in hidden states of the recurrent layers	128	32	16	128	64
Number of linear layers	1	2	2	4	2
Dropout strength	0.418	0.0503	0.4501	0.460	0.0814
Learning rate	1.29e-4	1.23e-5	2.07e-2	0.1254	0.0049
Weight decay	0.0127	5.81e-4	0.00151	0.4455	0.1551
Batch size	256	1024	1024	256	1024

Table 5.8: Results of hyperparameter tuning in Optuna for NLARX-LSTM models.

Hyperparameter	**LTV**	**NL**	**NLD**	**LTV inst reconfig**	**LTV reconfig**
Model order	11	5	4	11	6
Include time	False	True	False	False	True
Number of LSTM layers	2	2	2	2	2
Number of features in hidden states of the LSTM layers	32	16	8	128	128
Number of linear layers	1	1	1	2	1
Dropout strength	0.285	0.280	0.418	0.300	0.282
Learning rate	2.09e-2	1.93e-3	2.10e-3	1.20e-3	2.075e-5
Weight decay	2.67e-3	7.65e-4	0.022	0.111	0.161
Batch size	1024	256	1024	1024	1024

trajectories. These metrics are presented in Table 5.9. The absolute values of position residuals were scaled by the mean absolute values of position for each trajectory as described in Section 4.4.2 and their ECDFs were plotted (Figure 5.1). As one can see, in all the scenarios the best models performed predictions that resulted in around 90% or more of scaled absolute residuals with magnitudes lower than 1.0. Based on the values of trajectory prediction loss and distributions of residuals we can initially select the best NLARX approaches, however, first it is important to evaluate the correlations of the residuals in the task of next state prediction.

Trajectory prediction one step ahead

We then continued with an analysis of residuals obtained in the task of testing trajectory prediction one step ahead. Their autocorrelations and cross correlations with system inputs were computed. Tables 5.11, 5.12 present the statistics of these correlations. Moreover, they were plotted as ECDFs (Figures 5.2, 5.3). As one can see, the values of autocorrelation of residuals were reported to be greater than 0.2 in approximately 20% of the samples,

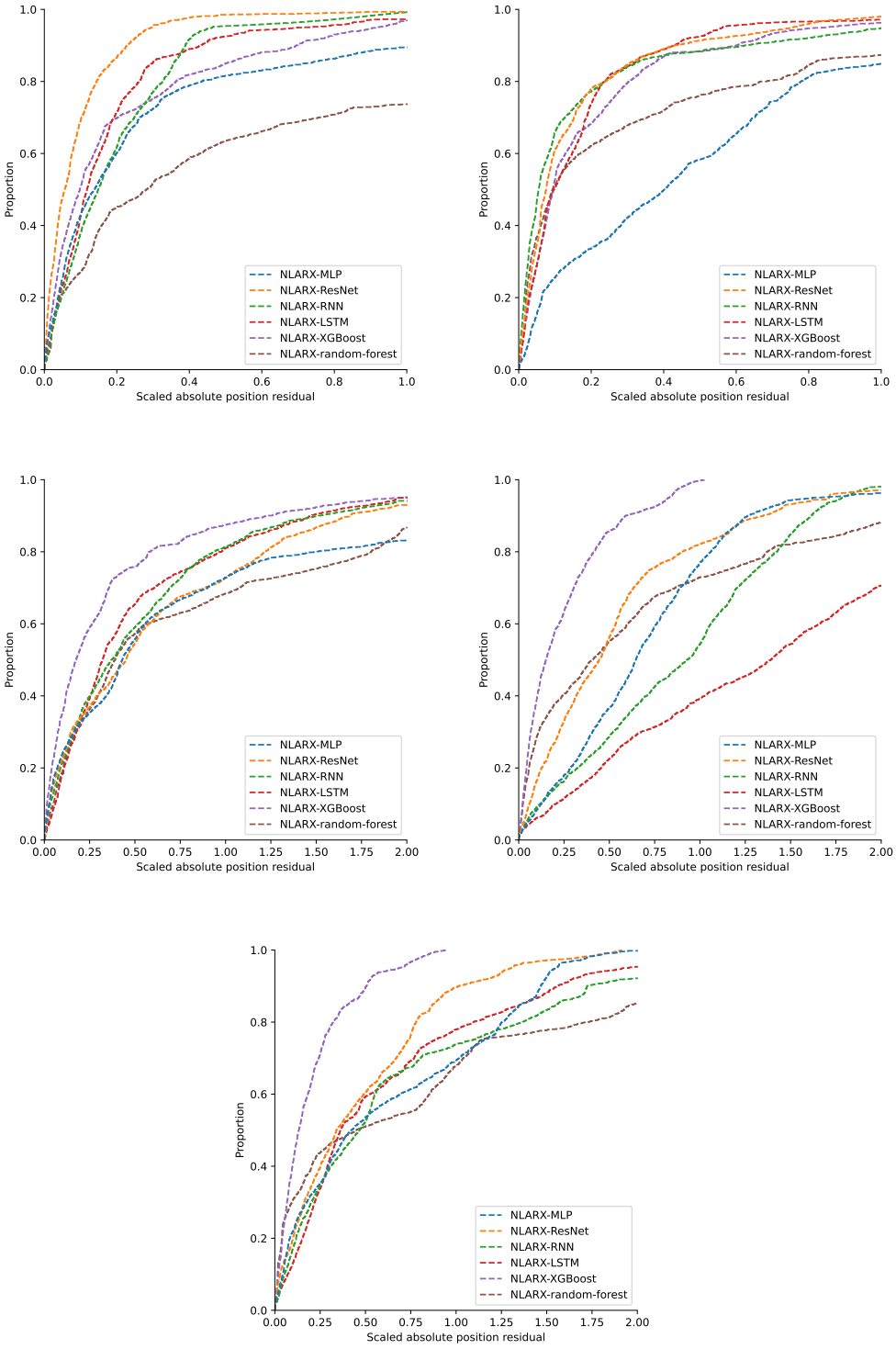

Figure 5.1: ECDFs of scaled absolute position residuals for NLARX models. Left to right and top to bottom: LTV, NL, NLD, LTV instantaneous reconfiguration, LTV reconfiguration. In each case we can find a model that produced around 90% of scaled absolute position residuals smaller than 1.0.

Table 5.9: Comparison of trajectory prediction loss in the task of testing trajectory prediction from initial conditions for NLARX models. Best results for each system are marked by bold font.

