Data-driven Surrogate Modeling of Structural Dynamical Systems via Latent Space Representations

Von der Fakultät Konstruktions-, Produktions- und
Fahrzeugtechnik der Universität Stuttgart und
dem Stuttgarter Zentrum für Simulationswissenschaft
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

von
Jonas Kneifl
aus Bad Mergentheim

Hauptberichter:	Prof. Dr.-Ing. Jörg Fehr
Mitberichter:	Prof. J. Nathan Kutz
Mitberichter:	Prof. Dr.-Ing. habil. Manfred Bischoff

Tag der mündlichen Prüfung: 29. August 2025

Institut für Technische und Numerische Mechanik
der Universität Stuttgart

2025

Preface

When I first sat down to write the speech for my doctoral celebration, I found myself staring at a blank page, wondering how to capture the journey that brought me to this moment. Life, I've come to realize, is nothing more—and nothing less—than a series of events and decisions, each one seemingly small in isolation, yet collectively shaping the trajectory of who we become.

What strikes me as remarkable is how unlikely this particular path has been. If someone had told me years ago to predict my current situation, I'm sure I would have been totally wrong. Yet here we are, and as I look back, I can trace the thread of moments and decisions that led me to this point.

This isn't just a story about academic achievements or professional milestones. It's a story about the people who shaped me: My parents, who gave me the freedom to explore my own path and blessed me with my wonderful siblings—Pierre, Lotta, and Nora—who taught me the value of companionship from the very beginning. It's about friends who shaped my biggest decisions about where and how to live, and colleagues who made the institute feel more like a home than just a workplace.

In particular, I want to give a big shout-out to my advisor, Jörg Fehr, who always showed deep interest not only in my research but also in me as a person and my personal growth. He valued the things I was doing outside academia and gave me the trust to follow my own academic curiosity.

And then there's the Fisch Crew—a group of distinct characters who transformed the institute from a mere office into a big playground. Florian, who patiently listened to countless thought experiments at our whiteboard—sometimes about research, but mostly about life; Julian, the official supervisor of my master's thesis; and of course Hannes, with whom I've started a semifunctional business and who has influenced my life and interests in countless ways and moments.

During my stay in Seattle, I had the privilege of working with Nathan Kutz, whose energy, expertise, and enthusiasm are unmatched. His Monday ritual—scootering into Wilson Annex with freshly roasted coffee beans—is burned into my memory. That was also the time when one of my most rewarding collaborations began with Paolo—a partnership that transcends research and is founded above all in friendship. No matter where our distinct paths may lead us, I'm certain they will remain entangled in ways that no VAE on the world could unravel.

Obviously, in Stuttgart, I got to dive into a lot of exciting research projects as well. I was fortunate to supervise my friend Jonas during his master's thesis and follow his impressive academic journey—though what actually mattered is being a part of his life. Working on projects with Johannes and trying to make sense of differential geometry was a lot of fun, too. I'm especially thankful for all the times he read through my thesis, brainstormed ideas with me, and also gave advice on personal matters.

There are so many other people and memories at the institute worth mentioning: Night excursions to rooftops, THE BÜRO, the birth of `mensa_bot` enabled by many coding days at Dreikönigstag with Tom and Niklas, walks with Andi when I needed advice, countless Fisch evenings, organizing YMMOR2024, SimTech and ITM Statusseminars, and unforgettable conferences in Vienna and San Diego. Thanks to Philipp, who in a sense took over from Florian as my Esslinger office-mate and helped me prepare my doctoral celebration; to Ben, for raving through Poland with me; to Jan, for taking over my doctoral hat-making duties; and to Norman, Pakka, Micha, Peter, and Peter, who keep the institute running.

As I prepare to close this particular chapter of my life, I find myself filled with both melancholic gratitude for what has been and excited curiosity about what lies ahead. Most importantly, I'm acutely aware of what I will miss most—the daily presence of the people who have made this journey not just possible, but meaningful.

What follows is my attempt to honor that journey and the remarkable people who shared it with me.

– *for you*

Contents

Kurzfassung

Numerische Simulationen verfügen über ein hohes Maß an prädiktiver Leistungsfähigkeit, um das dynamische Verhalten komplexer Systeme zu analysieren. Außerdem ermöglichen sie Erkenntnisse in Bezug auf physikalische Phänomene, die andernfalls nicht zugänglich wären. Entsprechend spielt die simulationsbasierte Analyse strukturdynamischer Systeme eine zentrale Rolle in der technischen Entwicklung oder der Bewertung von Sicherheitsaspekten und kann bei der fundierten Entscheidungsfindung unterstützen. Der signifikante Rechenaufwand konventioneller High-Fidelity (HF)-Modelle begrenzt jedoch deren Einsatz – insbesondere in Mehrfach-Abfrage-Szenarien, Echtzeitanwendungen oder ressourcenbeschränkten Umgebungen.

Dies motiviert die Entwicklung sogenannter Surrogatmodelle – effizienter Approximatoren, die das wesentliche Verhalten der HF Simulationen nachbilden und dabei den Rechenaufwand drastisch reduzieren. Der Fokus dieser Arbeit liegt auf nicht-intrusiven, datengetriebenen Surrogatmodellierungsansätzen, welche die Synergie zwischen klassischen numerischen Verfahren und modernen datenbasierter Techniken nutzen.

Die in dieser Arbeit präsentierten Methoden sind in einem einheitlichen Rahmen integriert, der Konzepte der Modellordnungsreduktion und des maschinellen Lernens vereint. Im Zentrum dieses Rahmens steht die Konstruktion kompakter, niederdimensionaler latenter Repräsentationen, in denen sich die Dynamik komplexer, hochdimensionaler Systeme effizient erlernen und genau vorhersagen lässt. Die vorliegende Arbeit untersucht verschiedene Strategien zur Identifikation geeigneter Koordinatenrepräsentationen und analysiert mehrere Ansätze zur Modellierung latenter Dynamiken.

Zu den analysierten Ansätzen zählen sowohl Black-Box-Methoden als auch Verfahren der Systemidentifikation, die analytische und interpretierbare Gleichungen liefern. Die Anwendungsbeispiele decken ein breites Spektrum strukturdynamischer Probleme ab: Sie reichen von einfachen akademischen Beispielen wie nichtlinearen Pendeln, Masse-Feder-Dämpfer-Systemen oder niedrigdimensionalen chaotischen Systemen bis hin zu komplexen, hochdimensionalen Mehrkörpersystemen sowie Finite-Elemente-Modellen. Letztere umfassen gekoppelte Simulationen sowie hochgradig nichtlineare sowie irreguläre Systeme mit Kontakt—etwa in vereinfachten Crashszenarien.

Zu den wesentlichen Beiträgen dieser Arbeit zählen ein systematischer Leistungsvergleich von Dimensionsreduktions- und Regressionsverfahren, eine multi-resolutionale Surrogatmodellierungsstrategie zur Erfassung von Mehrskalenphänomenen mit variierenden Hardwareanforderungen sowie strukturerhaltende Modellidentifikationsverfahren. Darüber hinaus wird ein variationales generatives Modellierungsframework entwickelt, das interpretierbare Surrogatmodelle samt Quantifizierung der Unsicherheit ermöglicht.

In ihrer Gesamtheit gestatten diese Ansätze die Entwicklung von Surrogatmodellen, die nicht nur recheneffizient und genau sind, sondern auch interpretierbar und in Echtzeit-, Mehrfach-Abfrage- und ressourcenbeschränkten Szenarien einsetzbar sind. Damit erweitern sie die Anwendbarkeit und Zugänglichkeit numerischer Simulationen in Wissenschaft und Technik.

Abstract

Numerical simulations provide powerful predictive capabilities to analyze the dynamic behavior of complex systems and offer insights into physical phenomena that would otherwise remain inaccessible. Consequently, simulation-based analysis of structural dynamical systems plays a crucial role in engineering design, safety verification, and informed decision-making. However, the significant computational cost of conventional High-Fidelity (HF) models limits their applicability—especially in multi-query scenarios, real-time contexts, or resource-constrained environments.

This motivates the development of surrogate models—efficient approximations that emulate the essential behavior of HF simulations while drastically reducing computational overhead. This thesis focuses on non-intrusive, data-driven surrogate modeling approaches that integrate classical numerical methods with modern Machine Learning (ML) techniques, thereby harnessing the synergy between scientific principles and data-driven technologies.

In detail, the proposed methods are embedded within a unified surrogate modeling framework that combines concepts from Model Order Reduction (MOR) and ML. At the heart of this framework lies the construction of compact, low-dimensional latent representations in which the dynamics of complex, high-dimensional systems can be efficiently learned and accurately predicted. The thesis explores various strategies for identifying suitable coordinate representations and investigates multiple techniques for modeling latent dynamics, including both black-box approximations and system identification approaches that yield analytic, interpretable equations.

The application examples cover a broad spectrum of structural dynamics problems: they range from simple academic cases—such as nonlinear pendulums, mass-spring-damper systems, or low-dimensional chaotic systems—to complex, high-dimensional multi-body systems and finite element models. The latter include coupled simulations as well as highly nonlinear and irregular systems involving contact, as found in simplified crash scenarios.