System	RMSE	NLARX-MLP	NLARX-ResNet	NLARX-RNN	NLARX-LSTM	NLARX-XGBoost	NLARX-random-forest
LTV	mean	0.67	**0.26**	0.53	0.49	0.54	1.55
	std	0.42	**0.19**	0.35	0.27	0.27	0.93
NL	mean	0.81	0.27	**0.23**	0.29	0.51	0.79
	std	0.42	0.22	**0.15**	0.14	0.43	0.31
NLD	mean	1.73	1.56	1.76	1.47	**1.40**	1.73
	std	1.26	**0.96**	1.28	1.16	0.99	1.33
LTV inst reconfig	mean	2.28	**1.45**	2.84	4.27	2.22	3.45
	std	0.88	**0.53**	1.10	1.39	1.50	2.26
LTV reconfig	mean	1.48	1.11	1.59	1.41	**0.81**	1.85
	std	0.94	0.62	0.54	0.50	**0.39**	1.44

Table 5.10: Selection of the best NLARX models for each scenario.

System scenario	LTV	NL	NLD	LTV reconfig	LTV inst reconfig
Selected NLARX model	NLARX-ResNet	NLARX-RNN	NLARX-XGBoost	NLARX-ResNet	NLARX-XGBoost

for all models and system scenarios. In the case of cross correlation of residuals and system inputs, we observe values greater than 0.2 for around 15% or less samples in all the considered cases. Therefore, we can draw a conclusion that the correlations of residuals in general were not significant and the models of the system were valid, as they explained most of the testing dataset.

Summary

Given that none of the discussed models were invalidated in the analysis of correlations of residuals in the task of next state prediction, we can select the best NLARX approaches based on Table 5.9 and Figure 5.1. We present this selection in Table 5.10.

Table 5.11: Statistical summary of autocorrelations of residuals for NLARX models in the next state prediction task.

System	Auto-correlation	NLARX-MLP	NLARX-ResNet	NLARX-RNN	NLARX-LSTM	NLARX-XGBoost	NLARX-random-forest
LTV	mean	0.11	0.14	0.16	0.15	0.08	0.12
	std	0.19	0.21	0.24	0.23	0.13	0.19
	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.95	0.97	0.98	0.98	0.83	0.97
NL	mean	0.17	0.16	0.16	0.18	0.10	0.11
	std	0.26	0.23	0.25	0.26	0.16	0.18
	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.97	0.98	0.99	0.98	0.92	0.96
NLD	mean	0.19	0.14	0.12	0.18	0.10	0.12
	std	0.27	0.22	0.20	0.26	0.16	0.19
	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.98	0.98	0.98	0.98	0.97	0.97
LTV	mean	0.22	0.17	0.21	0.17	0.11	0.12
inst	std	0.30	0.24	0.30	0.25	0.19	0.20
reconfig	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.98	0.97	0.99	0.97	0.91	0.96
LTV	mean	0.14	0.18	0.14	0.18	0.10	0.10
reconfig	std	0.22	0.26	0.22	0.26	0.17	0.18
	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.98	0.97	0.97	0.98	0.91	0.92

Table 5.12: Statistical summary of cross correlations of residuals and inputs for NLARX models in the next state prediction task.

System	Cross-correlation	NLARX-MLP	NLARX-ResNet	NLARX-RNN	NLARX-LSTM	NLARX-XGBoost	NLARX-random-forest
LTV	mean	0.07	0.10	0.09	0.09	0.07	0.08
	std	0.09	0.12	0.14	0.14	0.14	0.13
	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.58	0.37	0.63	0.79	1.00	0.90
NL	mean	0.08	0.08	0.07	0.07	0.07	0.07
	std	0.15	0.12	0.11	0.13	0.15	0.14
	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.71	0.55	0.64	0.75	1.00	0.93
NLD	mean	0.08	0.06	0.06	0.07	0.07	0.07
	std	0.13	0.09	0.11	0.11	0.13	0.13
	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.75	0.48	0.66	0.70	0.90	0.69
LTV	mean	0.08	0.08	0.08	0.09	0.09	0.10
inst	std	0.10	0.12	0.12	0.12	0.15	0.15
reconfig	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.38	0.44	0.61	0.50	0.67	0.59
LTV	mean	0.08	0.08	0.09	0.08	0.09	0.10
reconfig	std	0.12	0.11	0.14	0.12	0.15	0.16
	min abs	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.52	0.45	0.61	0.40	0.66	0.63