Key contributions include a systematic benchmarking of dimensionality reduction and regression techniques, a multi-resolution surrogate modeling strategy capable of capturing multi-scale phenomena under varying hardware constraints, as well as structure-preserving model discovery methods. Furthermore, a variational generative modeling framework is developed to enable interpretable surrogate modeling under uncertainty.

Collectively, these approaches enable the development of surrogate models that are not only computationally efficient and accurate, but also interpretable and deployable in real-time, multi-query, and resource-constrained settings—thereby broadening the applicability and accessibility of numerical simulation in science and engineering.

Chapter 1

Introduction

No matter what you do
You change someone
One changes you
Do you have control?

Wet Paint, Do you decide?

The Hare and the Hedgehog is a Low Saxon fable written down by the Brothers Grimm [Grimm56]. It tells the story of an arrogant hare who mocks the hedgehog for his short, crooked legs. In response, the hedgehog challenges him to a race. Confident in his speed, the hare accepts and sprints ahead—only to find, at the finish line, the hedgehog already waiting. Unbeknownst to the hare, the hedgehog's wife—who looks just like him—had been stationed there from the beginning. The hare repeats the race again and again, but each time meets the same fate: "I'm already here," the hedgehog's double calls out. Variants of this story exist across many cultures, such as Aesop's *The Tortoise and the Hare* in ancient Greece or *The Rabbit and the Tortoise* in East Asia, all of which center around the idea that cleverness and strategy can triumph over speed and brute force.

Yet, in the Grimm version, the hedgehog's victory is not merely strategic—it's deceptive. He has to cheat to level the playing field. In an ideal world, however, one wouldn't have to choose between raw speed and cleverness. Instead, one would combine both: the swift execution of the hare with the cunning of the hedgehog.

In computational science, this contrast maps naturally onto two major paradigms. Conventional numerical simulation methods resemble the hedgehog: grounded in physical principles and capable of delivering deep insights, but often computationally expensive and slow. Machine Learning (ML), by contrast, is like the hare: remarkably fast and flexible, but frequently used in a black-box manner, lacking interpretability and physical

grounding. The central aim of this work is to unite these two worlds. We seek to extract the rich physical knowledge embedded in traditional simulation schemes and distill it into efficient, data-driven models using ML. In a way, the goal is to build a new kind of runner: a clever and fast descendant of both hare and hedgehog.

But first things first: Why do computational science and the simulation of physical phenomena matter in the first place? What is the fundamental role of numerical simulations in modern engineering and science?

On the one hand, simulations provide predictive power. They offer a means of anticipating how physical systems respond under specific scenarios. This foresight enables optimization in system design, informs decision-making, and plays a critical role in safety assessments. On the other hand, simulations serve as a kind of virtual microscope, offering insights that would be difficult—or even impossible—to obtain through real-world experiments, whether due to issues of scale, ethics, or cost. A notable example of the importance of this "microscopic" capability can be found in a pivotal event of the 20th century.

One of the early drivers of simulation-based structural analysis was not success, but failure: the collapse of the Tacoma Narrows Bridge in 1940. Driven by wind-induced forces, the bridge began to oscillate vertically and ultimately collapsed. This dramatic event exposed the limitations of purely static analysis methods in capturing the complexities of dynamic, real-world interactions. Subsequent investigations, including wind tunnel experiments, revealed that wind-induced oscillations was the key contributor to the failure. In the years that followed, engineers increasingly turned to early computers and numerical simulation techniques to study structural dynamics in ways that neither theory nor experiment alone could provide. Today, we are capable of simulating a wide range of dynamic phenomena to gain deeper insights into the physical world and analyze why the bridge collapsed with numerical experiments [Larsen00, ArioliGazzola15].

This example also highlights the predictive power of modern simulation-based structural analysis. Rather than relying solely on expert knowledge or static computations, engineers can now simulate fatigue, failure, and load-bearing capacity under realistic conditions—before a structure is even built. A contemporary illustration of this approach can be found in the structural design of the Stuttgart 21 underground station. The station features a basement and an architecturally integrated shell roof, both constructed from reinforced concrete. A High-Fidelity (HF) 3D numerical simulation model was employed to assess loads and constraining forces acting on the structure and served as a foundation for the structural design of the reinforced concrete elements [BechmannSchmid24].

One particular branch of structural analysis that forms the fundamental domain within this thesis is structural dynamics. It concerns the behavior of physical structures under dynamic loading—such as vibrations, impacts, or varying environmental conditions. Applications are wide-ranging, spanning aerospace, automotive engineering, and biomedical systems.

One technique that has revolutionized this field is the Finite Element (FE) method which enables the accurate prediction of structural behavior under various loads and conditions. It provides a robust and HF representation of complex systems, capable of modeling systems such as vehicle crashes or physiological dynamics in human tissues. However, this accuracy comes at a cost: Conventional simulation methods typically require significant computational resources, long runtimes, and high hardware demands as well as associated energy consumption. When using such methods, we are often confronted with a fundamental trade-off—the high-fidelity, high-cost dilemma.

The associated computational burden significantly limits the applicability of conventional HF simulation methods in many real-world scenarios. One such class of scenarios involves *multi-query problems*, where simulations must be performed repeatedly—for example, in design loops, optimization routines, or safety evaluations. Consider crash simulations: It is insufficient to simulate a single, idealized collision. Instead, a comprehensive analysis requires simulations across a range of impact speeds, angles, and collision partners to gain a realistic scenario-based understanding of vehicle safety.

In addition, many modern applications demand *real-time capability*. In control systems, for instance, the system response must be computed on the fly to determine optimal control actions that compensate for deviations from desired behavior. A further example is *digital twins*, where simulation models run in parallel with their physical counterparts to enable real-time monitoring, early fatigue prediction, and predictive maintenance.

Moreover, there are increasing demands for simulation in contexts with *limited computational resources*, such as embedded systems or edge devices. Examples include smartphones, where privacy concerns or latency constraints necessitate on-device computations, and extended reality headsets, where power and performance budgets are inherently constrained.

These diverse requirements for speed, responsiveness, and resource efficiency highlight the limitations of conventional simulations and motivate the use of *surrogate models*: computationally efficient approximations that retain essential features of HF models while enabling faster execution. There are various types of surrogate models, each offering computational efficiency through different mechanisms. Some are *physics-based*, relying on simplified representations of the governing equations or problem-specific assumptions. Others are purely *data-driven*, learning system behavior solely from input-output observations.

One of the most powerful and widely used research fields to derive surrogate models is Model Order Reduction (MOR). This field is primarily concerned with reducing the dimensionality of a system—that is, decreasing the number of variables required to describe its behavior—without sacrificing accuracy. MOR techniques typically achieve this by constructing a reduced basis that captures the dominant features of the system's dynamics, often through *projection-based methods*.

In the context of MOR, it is common to distinguish between *intrusive* and *non-intrusive* approaches. Intrusive methods operate directly on the mathematical operators of the HF model and therefore require access to the internal structure of the simulation code. They accelerate computations by reformulating the governing equations within a reduced subspace. Non-intrusive methods, by contrast, treat the original model as a black-box and learn a surrogate solely from data generated via parametric simulations or other input-output evaluations. These different approaches involve trade-offs in terms of interpretability, computational speed, and generalization capabilities. The choice of surrogate model depends not only on the desired accuracy and efficiency but also on the accessibility of the underlying model and the availability of training data.

When it comes to surrogate modeling of structural dynamical systems, several specific challenges arise, which will be addressed throughout the course of this thesis. A fundamental challenge shared across many topics in this work is that of *high dimensionality*. Conventional numerical modeling schemes achieve their accuracy through fine spatial and temporal discretizations, leading to extremely high-dimensional representations of the system state. In addition, many widely used software solutions for the simulation of those systems are commercial, meaning that access to the underlying computational operators is restricted due to proprietary code or intellectual property constraints. As a result, the HF model often behaves as a black-box simulator. In other cases, the governing equations themselves may be only partially known—or entirely unknown—further complicating the development of surrogate models based on physical insight.

Another layer of complexity arises from the multi-physics and multi-scale nature of many structural dynamical systems. These systems often involve coupled phenomena acting across different spatial and temporal scales, requiring surrogate models that can handle such heterogeneity. All in all, the focus of this thesis lies on the development of surrogate models for high-dimensional, inaccessible systems, where only input-output data from simulations is available. This situates the work firmly within the domain of *non-intrusive* surrogate modeling.

However, learning from data poses its own challenges. The data may be corrupted by noise—stemming from measurement errors or numerical artifacts, such as those introduced by numerical differentiation. Moreover, generating high-fidelity data is computationally expensive, which means the available data is often limited in both quantity and quality. The goal is to build surrogate models that remain efficient and accurate despite these limitations—and ideally still respect key physical principles and constraints.

To address the aforementioned challenges, the methods introduced in this thesis are based on a unified conceptual framework that combines essential principles of MOR with recent advances in scientific ML. At their core, these methods aim to learn a low-dimensional *latent representation* that enables efficient and accurate prediction of a system's dynamical behavior. This involves two key steps: First, a (possibly nonlinear) coordinate

transformation compresses essential information of the high-dimensional system state into a compact and tractable latent space. Second, the temporal evolution of the system is approximated directly in this new set of low-dimensional coordinates, potentially as a function of time and system parameters.

This procedure is schematically illustrated in Fig. 1.1. Given high-fidelity data $\boldsymbol{x}$, a *reduction mapping* $\boldsymbol{\phi}$ transforms the data from the physical state space into a latent space, yielding latent variables $\boldsymbol{z}$. A corresponding *reconstruction mapping* $\boldsymbol{\psi}$ allows for the recovery of physical variables from the latent space. Within the latent space, the dependency of the temporal evolution of the reduced system states on time t and parameters $\boldsymbol{\mu}$ are resolved through ML techniques. The mathematical foundations and algorithmic details of this process are discussed in the following chapters.

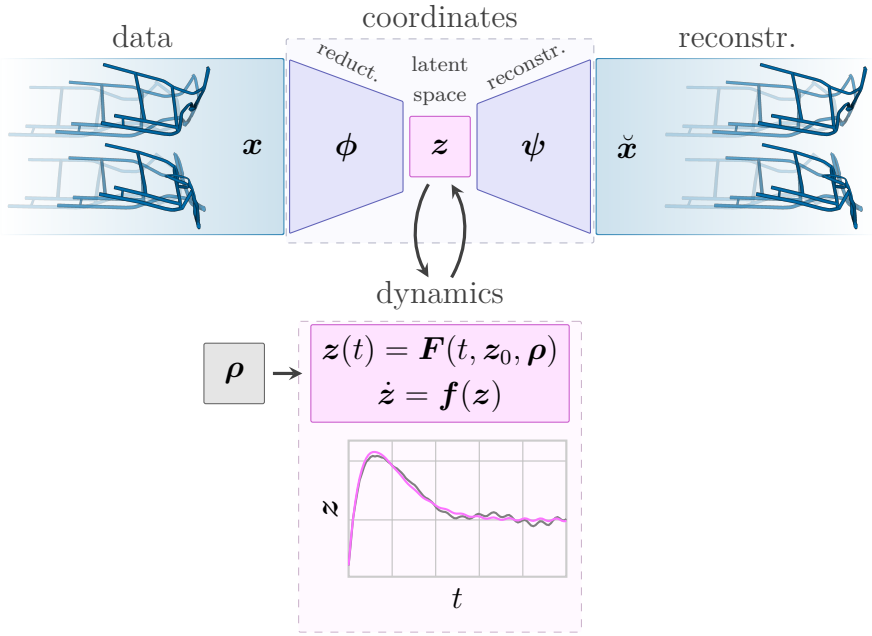


Figure 1.1: Schematic illustration of the high-level idea for the surrogate modeling procedures introduced throughout this thesis. The central paradigm is to compress essential information about the system into a latent space where the dynamics can be more efficiently captured.

Throughout this thesis, we explore different instantiations of this framework. Some approaches treat the latent dynamics as a pure black-box model and are grouped under the umbrella term Latent Approximation of Dynamics (LADy). Others aim to extract explicit knowledge from the data by discovering analytic governing equations or latent-space operators—these are referred to as Latent Discovery of Dynamics (LDD). Specifically, those methods address the question of how the nonlinear, parametric, and transient behavior of structural dynamical systems can be captured using a compact coordinate representation with as few variables as possible. Moreover, we investigate how to identify meaningful and parsimonious analytical equations that govern the latent dynamics of

(potentially high-dimensional and parametric) systems. While the focus is on structural dynamics, the methodology introduced are applicable to broader classes of dynamical systems. This thesis does not address real-time experimental coupling, evaluation on edge computing devices, or control loop integration, which are left for future work.

In conclusion, the surrogate models developed in this thesis aim to distill knowledge from HF physical simulation data. This is achieved by partially retaining physical consistency, accounting for multi-physics and multi-scale phenomena, preserving system-theoretic properties, and—most importantly—enabling fast yet accurate simulations for multi-query tasks, real-time applications, and hardware-constrained environments. In a sense, one could say that the hedgehog teaches the hare—so that it may combine cleverness with speed—without relying on deception or brute force.

1.1 Content of this Thesis

This thesis is structured into three main parts. The first part provides the foundational knowledge necessary for the subsequent chapters. It explains why surrogate models are essential in computational science and introduces the core techniques used in surrogate modeling. This includes how high-dimensional system states can be compressed into low-dimensional representations, and how their temporal evolution—i.e. the system dynamics—can be captured using different modeling approaches.

The second part focuses on black-box surrogate modeling techniques. Here, various combinations of MOR and ML are rigorously evaluated, including approaches tailored to sequential data. In addition, a multi-hierarchic framework is introduced, in which several surrogate models of varying resolution and fidelity are constructed to address different aspects of the simulation task.

The third and final part of the thesis explores how analytical equations and latent-space operators can be extracted from high-fidelity data. These methods place special emphasis on preserving system-theoretic properties and are particularly designed to address uncertainty—both in the data and in the resulting models. The specific contents of the chapters are outlined below, providing a structured overview of the topics covered and the progression of the methodology and applications presented in this thesis.

Part I Preliminary deals with the fundamentals in surrogate modeling, MOR, and scientific ML.

Chapter 2 The Heart of Surrogates: Fundamental Concepts for Modern Modeling introduces the mathematical foundations for modeling physical processes, with a particular focus on structural dynamics. It then presents HF modeling techniques and introduces first surrogate modeling concepts.

Chapter 3 Shrinking the Problem: Identification of Low-dimensional Representations focuses on the challenge of dimensionality in numerical simulation models. It explores the reasons behind high-dimensional representations in classical numerical schemes and introduces methods for identifying low-dimensional, yet expressive system descriptions. In particular, the chapter discusses MOR as a key approach for achieving efficient reduced-order representations.

Chapter 4 Dynamics Decoded: From Black-box Approaches to System Identification addresses the second essential component of surrogate modeling: the approximation of system dynamics. This chapter explores strategies for modeling dynamics, including black-box modeling and system identification. It distinguishes between approaches that predict system states directly and those that aim to reconstruct underlying governing differential equations of the system.

Part II Latent Approximation of Dynamics discusses various black-box methods to create efficient surrogate models.

Chapter 5 Learning in Reduced Coordinates: Model Reduction Meets Machine Learning introduces a general surrogate modeling framework that systematically combines MOR and ML. It evaluates a variety of dimensionality reduction and regression techniques, providing practical guidance on which combinations are most effective under different conditions. The chapter includes a rigorous comparison of the resulting model variants and also proposes an approach to handle time-varying input parameters.

Chapter 6 Multi-hierarchic Surrogate Modeling introduces a multi-hierarchical surrogate modeling framework designed to efficiently capture both macro- and microscale features while addressing the challenge heterogeneous computing environments by constructing a series of surrogates at varying resolutions. For this procedure, the foundations of Graph Convolutional Neural Networks (GCNNs) and mesh simplification are discussed, followed by the application of the Multi-Hierarchical (MH) framework to a numerical case.

Part III Latent Discovery of Reduced Order Models shifts from black-box surrogate modeling to approaches that explicitly discover the governing equations of high-fidelity systems.

Chapter 7 Structure-preserving Latent Discovery addresses the task to derive operators defining interpretable and predictive Reduced Order Models (ROMs) directly from data. The chapter introduces a structure-preserving model discovery technique based on the Port-Hamiltonian (PH) formulation, which inherently preserves system-theoretic properties and is particularly well-suited for multi-physics systems. The fundamentals of PH systems are presented, followed by a detailed description of the proposed method. The chapter concludes with numerical examples demonstrating the

effectiveness of the approach.

Chapter 8 Generative, Physically Consistent and Uncertainty-Aware Latent Discovery introduces a novel physical generative modeling framework that unifies dimensionality reduction, dynamical system identification, and uncertainty quantification within a variational setting. This approach enables the discovery of interpretable, compact, and physically consistent ROMs from limited and noisy high-dimensional data. The chapter presents key components of the framework, including a variational autoencoder for latent space modeling and a newly developed variational extension of Sparse Identification of Nonlinear Dynamics (SINDy) prior to results for a set of numerical benchmarks.

The content of this thesis, hence, bridges the gap between physical modeling and ML by combining dimensionality reduction, and data-driven dynamics approximation. The proposed surrogate modeling methods extend the reach of traditional simulations to real-time, multi-query, and hardware-constrained applications—paving the way for a new generation of physics-aware, data-driven engineering tools. The key novelties, key publications, and contributions of this work can be summarized as:

i) A general and extensible surrogate modeling framework for efficient, non-intrusive modeling of high-dimensional structural dynamical systems.

ii) Rigorous benchmarking of dimensionality reduction and dynamics approximation techniques.

iii) Development of a multi-hierarchical surrogate modeling approach to efficiently capture multi-scale effects and varying resolutions within a single framework, cf. [KneiflEtAl24].

iv) Introduction of a structure-preserving model discovery method with guaranteed physical properties, cf. [RettbergEtAl24].

v) Development of a generative framework to integrate uncertainty quantification into interpretable surrogate modeling, cf. [ContiEtAl24].

Part I

Preliminary

Chapter 2

The Heart of Surrogates: Fundamental Concepts for Modern Modeling

So follow, follow the sun
And which way the wind blows
When this day is done...

Xavier Rudd, Follow the Sun

Information is the foundation of sound decision-making: A physician must be informed about the patient's complaints and symptoms to accurately diagnose him, a politician must be knowledgeable about the situation in his country to effectively steer its destiny, and an engineer must understand the properties of a material to assess its reliability in a technical system. For many complex phenomena, however, the extraction of information is not straightforward, not feasible at all, or requires the use of costly experimental procedures.