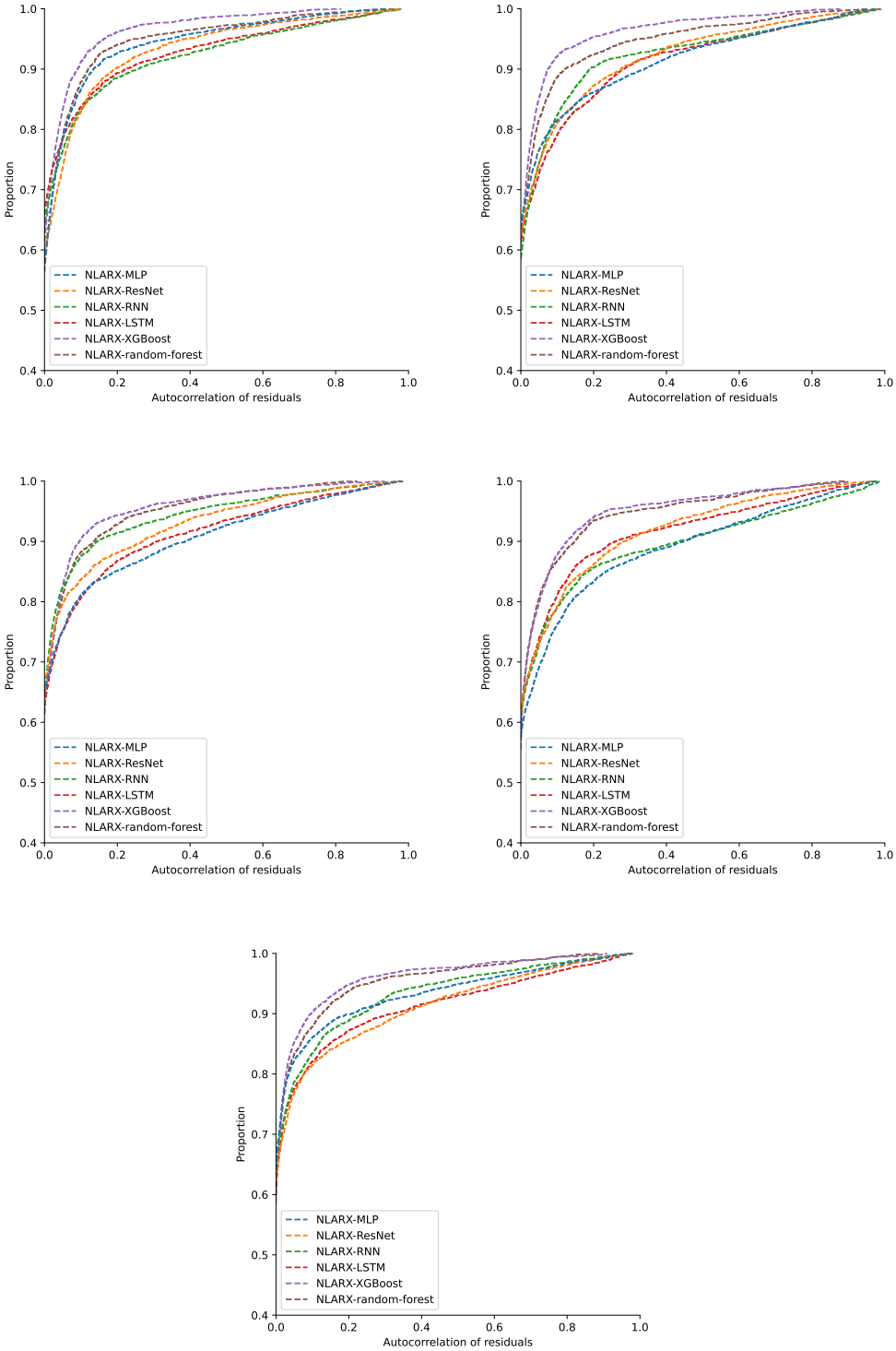

Figure 5.2: ECDFs of autocorrelations of residuals for NLARX models. Left to right and top to bottom: LTV, NL, NLD, LTV instantaneous reconfiguration, LTV reconfiguration. The observed distributions imply that these autocorrelations were not significant.

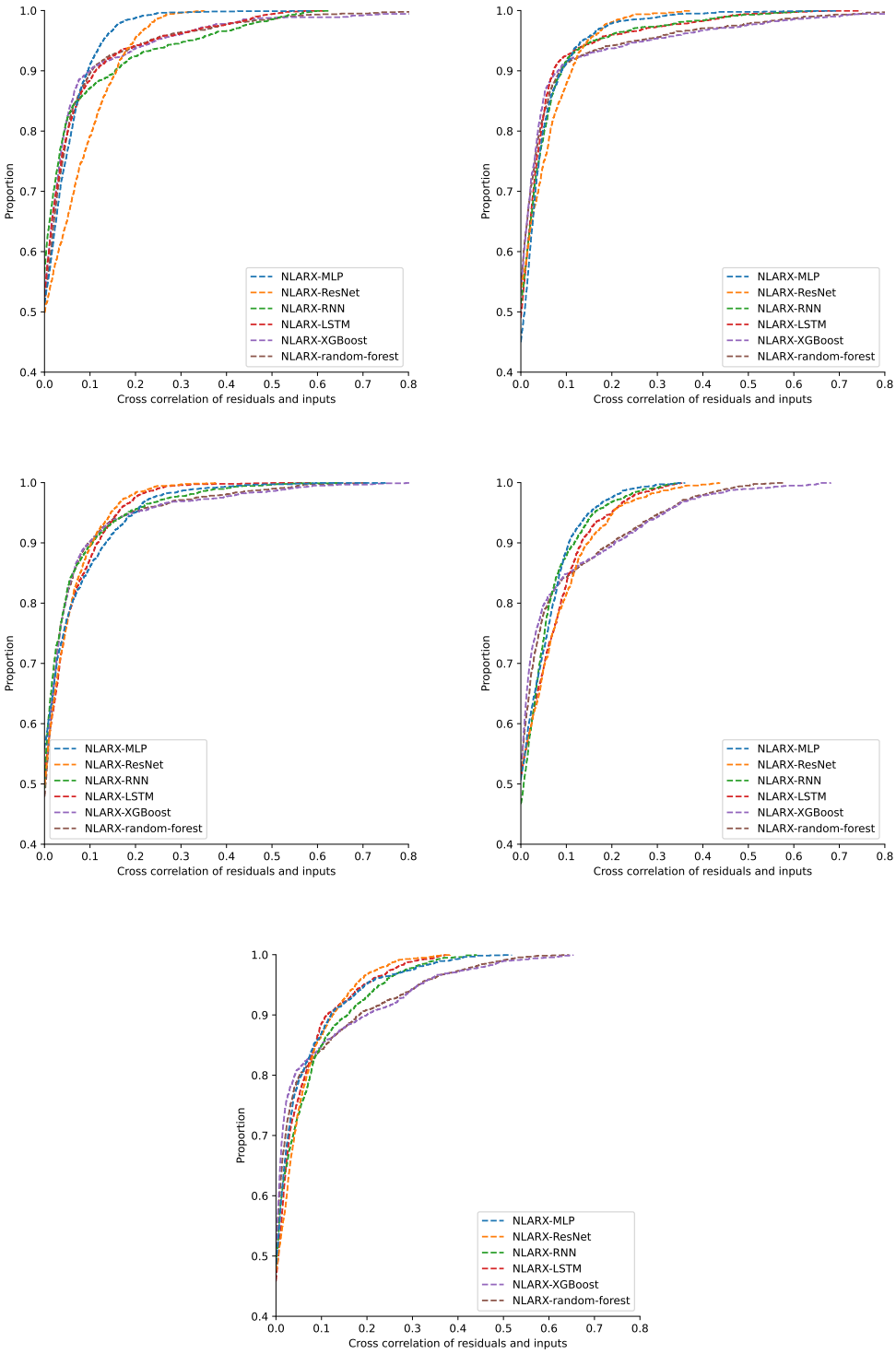

Figure 5.3: ECDFs of cross correlations of residuals and inputs for NLARX models. Left to right and top to bottom: LTV, NL, NLD, LTV instantaneous reconfiguration, LTV reconfiguration.The observed distributions imply that these cross correlations were not significant.

5.6 Comparison of test results

In this section we present a comparison of test results for all the experiments described in Chapter 4. The NLARX group of experiments is represented by the best approaches for a given system as selected in Section 5.5 of this Chapter. Each experiment name corresponds to an adequate model trained with optimal hyperparameters.

5.6.1 Trajectory prediction from initial conditions

The first part of this comparison focuses on the task of trajectory prediction from initial conditions. For the predicted trajectories, we computed the trajectory prediction loss, i.e., the mean value of RMSE, and standard deviations of these RMSEs across the trajectories. These results are presented in Table 5.13.

The absolute values of position residuals in the task of testing trajectory prediction from initial conditions were scaled by the mean absolute values of position for each trajectory as described in Section 4.4.2. We plotted the resulting quantities against the discrete time index for each of the systems (Figure 5.4). We also present their ECDFs (Figure 5.5). In these plots, we included the residuals resulting from trajectory predictions using an adequate discrete-time linearization of a system, given by formulas provided in subsequent sections of Chapter 3.