Fortunately, mathematics provides a remedy by offering a language that is capable of describing the underlying (physical) laws, thereby facilitating a deeper understanding. In this language, an idealized system, referred to as a mathematical model, is formulated to represent the physical system. Numerical simulations of such models give us the possibility to undertake simulations of intricate phenomena, gain insights into complex dynamical problems, to use them for optimization, control, uncertainty quantification, or to analyze the system's behavior under varying, even extreme, conditions. Thus, the so generated knowledge can help in informed decision-making.

Structural dynamics [CraigKurdila06] is one area in which such models and simulations play a crucial role. They are used in the development of technical prototypes, e.g. for determining weaknesses in structure design or the optimization of material

use, but also represent a ubiquitous tool as digital twin for an entire product life cycle [HartmannAuweraer21, ChinestaEtAl20]. In particular, structural dynamics represents a specific branch of structural analysis that pertains to the behavior of a physical structure when subjected to dynamic forces or loading due to natural phenomena and human activities. Such structures can range from a simple steel pipe to full-scale vehicles constructed from a variety of materials, and from man-made objects like machines to biological systems such as musculoskeletal systems. The response of these structures to loading often involves forces that are dependent on both displacements and velocities.

This, typically, results in highly nonlinear governing equations of the dynamic system described as Partial Differential Equations (PDEs) [PazKim19]. PDEs are exceedingly challenging to solve mathematically and entail a significant computational burden. Usually they are discretized in space, often resulting in an extremely large system of Ordinary Differential Equations (ODEs). However, the computational costs associated with these models limit the resolution of geometric details and the complexity of the considered physical laws. This either renders the model unfeasible for certain applications or leads to models that are unusable for simulation-based decisions due to a vast of simplifying assumptions.

Nevertheless, a multitude of user groups, including domain experts such as researchers, engineers, and physicians, as well as end-users such as car drivers and athletes, require direct access to insights derived from High-Fidelity (HF) simulations, i.e. simulations of high accuracy and strong expressiveness. Consequently, the knowledge extracted from an initial HF model developed by domain experts must be disseminated to other domain experts or individual end-users. To achieve this, simulation models must be adaptively tailored to accommodate the diverse computing environments and applications all offering different time and hardware limitations. As a consequence, the used models often must be real-time capable, resource-saving, or suitable for multi-query evaluations. This is one of the primary motivations for the development of efficient yet accurate surrogate models that are capable of capturing the high-fidelity counterparts in terms of expressiveness, while also being cost-effective in terms of evaluation.

Surrogate modeling techniques can be classified into two categories: intrusive and non-intrusive. The former require access to system information and manipulate the original high-dimensional governing equations and the original model (hence the term ”intrusive”), whereas the latter treat the HF model as a black-box and solely require input-output data. This makes non-intrusive methods especially suited for scenarios where the original governing equations are either not known or where they remain hidden due to closed-source software, which is often the case in commercial software.

In the first scenario, the sole remaining source material for the derivation of a surrogate model is data in the form of real-world measurements. In contrast, in the second scenario, the user is ultimately left with data in the form of simulation results. Once a surrogate

model is derived in either way it can be queried in the desired application. The complete workflow, which details the process of transforming a physical system into an efficient surrogate simulation, is illustrated in Fig. 2.1. It should be noted that alternative types of surrogate models exist, such as those based on simplified physical representations—e.g. mass-spring-damper systems—that approximate the behavior of high-fidelity models. However, this work is not concerned with such types of surrogates.

Prior to an in-depth examination of surrogate models and the methodologies employed to derive them, this chapter describes how a physical process can be described using mathematics first. This is followed by a more detailed analysis of a popular modeling approach in structural dynamics, namely the Finite Element Method (FEM), and according models.

2.1 High-fidelity Models

Many physical phenomena can be described using parametrized PDEs of the form

$$\boldsymbol{x}_t = \boldsymbol{f}_{\text{PDE}}(\boldsymbol{x}, \boldsymbol{x}_{\chi_1}, \dots, \boldsymbol{x}_{\chi_{n_\chi}}, \boldsymbol{x}_{\chi_1\chi_1}, \dots, t, \boldsymbol{\chi}; \boldsymbol{\mu}) \tag{2.1}$$

where the solution $\boldsymbol{x}(t, \boldsymbol{\chi}; \boldsymbol{\mu}) \in \mathcal{X}$ is a function that depends on the temporal $t \in \mathcal{T}$ and spatial coordinates $\boldsymbol{\chi} = \left[\chi_1, \dots, \chi_{n_\chi}\right] \in \Upsilon \subset \mathbb{R}^{n_\chi}$ (usually a 3-dimensional space $\boldsymbol{\chi} = [\chi_1, \chi_2, \chi_3]$ is used, $n_\chi = 3$). Moreover, it is parametrized by external inputs or parameters $\boldsymbol{\mu} \in \mathcal{P}$ defined on a *parameter space* $\mathcal{P} \subset \mathbb{R}^{n_\mu}$, $n_\chi, n_\mu \in \mathbb{N}_0$. Those parameters can represent material properties, boundary conditions, or (time-dependent) forcing terms. The nonlinear evolution operator $\boldsymbol{f}_{\text{PDE}} : \mathcal{X} \times \mathcal{T} \times \Upsilon \times \mathcal{P} \to \mathcal{X}$ of (2.1) connects the first order time derivative $\boldsymbol{x}_t = \frac{\partial \boldsymbol{x}(t, \boldsymbol{\chi}; \boldsymbol{\mu})}{\partial t}$ of the state variable $\boldsymbol{x}$, i.e. the temporal evolution, to spatial dependencies including partial derivatives like $\boldsymbol{x}_{\chi_i} = \frac{\partial \boldsymbol{x}(t, \boldsymbol{\chi}; \boldsymbol{\mu})}{\partial \chi_i}$ or $\boldsymbol{x}_{\chi_i\chi_i} = \frac{\partial^2 \boldsymbol{x}(t, \boldsymbol{\chi}; \boldsymbol{\mu})}{\partial \chi_i^2}$ and other factors. Please note that a PDE is, in general, well-defined only when accompanied by appropriate boundary conditions and, for time-dependent problems, an initial condition of the form $\boldsymbol{x}(0, \boldsymbol{\chi}, \boldsymbol{\mu}) = \boldsymbol{x}_0$, where $\boldsymbol{x}_0$ specifies the state of the system at the initial time.

PDEs are used in various scientific fields to describe physical phenomena: Maxwell's equations form the basis of electromagnetics, Navier-Stokes equations describe fluid-gas dynamics, and Schrödinger's equation can be used to calculate the propagation of waves in quantum mechanics. Another common example is the heat equation

$$\boldsymbol{x}_t = \mu \left(\boldsymbol{x}_{\chi_1\chi_1} + \dots + \boldsymbol{x}_{\chi_{n_\chi}\chi_{n_\chi}} \right), \tag{2.2}$$

which finds application not only in heat transfer, but also in diffusion processes, population dynamics, and many others. In engineering contexts, the parameter $\mu > 0$ describes the thermal diffusivity. However, for most PDEs no analytic solution exists and solving them

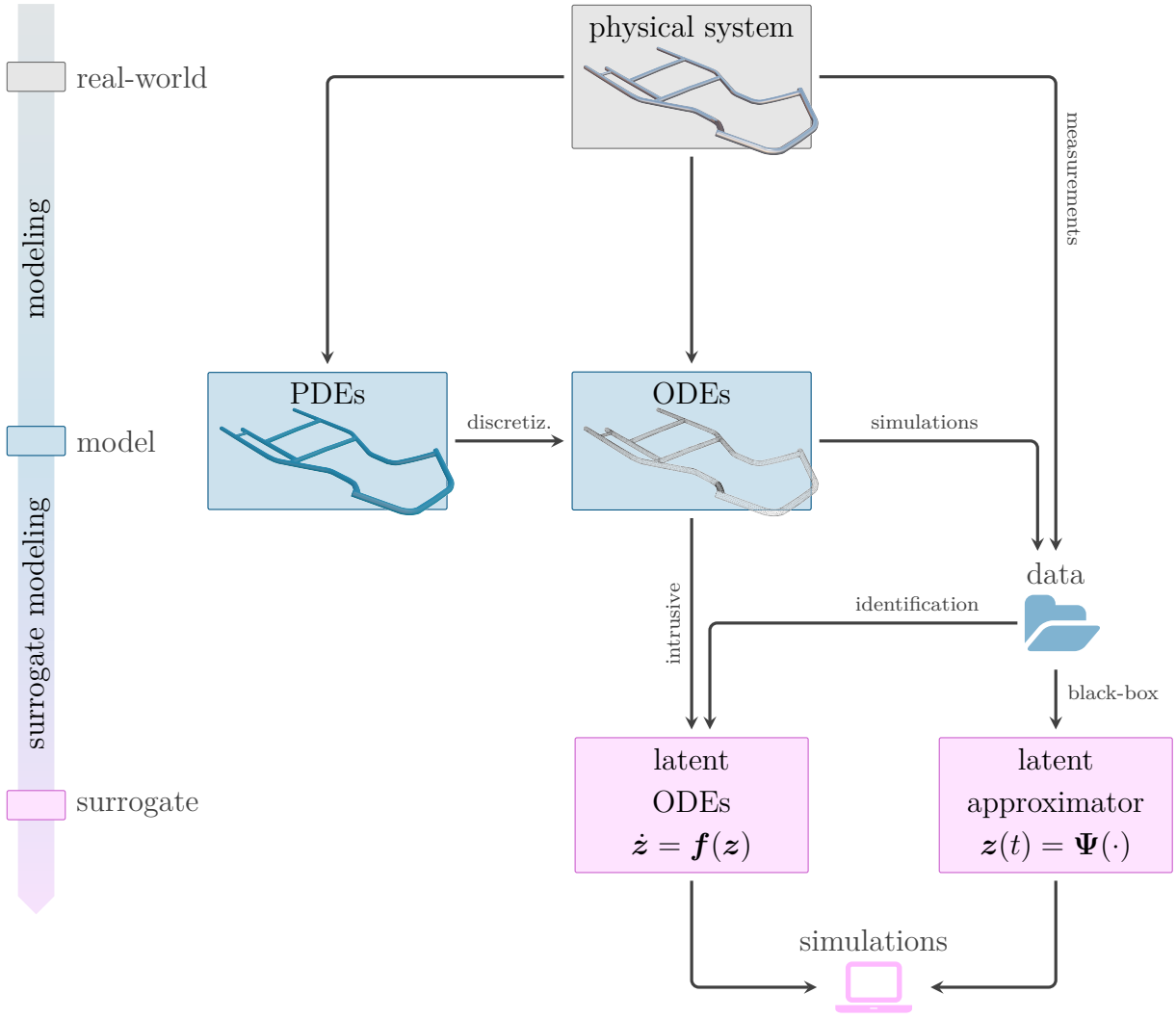


Figure 2.1: Representation of the surrogate modeling workflow, from the physical system to the generation of efficient surrogate models. The physical system is either modeled (as a partial differential equation that is discretized as an ordinary differential equation or as an ordinary differential equation directly) or measurements are taken to obtain a dataset. Furthermore, the simulation models can be queried to obtain data. In non-intrusive surrogate modeling, this data is utilized to obtain latent approximators via black-box modeling or to obtain latent governing equations via system identification. Alternatively, latent models can be obtained through intrusive surrogate modeling methods. Once surrogate models are created, they can be employed as rapid simulators, for instance, in multi-query applications.

numerically on a continuous domain is difficult. As a consequence, PDEs are usually discretized with respect to their spatial coordinates, which means that the spatial domain is divided into a finite number of points, transforming it into a system of ODEs. In order to achieve this objective, it is often necessary to derive the weak form of the differential equation by multiplying the equation with a test function and integrating over the domain. This process ensures that the solution satisfies the equation in an averaged sense and also lowers the derivative order, making it more suitable for numerical approximation. The solution is then approximated as a combination of basis functions with unknown coefficients, which are computed at the discrete points using numerical algorithms. This approach not only enables the computation of the solution at discrete points but also allows for the reconstruction of the approximate solution throughout the entire domain. Consequently, discretizing the weak form of PDEs in space is a fundamental step in solving them numerically.

Once the system (2.1) is discretized, it can be formulated as a set of ODEs

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t,\boldsymbol{\mu}) &= \boldsymbol{f}(t,\boldsymbol{x}(t,\boldsymbol{\mu}),\boldsymbol{\mu}), \quad t \in \mathcal{T}, \\ \boldsymbol{x}(0,\boldsymbol{\mu}) &= \boldsymbol{x}_0(\boldsymbol{\mu}) \end{aligned} \tag{2.3}$$

defined on a *time interval* $\mathcal{T} := [t_0, t_{\text{end}}]$ with $t_{\text{end}} > t_0$. The state variables live on a n-dimensional vector space $\mathcal{X} \cong \mathbb{R}^n$, referred to as *state space*. The right-hand side $\boldsymbol{f}$ of the set of ODEs can be queried to obtain the temporal evolution of the state variables starting from the initial value $\boldsymbol{x}_0$. Typically, the dimension $n \gg 1$ must be chosen extremely large for the sake of approximation quality. Hence, system (2.3) is often denoted as Full Order Model (FOM). The dimensionality renders the model computationally expensive and drives the demand for more efficient surrogate models.

Mesh-based discretization approaches are very common and operate on a connected mesh of nodes. Examples are the finite difference method, the finite element method, and the finite volume method. In contrast, mesh-free methods, as the name implies, do not necessitate the use of a mesh. A notable example is the method of smoothed-particle hydrodynamics, wherein data points are treated as physical particles.

2.1.1 Finite Element Method

In addition to numerous other numerical techniques for addressing structural-dynamical challenges, the FEM has emerged as one of the most prevalent approaches over the past few decades. In mathematical terms, it is a numerical discretization method that transforms continuous PDEs into a finite set of ODEs. This is accomplished through the fundamental concept of FEM, which entails the construction of a complex, continuous system from a multitude of simpler components, known as finite elements. An exhaustive overview over the FEM can be found in [BelytschkoEtAl14] or [ZienkiewiczTaylorFox14].

Finite elements are typically distinguished by a relatively simple geometry, encompassing structures such as trusses, beams, plates, and shells. The number of nodes at which each of these elements is connected is dependent on its geometry. The local equations of motion for one element can be calculated by taking properties such as elasticity, inertia, and forces into account. Subsequently, the global equations of motion for the overall system are assembled by combining the entirety of all elements. In the initial stage of the analysis, each finite element is treated as a free body devoid of external constraints. With the aid of material laws, trial functions, and d'Alembert's principle, the local equations of motion are identified. The selected trial functions are typically polynomials. In the context of structural problems, they represent the displacement of a point within the element through the interpolation of the nodal values associated with the element. Please note that the FEM can also be used for a wide range of other domains, e.g. electromagnetic or fluid dynamics. Accordingly, other physical quantities as displacements can be considered as well.

Once the individual equations are assembled in a global definition, the governing equation of a nonlinear dynamical mechanical system read

$$\boldsymbol{M}\left(\boldsymbol{d}(t)\right)\ddot{\boldsymbol{d}}(t)+\boldsymbol{\zeta}_{\text{int}}\left(\boldsymbol{d}(t),\dot{\boldsymbol{d}}(t)\right)=\boldsymbol{\zeta}_{\text{ext}}(t) \tag{2.4}$$

with the n-dimensional quadratic and possibly configuration-dependent mass matrix $\boldsymbol{M}$: $\mathbb{R}^n \rightarrow \mathbb{R}^{n\times n}$, the vector of internal forces $\boldsymbol{\zeta}_{\text{int}} : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ that arise within the material or structure due to deformation, and the vector of external forces $\boldsymbol{\zeta}_{\text{ext}} : \mathbb{R} \rightarrow \mathbb{R}^n$ applied to the system, e.g resulting from boundary conditions like contact. The nodal displacements are represented by $\boldsymbol{d} \in \mathbb{R}^n$, with the corresponding velocities $\dot{\boldsymbol{d}} \in \mathbb{R}^n$ and accelerations $\ddot{\boldsymbol{d}} \in \mathbb{R}^n$. For a linear mechanical system on the contrary, the equation simplifies to

$$\boldsymbol{M}\ddot{\boldsymbol{d}}(t)+\boldsymbol{C}\dot{\boldsymbol{d}}(t)+\boldsymbol{K}\boldsymbol{d}(t)=\boldsymbol{\zeta}_{\text{ext}}(t), \tag{2.5}$$

see [Hughes00]. Here, $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is the linear mass matrix, $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ the damping matrix, and $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ the stiffness matrix. The FEM enjoys huge popularity due to its versatility, i.e. the wide range of possible applications including structural analysis, heat transfer, or fluid dynamics, and its large generality [Bathe14].

2.1.1.1 Application-specific Finite Element Models Implementation

The Finite Element (FE) models used throughout this thesis are implemented in the commercial software solution Ansys LS-Dyna [HallquistEtAl06, LSTC21]. It is a widely used explicit simulation program that is employed to simulate highly nonlinear physical processes, with applications that encompass drop tests, crashworthiness investigations, impact and penetration, occupant safety, and biomedicine.

The essential steps involved in FE modeling are (i) the problem definition, (ii) preprocessing, (iii) solving, and (iv) postprocessing. In the first step (i), the geometry of the system and its corresponding material properties must be analyzed while a clear definition and parameterization of the scenario in which the model is tested is developed. For example, if a new car concept is to be studied in the context of crash safety, its dimensions could be measured or extracted from Computer-Aided Design (CAD) models, while the corresponding material properties are taken from manufacturer material specifications or open-source material databases. Last but not least, the crash scenarios to be considered can be derived from government regulations or car safety evaluation agencies. In the second step (ii), the geometry is imported or created in the FE software and discretized into finite elements, what is known as meshing. The according material properties are assigned to the elements and boundary and initial conditions as well as loads are defined.