The simplest observation drawn from these figures is the fact that the residuals grow as the time index increases (Figure 5.4). This occurs due to the accumulation of errors in state propagation. In Figure 5.5, we can see that for each method in the LTV scenario, almost 100% of the scaled absolute values of position residuals did not exceed 1.0. The model identified by COSMIC, in this case, produced almost perfect predictions, obtaining performance comparable to the linearization. For more complex scenarios, such as NL or NLD, the best models, i.e., identified by NLARX, yielded around 100% to 90% of these values with a magnitude lower than 1.0. For the worst models this proportion fell to 60 - 40%. Finally for the reconfiguring scenarios, the best method, i.e., COSMIC,

Table 5.13: Comparison of trajectory prediction loss in the task of testing trajectory prediction from initial conditions for different LTV system identification methods. Best results for each system scenario are marked by bold font.

System	RMSE	TVERA	TVOKID	LTV-Models	COSMIC single trajectory	COSMIC	NLARX	PINN
LTV	mean	0.71	0.52	0.37	0.36	**0.01**	0.29	0.25
	std	0.52	0.37	0.21	0.18	**0.01**	0.42	0.15
NL	mean	1.15	1.07	0.32	0.32	0.35	**0.23**	0.38
	std	0.91	0.55	0.21	0.21	0.23	**0.15**	0.31
NLD	mean	5.95	2.61	2.41	2.41	2.20	**1.40**	1.97
	std	2.21	**0.90**	1.04	1.16	0.94	0.99	1.22
LTV inst reconfig	mean	0.61	0.47	2.14	2.50	**0.02**	1.45	1.50
	std	0.41	0.33	1.13	1.51	**0.01**	0.53	1.14
LTV reconfig	mean	0.65	0.31	1.41	1.36	**0.01**	0.81	1.31
	std	0.52	0.23	0.71	0.76	**0.00**	0.39	0.93

Table 5.14: Selection of the best LTV system identification methods for each scenario.

System scenario	LTV	NL	NLD	LTV reconfig	LTV inst reconfig
Selected method	COSMIC	NLARX	NLARX	COSMIC	COSMIC

produced almost perfect predictions (100% of scaled absolute position residuals below 0.05), whereas the worst ones resulted in only around 60% of scaled residuals lower than 1.0. Moreover, in Figures 5.4, 5.5 we can see that for each considered system scenario we could find a machine-learning-based approach which performed better than the classical system identification approaches in terms of the state propagation accuracy of a resulting model. Finally, we notice that, for all these systems except for LTV, the best approaches resulted in better predictions than their linearizations. In the LTV case, the best model, i.e., the one obtained by COSMIC had a comparable performance to the corresponding linearization.

5.6.2 Trajectory prediction one step ahead

We continued with the validation of the models in the task of next step prediction for the testing trajectories. The resulting residuals were collected and we computed their auto-correlations and cross correlations with system inputs. The statistics of these correlations were calculated (Tables 5.15, 5.16) and their ECDFs were plotted (Figures 5.6, 5.7). In each system scenario and system identification method, around 85% of the autocorrelations of residuals had values below 0.2. Regarding the cross correlations of residuals and inputs to the system, we observe the same property for around 90% of the values. These magnitudes of correlations are not significant, therefore we can draw a conclusion that the obtained models of the systems were valid, as they explained most of the testing data.

5.6.3 Summary

From this analysis we can draw conclusions about which system identification method worked best in each scenario (see Table 5.14)

For all the systems, except for LTV, these best methods resulted in models that were more accurate for state propagation than the corresponding linearizations. In the LTV case, this accuracy was comparable. Furthermore, this benchmark shows the superiority of machine-learning-based approaches over classical system identification methods, as this first class of algorithms dominated the selection of best methods. Nevertheless, as one can see, it is impossible to choose one best system identification method, because this choice depends on the considered scenario. This observation is in-line with the *No free lunch theorem* [33].

Table 5.15: Statistical summary of autocorrelations of residuals for different LTV system identification methods in the next state prediction task.

System	Autocorrelation	TVERA	TVOKID	LTV-Models	COSMIC single trajectory	COSMIC	NLARX	PINN
LTV	mean	0.01	0.02	0.12	0.12	0.05	0.14	0.07
	std	0.04	0.05	0.20	0.19	0.08	0.21	0.23
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.0
	max abs	0.43	0.50	0.96	0.98	0.42	0.97	0.99
NL	mean	0.04	0.04	0.13	0.13	0.14	0.16	0.12
	std	0.08	0.07	0.21	0.21	0.23	0.25	0.20
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.70	0.43	0.95	0.94	0.97	0.99	0.99
NLD	mean	0.14	0.09	0.05	0.06	0.04	0.10	0.07
	std	0.22	0.17	0.11	0.10	0.07	0.16	0.15
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.99	0.94	0.81	0.73	0.76	0.97	0.95
LTV inst reconfig	mean	0.01	0.02	0.06	0.07	0.05	0.17	0.05
	std	0.22	0.05	0.13	0.14	0.08	0.24	0.09
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.99	0.51	0.92	0.92	0.55	0.97	0.95
LTV reconfig	mean	0.03	0.03	0.09	0.06	0.06	0.10	0.06
	std	0.07	0.05	0.16	0.12	0.09	0.17	0.12
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.63	0.49	0.92	0.82	0.69	0.91	0.97

Table 5.16: Statistical summary of cross correlations of residuals and inputs for different LTV system identification methods in the next state prediction task.

System	Cross correlation	TVERA	TVOKID	LTV-Models	COSMIC single trajectory	COSMIC	NLARX	PINN
LTV	mean	0.07	0.11	0.08	0.08	0.05	0.10	0.07
	std	0.08	0.12	0.13	0.11	0.07	0.12	0.10
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.24	0.25	0.62	0.66	0.64	0.37	0.53
NL	mean	0.11	0.05	0.06	0.06	0.06	0.07	0.05
	std	0.13	0.06	0.12	0.12	0.12	0.11	0.09
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.39	0.25	0.87	0.86	0.83	0.64	0.66
NLD	mean	0.08	0.06	0.04	0.06	0.04	0.07	0.04
	std	0.11	0.09	0.07	0.09	0.07	0.13	0.07
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.55	0.36	0.43	0.41	0.48	0.90	0.45
LTV inst reconfig	mean	0.08	0.12	0.04	0.07	0.07	0.08	0.05
	std	0.09	0.13	0.07	0.11	0.09	0.12	0.08
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.21	0.25	0.49	0.41	0.62	0.44	0.36
LTV reconfig	mean	0.10	0.10	0.05	0.06	0.05	0.09	0.05
	std	0.12	0.11	0.09	0.11	0.09	0.15	0.08
	min abs	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	max abs	0.29	0.24	0.46	0.46	0.64	0.66	0.33