Once the FE model is in place, the governing equations can be derived and solved using explicit or implicit integration schemes (iii). In the postprocessing step (iv), the simulation results can be visualized and subjected to an analysis. Furthermore, and this is of paramount importance in the context of surrogate modeling, the quantities of interest can be extracted and exported for subsequent processing. Some of the mentioned FE modeling aspects are visualized in Fig. 2.2. Nowadays, many of these steps are automated by commercial software solutions so that the end user's workload is reduced. For example, they offer the direct import of CAD models, built-in libraries with standard material properties or automated meshing.

While FEM is typically used for large deformations and stresses in a body where no significant rigid body motion is involved, another modeling approach usually deals with the opposite phenomenon: Multi-Body (MB) simulations. It is consequently often concerned with kinematics (the study of geometric aspects of continuum motion such as rotation and translation) and kinetics (the study of forces and moments acting on a free body) of large displacements and rotations but small deformations. In general, a Multi-Body System (MBS) consists of rigid bodies with a fixed geometry that are coupled to each other and/or to the environment and loaded with discretized forces. For further details, please refer to [EberhardSchiehlen06, SchiehlenEberhard14].

2.2 Example Models

2.2.1 A Racing Kart Frontal Collision Simulation

This thesis employs a simplified representative of a crash simulation as an illustrative example. Explicit solving algorithms are the preferable methodology in such a setting, as they can handle highly nonlinear problems without numerous iterations and convergence

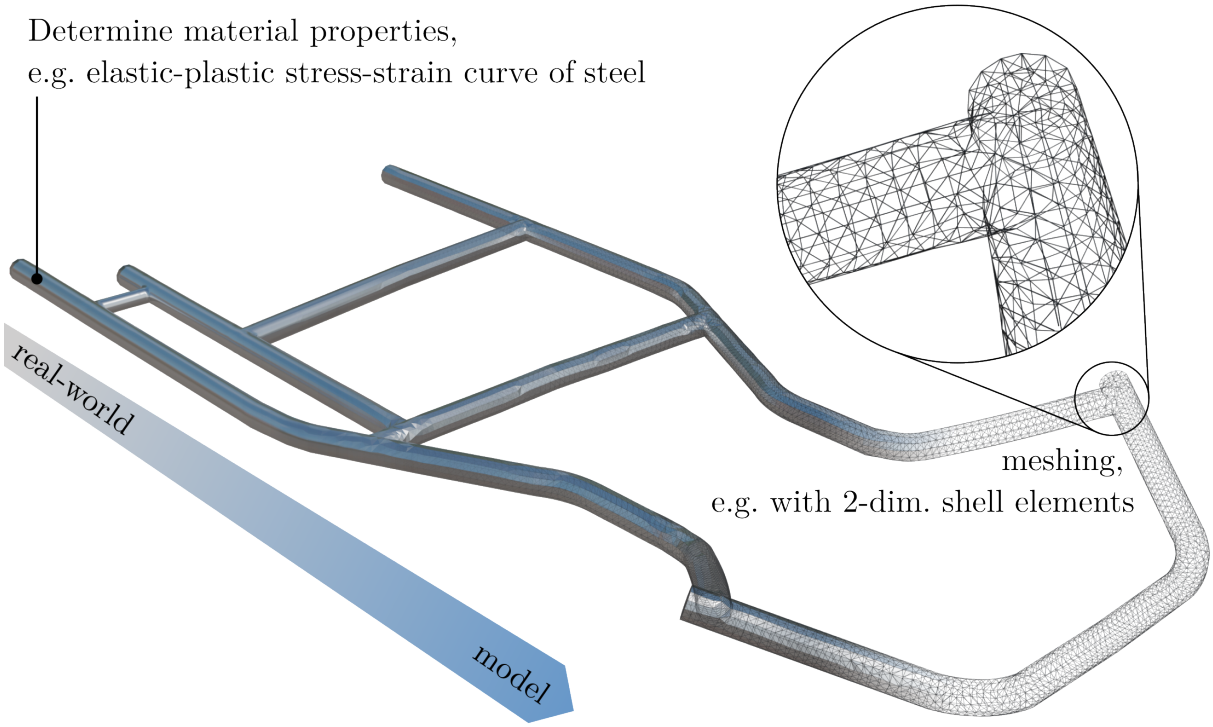


Figure 2.2: Some aspects of FE modeling represented for a racing kart. The considered system is decomposed into its fundamental components, in this case its metallic frame. The geometry and material properties of these components are then measured and exported into a FE software. The geometry is discretized into finite elements, and the material properties are assigned accordingly.

issues, as outlined in the literature [ChangEtAl07]. Crash simulations are a fundamental pillar in the enhancement of passenger safety, optimization of designs, and acceleration of development cycles. They are particularly relevant in integral vehicle safety, which requires scenario-based testing for the examination of a wide range of driving and collision scenarios such as those defined by NCAP [EuroNCAP23]. As a consequence, frontal, side, or gazing collisions must be covered in order to gain a comprehensive understanding of the safety of vehicles. Simulation models are invaluable in this regard, as they facilitate evaluations across a spectrum of scenario parameters without the necessity for expensive physical prototypes.

However, a full-scale safety evaluation of vehicle crash simulations is not the subject of this thesis, and the model used is intended only as an illustrative example with relevance. Accordingly, we only consider a frontal crash, which is one of the most frequently occurring collisions. Additionally, a simplified vehicle model is utilized that is not comparable to an industry-relevant crash simulation. Unlike full-size vehicle models, it omits certain components found in more complex systems, such as energy absorption zones, a safety cage for occupants, and the occupants themselves. Nevertheless, it aligns with the prevailing conditions found in common crash simulation settings, namely the use of commercial closed-source software and limited data obtained from explicit simulations.

Scenario Description As mentioned, we consider a frontal collision scenario between a lightweight racing kart shown in Fig. 2.3a, and a rigid wall with respect to simulation parameters $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^3$ and time t. In detail, it is parametrized by the impact speed $\mu_{\mathrm{I}} \in [5, 35]\,\mathrm{m/s}$, impact angle $\mu_{\mathrm{II}} \in [-45, 45]\,^\circ$, and yield stress $\mu_{\mathrm{III}} \in [168, 758]\,\mathrm{MPa}$. The kart is initialized at the same position through all simulations but with varying speed, which it maintains until the impact occurs. The collision wall is rotated along the z-axis to adjust the impact angle, defined as the angle between the normal of the wall and the orientation of the kart. In addition, the yield stress value μ_{III} serves as an adjustment parameter for the material behavior of the kart. It shifts the effective plastic stress-strain curve (which corresponds to the course of a typical steel curve), see Fig. 2.3d.

Each crash simulation has a simulated time of $t_{\mathrm{end}} = 30\,\mathrm{ms}$, with results exported at intervals of $0.3\,\mathrm{ms}$, resulting in $n_t = 101$ samples per simulation. The internal time step during the FE simulation is much smaller and chosen adaptively. In total, $n_{\mathrm{sim}} = 192$ quasi-random parameter combinations are sampled using Halton sequences. Of these high-fidelity simulations, $n_{\mathrm{sim}}^{\mathrm{train}} = 96$ are used to build surrogate models. The simulation results are organized into a snapshot matrix, $\boldsymbol{X} \subseteq \mathbb{R}^{n \times n_{\mathrm{sim}}^{\mathrm{train}} n_t}$, which captures the system states $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ at various times and across different simulations. Two example simulations are illustrated in Fig. 2.3c.

FE Implementation As mentioned earlier, an important step in modeling is the choice of the level of abstraction and the parts of a system to be considered. For a racing kart, the dynamic behavior depends strongly on the structural properties of its metal frame [ShiibaFehrEberhard12]. Accordingly, the remaining bodies, such as its wheels, chassis, rear axle, or rotary joints in the front knuckles are neglected in the modeling process to trace a more tractable model. In addition, to acknowledge the effect of the mass introduced by the driver and engine, point masses with the according weight are introduced to the model. Slight variations of this model have already been used in [FehrHolzwarthEberhard16, KneiflGrunertFehr21, KneiflEtAl24].

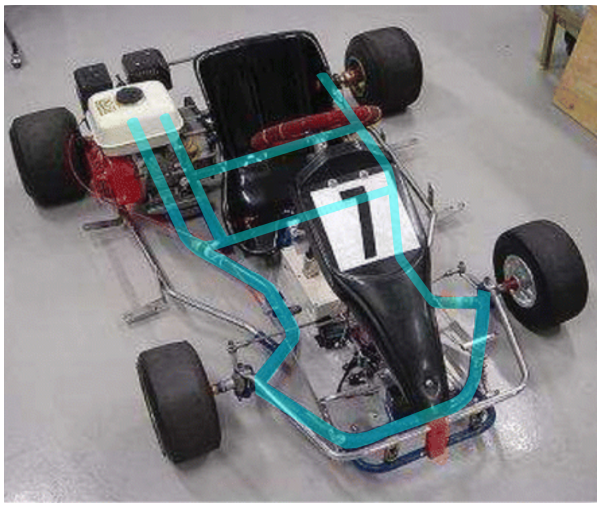
The frame itself is realized as a FE model in the commercial software LS-Dyna. It is constructed from steel pipes, which are modeled as thin-walled tubes using $n_{\text{elem}} = 9360$ shell elements resulting in $n_{\text{node}} = 9314$ nodes, each with three translational as well as rotational degrees of freedom. The point masses of the driver and the engine are connected to the frame at four areas each by lightweight and stiff beam elements, see Fig. 2.3b. The rigid wall and the ground are built from shell elements with 36 nodes each lying on a 5×6 grid on the $y - z$ plane and the $x - y$ plane respectively.