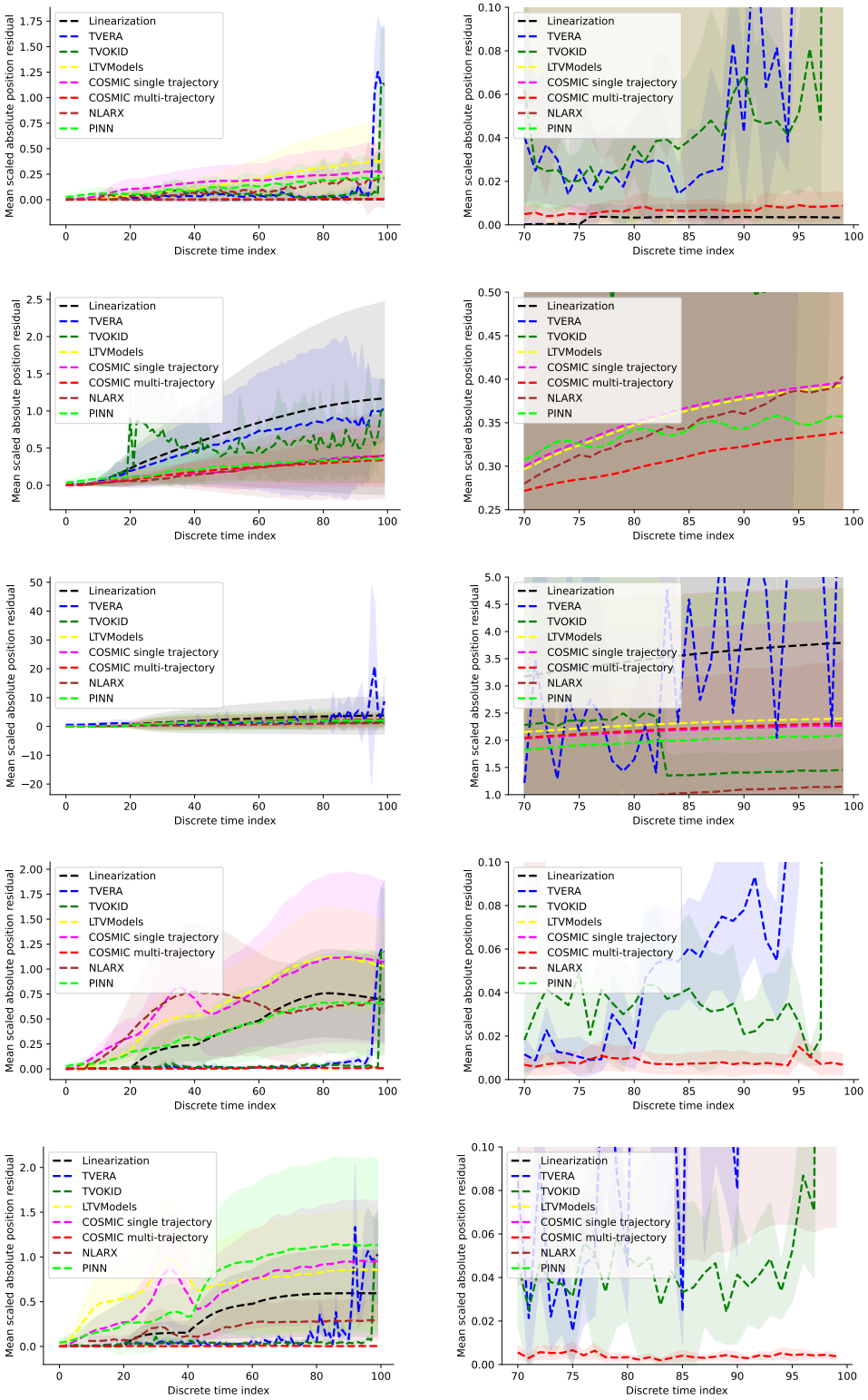

Figure 5.4: Scaled absolute position residuals averaged over testing trajectories for different LTV system identification methods. Shaded area corresponds to standard deviation of the discussed values. Left to right: full plot, zoom to the last 30 time indices. Top to bottom: LTV, NL, NLD, LTV instantaneous reconfiguration, LTV reconfiguration. In each case at least one machine-learning-based approach performs better than the classical methods with performance better than or comparable to linearization.

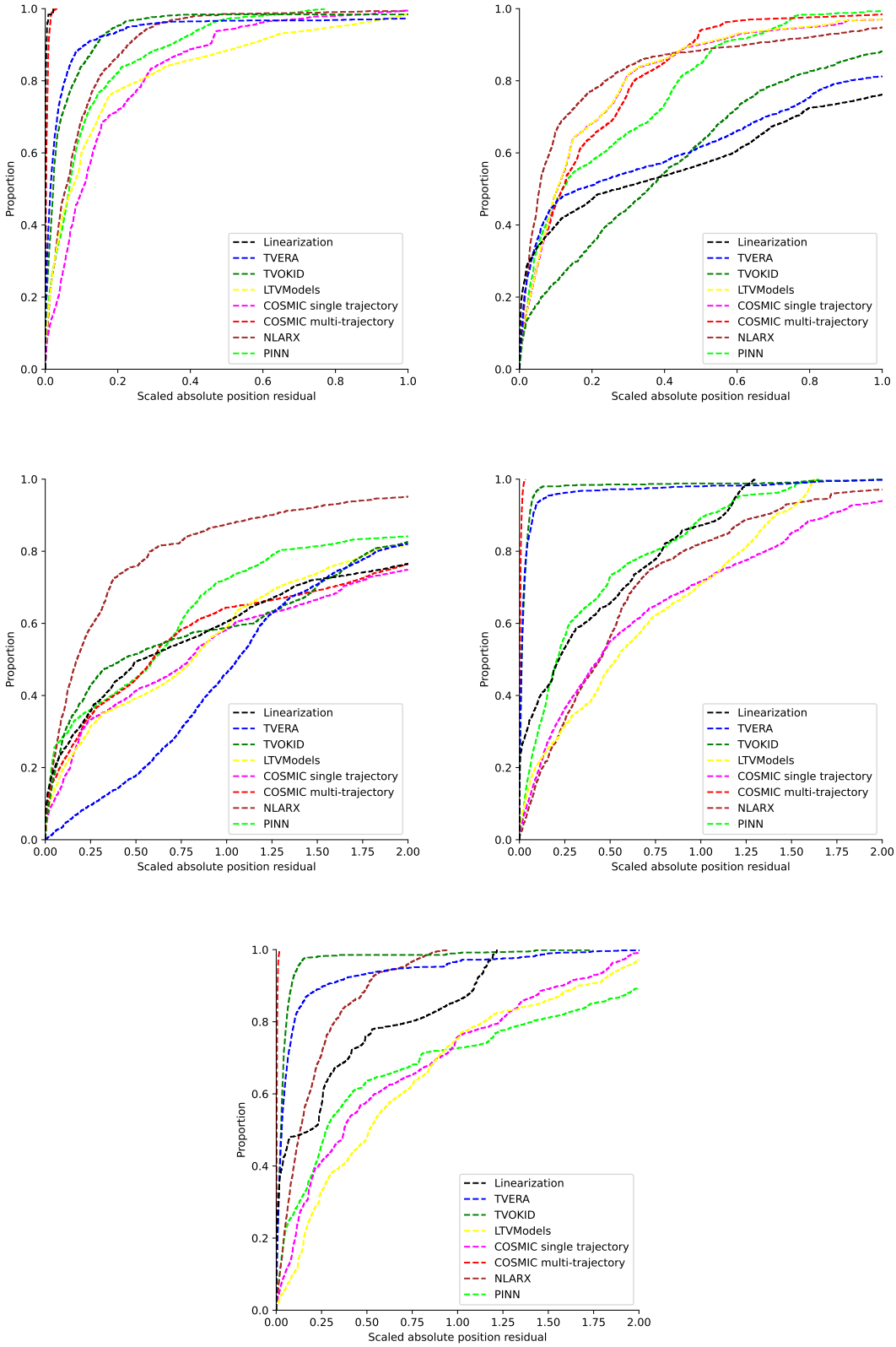

Figure 5.5: ECDFs of absolute position residuals for different LTV system identification methods. Left to right and top to bottom: LTV, NL, NLD, LTV instantaneous reconfiguration, LTV reconfiguration. In each case at least one machine-learning-based approach performs better than the classical methods with performance better than or comparable to linearization.

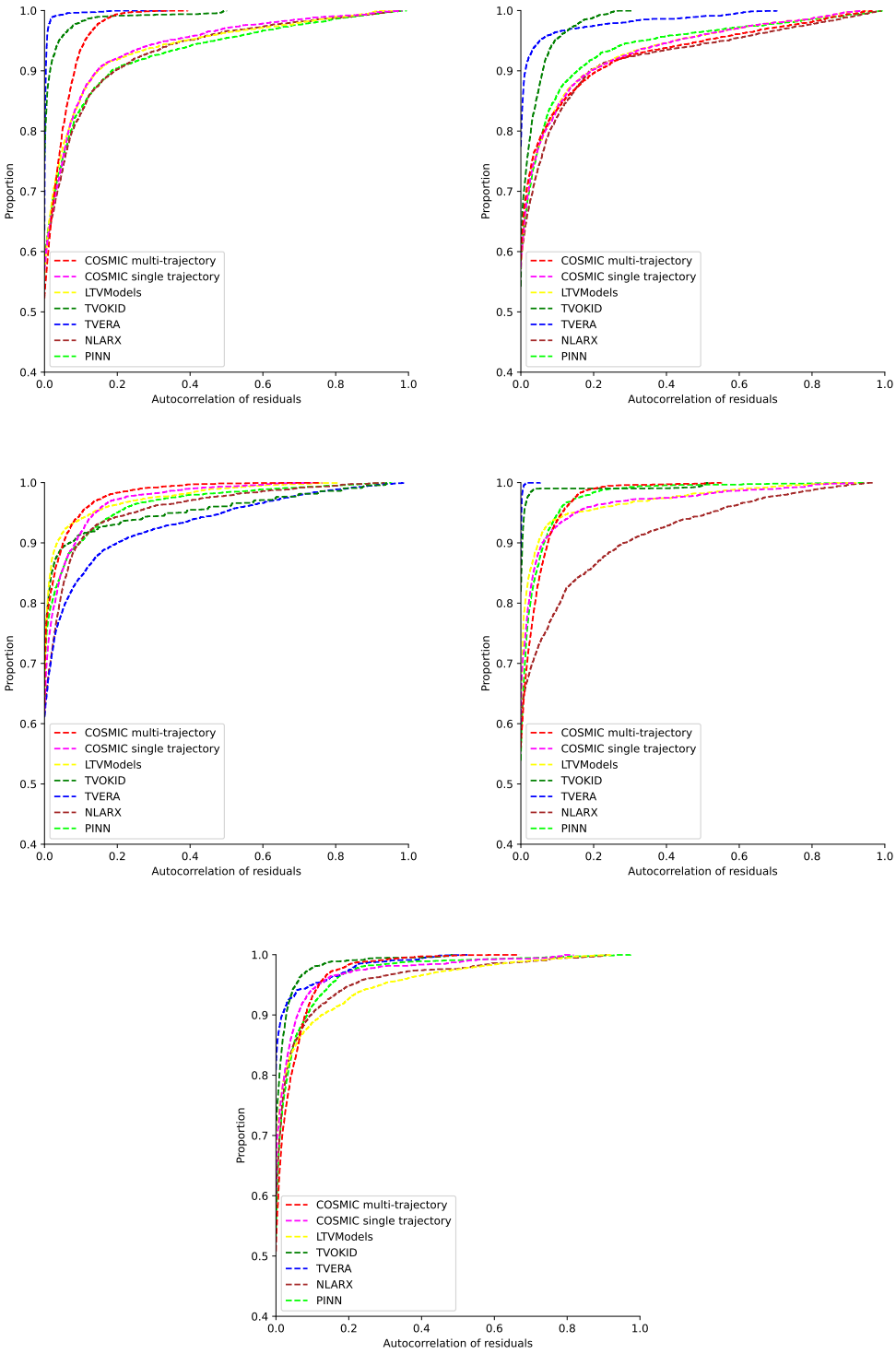

Figure 5.6: ECDFs of autocorrelations of residuals for different LTV system identification methods. Left to right and top to bottom: LTV, NL, NLD, LTV instantaneous reconfiguration, LTV reconfiguration. The observed distributions imply that these autocorrelations were not significant.