Typical quantities that are of interest in structural dynamical analysis are displacements $\boldsymbol{d} \in \mathbb{R}^{n_{\text{node}} \times 3}$, which describe the deformations along the three spatial directions, and stress measures—such as equivalent stress or von Mises stress—that serve as indicators for material failure or fatigue. Von Mises stress

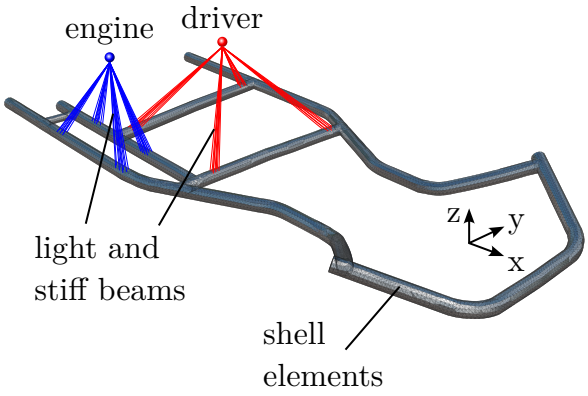
$$\sigma^2 \coloneqq \frac{1}{2}\left((\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2\right) + 3(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2) \tag{2.6}$$

offers the advantage to combine the individual stress components $\sigma_x, \sigma_y, \sigma_z, \sigma_{xy}, \sigma_{yz}$ and σ_{zx} with respect to the coordinate x-, y- and z-axis into a single value. The stress values of all elements are combined in the stress vector $\boldsymbol{\sigma} \in \mathbb{R}^{n_{\text{elem}}}$. As a consequence, displacements and von Mises stress are the quantities considered in this work as target values for surrogate models. Instead of a direct approximation of stress values through data-driven surrogate models as investigated in a previous study for a continuum-mechanical musculoskeletal system [KneiflEtAl23], they can also be derived from approximated displacements using standard FEM tools as is done in [FrescaEtAl22]. An overview of the data dimensions of the model is given in Table 2.1.

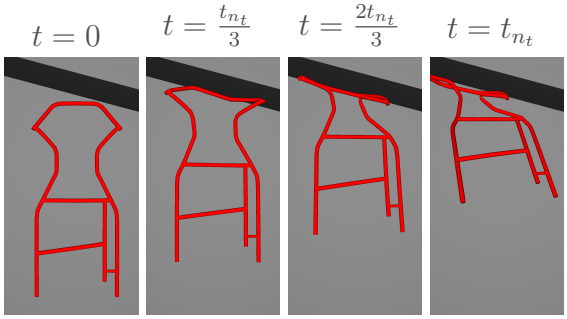
Model challenges To provide the reader with an intuition for the specific challenges and complexity of the presented kart simulation model in the context of surrogate modeling and Model Order Reduction (MOR), it is useful to place it in perspective relative to commonly used benchmark examples. Often used benchmarks in the context of data-driven Reduced Order Models (ROMs) can be found in fluid dynamics and include the Navier-Stokes equation [HesthavenUbbiali18, DalSantoDeparisPegolotti20, FrescaManzoni22] or fluid flow around a cylinder [BruntonProctorKutz16, LuschKutzBrunton18,



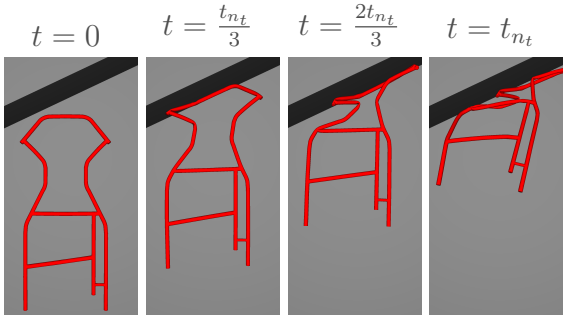
(a) Considered racing kart as presented in [ShiibaFehrEberhard12] with visually highlighted frame.



(b) Finite element model implementation of the frame with the positions of the point masses of the driver and engine.

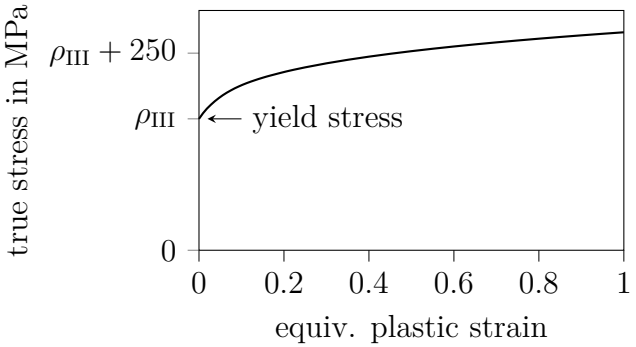


$\rho_{\mathrm{I}} = 27.9\,\frac{\mathrm{m}}{\mathrm{s}}$, $\rho_{\mathrm{II}} = 14.9°$, $\rho_{\mathrm{III}} = 489\,\mathrm{MPa}$



$\rho_{\mathrm{I}} = 34.0\,\frac{\mathrm{m}}{\mathrm{s}}$, $\rho_{\mathrm{II}} = -25.3°$, $\rho_{\mathrm{III}} = 431\,\mathrm{MPa}$

(c) Two example crash simulations of the kart frame. From each simulation four snapshots at different points in time are used for the visualization and the corresponding parameters are stated below.



(d) Stress-strain curve defining the material properties of the kart.

Figure 2.3: Image of the considered racing kart (a) and the FE implementation of its frame (b) along with two example simulations of the modeled frame (c) and the stress-strain curve (d) of the simulated material.

Table 2.1: Hyperparameters for the kart and occupant numerical examples.

Hyperparam. Category	**Kart**	**Occupant**
Dimensions	$n = 27942$ (displacements), $n = 9314$ (stress)	$n = 76233$
Data	$n_{\text{sim}} = 192$, $n_t = 102$, $t_{\text{end}} = 0.3\,\text{ms}$	$n_{\text{sim}} = 117$, $n_t = 80$, $t_{\text{end}} = 1.975\,\text{s}$

ContiEtAl23a, OttoMacchioRowley23]. In the context of mechanics, mass-spring damper chains [MorandinNicodemusUnger23, RettbergEtAl24] are commonly used. Moreover, many investigations deal with stable dynamics on an attracting manifold based on single mode excitations with harmonic inputs [ContiEtAl23a, ContiEtAl24]. Other benchmark models include the 1D or 2D Burger's equation [PeherstorferWillcox16, LeeCarlberg20] or diffusion models on rectangular or cylindrical geometrical domains [BruntonProctorKutz16, DalSantoDeparisPegolotti20, FrescaManzoni22, ContiEtAl24].

Clearly, all of the listed models represent sophisticated dynamical systems and posses difficult challenges. Thus, they are extensively used in various fields, including education, science, and industry. However, for the sake of conducting research we often have to simplify the traditional examples. Hence, we limit them to 2D problems or simple geometries, consider low-dimensional parameter dependencies or focus on specific regimes. Moreover, we simplify physics such as incompressibility and operate in weakly nonlinear regimes, assume linear elastic forces or linear damping but do not consider multi-physics interaction or multi-scale phenomena, and have idealized boundary conditions on simple domains.

Needless to say, the kart model itself also is built using strong simplifications and assumptions, limitations in the scenario descriptions, and a low-dimensional parameter space. Nevertheless, the kart model has distinctive features and Table 2.2 sheds light on them comparing them to traditional models. All in all, the kart frame represents a sophisticated nonlinear example with complex irregular 3D geometry and transient dynamics that is more application-oriented than many standard models used in data-driven surrogate modeling. It serves as a bridge between traditional academic examples and full-scale industrial models.

2.2.2 Active Occupant Model

Besides the main model used throughout this thesis, other models are introduced to address more specialized topics. One of them is an elastic MB model placed in critical pre-crash