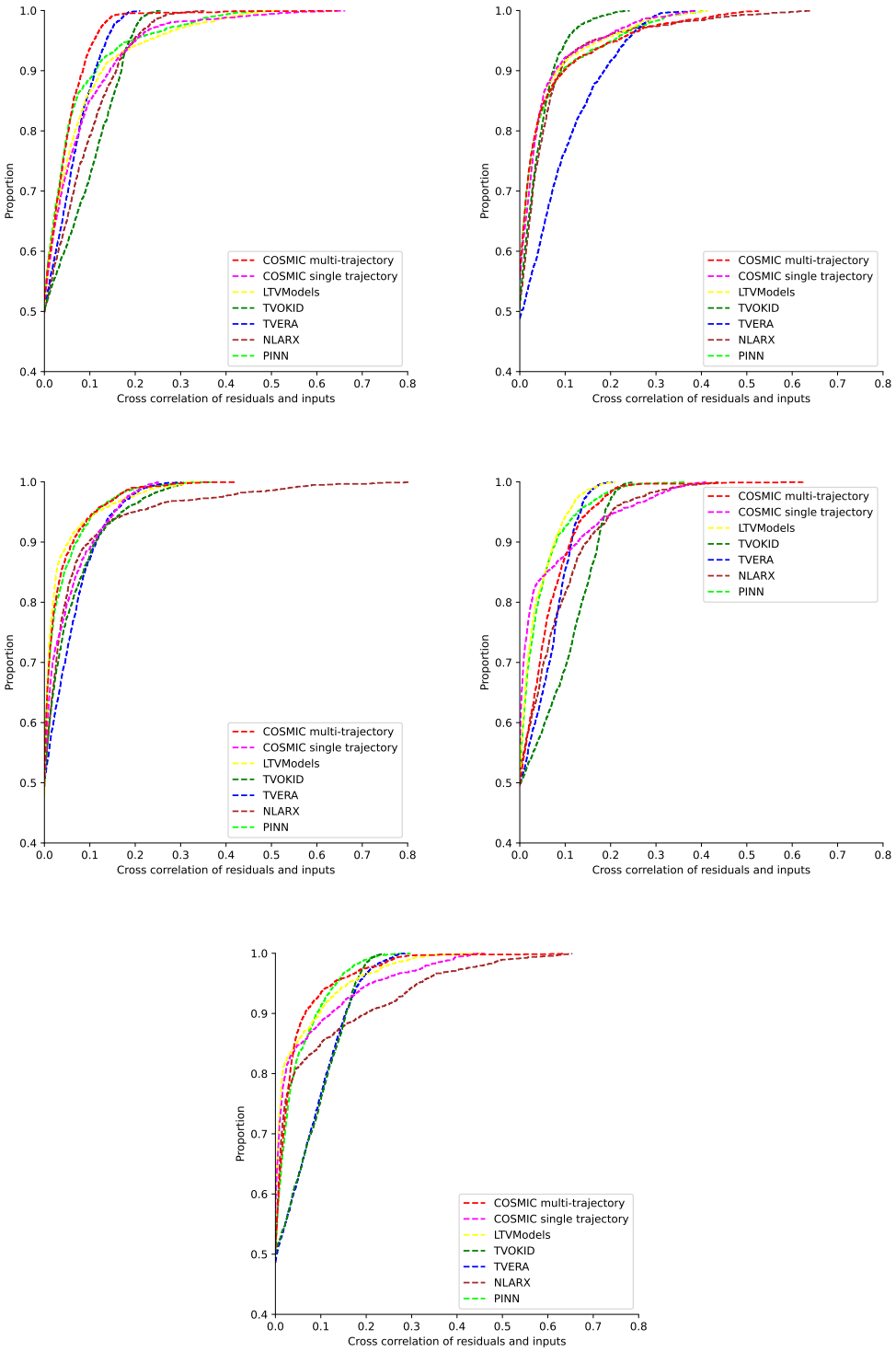

Figure 5.7: ECDFs of cross correlations of residuals and inputs for different LTV system identification methods. Left to right and top to bottom: LTV, NL, NLD, LTV instantaneous reconfiguration, LTV reconfiguration. The observed distributions imply that these cross correlations were not significant.

6

Conclusions and future work

The work developed in this thesis aims to propose and compare machine-learning-based approaches for LTV system identification with classical methods. The focus is put on system identification for reconfigurable spacecraft such as e.g., space structures during an assembly performed by a robotic manipulator. In this scenario, the control of the discussed system is challenging, as its dynamical properties undergo continuous and instantaneous changes. Applying first-principles of physics to model complex spacecraft systems is challenging, time-consuming, and financially demanding, accounting for around 50% of total project cost. However, utilizing a data-driven approach to this task could substantially reduce project expenses, thus enhancing the feasibility and sustainability of space exploration.

With the work developed in this thesis, we present the following contributions:

Benchmark of LTV system identification methods: The work completed in this thesis can be considered a benchmark comparing classical system identification methods with machine-learning-based approaches. The discussed methods are evaluated in five scenarios of a spring mass damper system, including nonlinear damping, disturbances, and reconfiguration. We end up showing superiority of machine-learning-based approaches, as this class of methods scored the best in all the considered cases.

Machine-learning-based NLARX system identification: We present how a variety of popular regression methods can be coupled with NLARX to perform the task of system identification. We compare different approaches using MLP, ResNet, RNN, LSTM, XGBoost, and random forest in five varied scenarios of simulations of a spring mass damper system. In the final comparison we show that these methods resulted in the most accurate state propagation accuracy for systems with nonlinear damping.

PINN-based system identification: We present a method to utilize PINNs for system identification of dynamical systems with control inputs and various initial conditions. Once again we benchmark this approach against the others using the five spring mass damper simulations.

Future work

In this section, we present some directions considered to be important as a continuation of the developed work, that we intend to explore in the future:

Evaluation of system identification methods in reconfiguring spacecraft simulation: In our work we intended to create simulations that would mimic the scenario of spacecraft reconfiguration. Even though we find our spring mass damper simulations a good approximation of this concept, it is crucial to further verify the effectiveness of the discussed methods in low- and high-fidelity simulations of reconfiguring spacecraft.

Evaluation of the obtained models of the system in control: After performing system identification it would be interesting to see how the obtained models of the dynamical system would serve the purpose of controller design using for instance an online version of Linear Quadratic Regulator (LQR) or Model Predictive Control (MPC). This would further validate the obtained models and emphasize the limitations of various methods discussed in this thesis.

Experiments with PINNs using more complex architectures: The modified MLP architecture that we used for PINN-based system identification was shallow due to the fact that the minimization of a loss function based on an ordinary differential equation (ODE) is a difficult task which suffers from the problem of vanishing gradients. Therefore, it would be interesting to explore more complex architectures, for instance ones utilizing skip connections to enable the training of deeper networks, thus resulting in more flexible model structure.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[2] ESA. *Building the International Space Station*. URL: https://www.esa.int/Science%7B%5C_%7DExploration/Human%7B%5C_%7Dand%7B%5C_%7DRobotic%7B%5C_%7DExploration/International%7B%5C_%7DSpace%7B%5C_%7DStation/Building%7B%5C_%7Dthe%7B%5C_%7DInternational%7B%5C_%7DSpace%7B%5C_%7DStation3 (cit. on p. 1).

[3] J. Briz Valero et al. "Guidance, Navigation, and Control of In-Orbit Assembly of Large Antennas - Technologies and Approach for IOANT". In: 41st ESA Antenna Workshop on Large Deployable Antennas. 2023 (cit. on p. 1).

[4] M. Carvalho et al. *COSMIC: fast closed-form identification from large-scale data for LTV systems*. 2022. arXiv: 2112.04355 [math.OC] (cit. on pp. 1, 3, 4, 12).

[5] F. Lamnabhi-Lagarrigue et al. "Systems & Control for the future of humanity, research agenda: Current and future roles, impact and grand challenges". In: *Annual Reviews in Control* 43 (2017), pp. 1–64. ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2017.04.001 (cit. on p. 1).

[6] R. V. Jategaonkar. "Introduction". In: *Flight Vehicle System Identification*, pp. 1–24. DOI: 10.2514/5.9781600866852.0001.0024 (cit. on p. 2).

[7] S. D. Fassois and D. E. Rivera. "Applications Of System Identification". In: *IEEE Control Systems Magazine* 27.5 (2007), pp. 24–26. DOI: 10.1109/MCS.2007.904658 (cit. on p. 2).

[8] J. P. Hespanha. *Linear Systems Theory: Second Edition*. Princeton: Princeton University Press, 2018. ISBN: 9781400890088. DOI: 10.23943/9781400890088 (cit. on pp. 2, 3).

[9] G. Pillonetto et al. *Regularized System Identification: Learning Dynamic Models from Data*. Springer Cham, 2022. ISBN: 978-3-030-95859-6. DOI: 10.1007/978-3-030-95860-2 (cit. on pp. 2, 4, 6–8, 16, 18).

[10] G. Pillonetto and A. Aravkin. "A new kernel-based approach for identification of time-varying linear systems". In: *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. 2014, pp. 1–6. DOI: 10.1109/MLSP.2014.6958894 (cit. on p. 3).

[11] Z. Guang, Z. Heming, and B. Liang. "Attitude Dynamics of Spacecraft with Time-Varying Inertia During On-Orbit Refueling". In: *Journal of Guidance, Control, and Dynamics* 41.8 (2018), pp. 1744–1754. DOI: 10.2514/1.G003474 (cit. on p. 3).

[12] B. M. Sanandaji et al. "Compressive System Identification of LTI and LTV ARX models". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. 2011, pp. 791–798. DOI: 10.1109/CDC.2011.6160935 (cit. on p. 7).

[13] L. Li, Y. Pu, and J. Chen. "Maximum Likelihood Parameter Estimation for ARMAX Models Based on Stochastic Gradient Algorithm". In: *2018 10th International Conference on Modelling, Identification and Control (ICMIC)*. 2018, pp. 1–6. DOI: 10.1109/ICMIC.2018.8529965 (cit. on p. 7).

[14] Q. Zhang and L. Ljung. "Multiple steps prediction with nonlinear ARX models". In: *IFAC Proceedings Volumes* 37.13 (2004). 6th IFAC Symposium on Nonlinear Control Systems (NOLCOS 2004), pp. 309–314. ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)31241-7 (cit. on p. 8).

[15] J.-N. Juang and R. S. Pappa. "An eigensystem realization algorithm for modal parameter identification and model reduction". In: *Journal of Guidance, Control, and Dynamics* 8.5 (1985), pp. 620–627. DOI: 10.2514/3.20031 (cit. on p. 9).

[16] J.-N. Juang et al. "Identification of observer/Kalman filter Markov parameters - Theory and experiments". In: *Journal of Guidance, Control, and Dynamics* 16.2 (1993), pp. 320–329. DOI: 10.2514/3.21006 (cit. on p. 10).

[17] M. Majji, J.-N. Juang, and J. L. Junkins. "Time-Varying Eigensystem Realization Algorithm". In: *Journal of Guidance, Control, and Dynamics* 33.1 (2010), pp. 13–28. DOI: 10.2514/1.45722 (cit. on p. 10).

[18] M. Majji, J.-N. Juang, and J. L. Junkins. "Observer/Kalman-Filter Time-Varying System Identification". In: *Journal of Guidance, Control, and Dynamics* 33.3 (2010), pp. 887–900. DOI: 10.2514/1.45768 (cit. on p. 10).

[19] F. B. Carlson, A. Robertsson, and R. Johansson. "Identification of LTV Dynamical Models with Smooth or Discontinuous Time Evolution by means of Convex Optimization". In: *2018 IEEE 14th International Conference on Control and Automation (ICCA)*. 2018, pp. 654–661. DOI: 10.1109/ICCA.2018.8444351 (cit. on p. 11).

[20] J. L. Proctor, S. L. Brunton, and J. N. Kutz. "Dynamic Mode Decomposition with Control". In: *SIAM Journal on Applied Dynamical Systems* 15.1 (2016), pp. 142–161. DOI: 10.1137/15M1013857 (cit. on p. 12).

[21] S. L. Brunton, J. L. Proctor, and J. N. Kutz. "Sparse Identification of Nonlinear Dynamics with Control (SINDYc)". In: *IFAC-PapersOnLine* 49.18 (2016), pp. 710–715. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2016.10.249 (cit. on p. 14).

[22] S. L. Brunton, J. L. Proctor, and J. N. Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113 (cit. on p. 14).

[23] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8 (cit. on p. 14).

[24] J. G. Kuschewski, S. Hui, and S. H. Zak. "Application of feedforward neural networks to dynamical system identification and control". In: *IEEE Transactions on Control Systems Technology* 1.1 (1993), pp. 37–49. DOI: 10.1109/87.221350 (cit. on p. 14).

[25] J. Gonzalez and W. Yu. "Non-linear system modeling using LSTM neural networks". In: *IFAC-PapersOnLine* 51.13 (2018), pp. 485–489. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2018.07.326 (cit. on p. 14).

[26] E. A. Antonelo et al. *Physics-Informed Neural Nets for Control of Dynamical Systems*. 2022. arXiv: 2104.02556 [cs.LG] (cit. on p. 14).

[27] A. Zhang et al. *Dive into Deep Learning*. https://D2L.ai. Cambridge University Press, 2023 (cit. on pp. 17, 24).

[28] I. E. Lagaris, A. Likas, and D. I. Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations". In: *IEEE Transactions on Neural Networks* 9.5 (1998), pp. 987–1000. DOI: 10.1109/72.712178 (cit. on p. 17).

[29] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1994 (cit. on p. 18).

[30] L. Ljung. *System identification (2nd ed.): theory for the user*. USA: Prentice Hall PTR, 1999. ISBN: 0136566952 (cit. on p. 25).

[31] G. Pillonetto et al. "Deep networks for system identification: A survey". In: *Automatica* 171 (2025), p. 111907. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2024.111907 (cit. on p. 30).

[32] S. Wang et al. *An Expert's Guide to Training Physics-informed Neural Networks*. 2023. arXiv: 2308.08468 [cs.LG] (cit. on p. 33).

[33] D. Wolpert and W. Macready. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82. DOI: 10.1109/4235.585893 (cit. on p. 46).