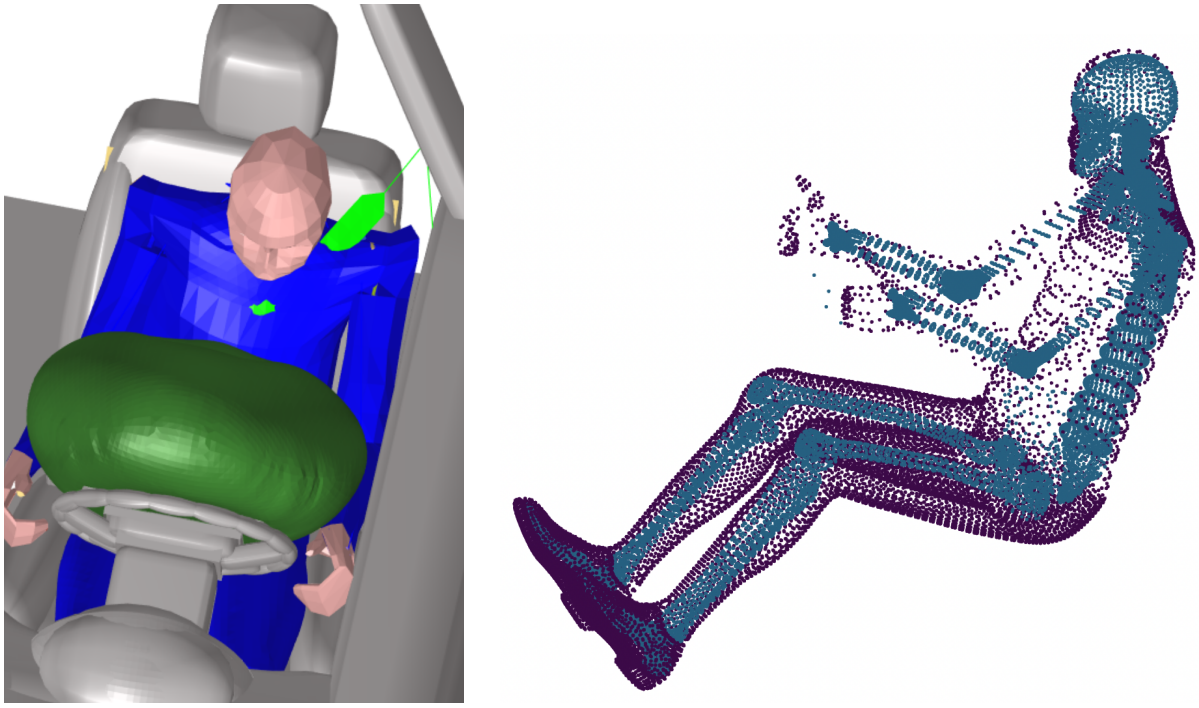
Table 2.2: Comparison of the kart simulation model with frequently used examples in surrogate modeling and model order reduction.

Feature	Traditional Examples	Kart Simulation Model
Geometry	Often simple (e.g. rectangular, cylindrical, or 2D domains).	Complex, 3D geometry with irregular discretization.
Physics	Often simplified (e.g. incompressibility in fluid dynamics or linear elastic forces in mechanical models).	Multi-scale physics using FEM.
Dynamics	Typically stable dynamics (e.g. single-mode excitations, weakly nonlinear regimes).	Transient dynamics with complex nonlinearities (e.g. contact, plastic deformation).
Parameter Dependencies	Often limited to one-dimensional dependencies (e.g. Reynolds number).	Multi-dimensional dependencies affecting material properties, scenario settings, and initial conditions.
Nonlinearities	Often weak or no nonlinearities (e.g. linear damping, linear elastic forces).	Strong nonlinearities due to contact and material behavior.
Computational Complexity	High for full-scale models; low for benchmarks.	Computationally accessible while retaining practical relevance.

scenarios to provide insights about occupant behavior. It is shown in Fig. 2.4a. In those pre-crash scenarios, the driver (occupant) of a vehicle experiences lateral or longitudinal displacements through the vehicle's accelerations. The resulting forces lead to postures differing the standard seating position. A change in seating position can have a significant impact on the resulting injury severity [ReedSchneiderBurney92, BassEtAl99, Hay22].

The model itself originates from Simcenter MADYMO [SISS20], a software package that has been developed with a focus on simulating human safety in transport. It incorporates both, MB and FE solvers. The simulation model includes a vehicle interior with control panel, and a reactive human body model positioned in the driver's seat, with hands on the steering wheel and a fastened seatbelt, as described by [MeijerEtAl13]. The skeleton is modeled with rigid bodies, while the skin is composed of FE shells to obtain higher accuracy in contacts with the surrounding interior. The model's active reaction to applied accelerations is designed to return to the initial seating position. This is achieved through 1D Hill-type muscles with activation controllers and torque actuators in the spine.

Scenario description For a given driving scenario including maneuvers such as breaking, acceleration and steering, the accelerations $\boldsymbol{\mu}(t) \in \mathbb{R}^3$ of the vehicle's center of gravity are applied to the model. For the considered scenarios, the translational accelerations in the x- and y-directions, as well as the rotational acceleration about the z-axis, lie within the range $[-5.31\,\mathrm{m/s^2}, 4.48\,\mathrm{m/s^2}] \times [-6.58\,\mathrm{m/s^2}, 5.69\,\mathrm{m/s^2}] \times [-4.38\,\mathrm{rad/s^2}, 1.77\,\mathrm{rad/s^2}]$ with standard deviations of $1.68\,\mathrm{m/s^2}$, $1.67\,\mathrm{m/s^2}$, and $0.24\,\mathrm{rad/s^2}$. The occupant's movement is captured by measuring nodal displacements during the simulation. By excluding static vehicle interior objects, only the relevant $n_{\mathrm{node}} = 25411$ nodes representing the human body model's displacement are considered, see Fig. 2.4b. Each node has three Degree of Freedoms (DOFs) resulting in 76233 DOFs in total. The total number of simulations is $n_{\mathrm{sim}} = 99$. Similar to the kart example, the relevant data dimensions and hyperparameters are summarized at a glance in Table 2.1.



(a) Active human body model of an occupant placed inside a generic vehicle interior.

(b) Exported nodes of the human occupant model. The skeleton that is modeled as rigid bodies is colored in blue, while the skin modeled using the FE method is colored in purple.

Figure 2.4: Occupant model (a) inside a vehicle as well as a visualization of the nodal information (b) used for surrogate modeling purposes.

2.3 Surrogate Modeling Methods

The reliable and physics-based HF models are prohibitively expensive to evaluate. As a consequence, we seek computationally efficient, predictive but accurate surrogate models. Throughout this thesis, most of the time we consider HF models in parameterized settings, where we do not have access to the underlying governing equation. Moreover, we pursue data-driven surrogate modeling and consequently rely on an online-offline split. During the expensive offline phase, data is obtained from the HF model. This data serves as basis for the subsequent surrogate modeling process. Once a surrogate is found, it can be evaluated cheaply during the online phase.

The starting point for the creation of surrogate models is a *black-box* HF model

$$\boldsymbol{x}(t, \boldsymbol{\mu}) = \boldsymbol{F}(t, \boldsymbol{\mu}, \boldsymbol{x}_0), \tag{2.7}$$

where the operator $\boldsymbol{F}$ represents a solution to (2.3) and is known as *flow* map. It maps the simulation parameters $\boldsymbol{\mu}$ to the solution at time $t \geqslant t_0$, $t \in \mathbb{T}$ for a given initial condition $\boldsymbol{x}_0 \in \mathbb{R}^n$. In this context, $\mathbb{T} = \{t_i \in \mathcal{T}\}_{i=0}^{n_t-1}$ describes the considered set of discrete time points within the time interval $\mathcal{T}$. Please note that we write $\boldsymbol{x}(t, \boldsymbol{\mu})$, omitting the dependence on the initial condition $\boldsymbol{x}_0$ for notational simplicity. In scenarios where the initial condition is fixed and known, the function indeed depends only on t and $\boldsymbol{\mu}$. In more general cases, the initial condition can be incorporated into the parameter $\boldsymbol{\mu}$, making the shorthand notation applicable without loss of generality. Nevertheless, we state it explicitly at certain points to emphasize its potential influence on the system's behavior.

As part of the data collection, the HF model (2.7) is queried for a finite set of parameters $\mathbb{P} = \{\boldsymbol{\mu}_i \in \mathcal{P}\}_{i=1}^{n_{\text{sim}}}$ based on the scenario parametrization. The created $n_s = n_t n_{\text{sim}}$ high-fidelity simulation samples, also called snapshots, are assembled in the *snapshot matrix*

$$\boldsymbol{X} = \Big[\boldsymbol{x}(t_0, \boldsymbol{\mu}_1), \dots \boldsymbol{x}(t_{n_t-1}, \boldsymbol{\mu}_1), \dots, \boldsymbol{x}(t_0, \boldsymbol{\mu}_{n_{\text{sim}}}), \dots, \boldsymbol{x}(t_{n_t-1}, \boldsymbol{\mu}_{n_{\text{sim}}})\Big] \in \mathbb{R}^{n \times n_t n_{\text{sim}}}, \tag{2.8}$$

which forms the foundation for the surrogate modeling process. As long as there are enough samples in the resulting subspace $\mathcal{S}_{\mathbb{P}} = \text{span}\{\boldsymbol{X}\}$, it is supposed that this subspace approximates the solution manifold

$$\mathcal{S}_{\mathcal{P}} = \{\boldsymbol{x}(t, \boldsymbol{\mu}) \mid t \in \mathcal{T}, \boldsymbol{\mu} \in \mathcal{P}\}. \tag{2.9}$$

This, in turn, provides grounds for the assumption that a surrogate trained on $\mathcal{S}_{\mathbb{P}}$ can also be used for $\mathcal{S}_{\mathcal{P}}$.

In addition to the data used for the surrogate model creation, often data for validation and testing is produced as well. Similarly to (2.8), the results from $n_{\text{sim}}^{\text{val}}$ simulations based on a finite set of parameters are used to form the validation data

$$\boldsymbol{X}^{\text{val}} \in \mathbb{R}^{n \times n_s^{\text{val}}}, \text{ with } n_s^{\text{val}} = n_t n_{\text{sim}}^{\text{val}} \tag{2.10}$$

and the results from $n_{\text{sim}}^{\text{test}}$ simulations are used to form the test data

$$\boldsymbol{X}^{\text{test}} \in \mathbb{R}^{n \times n_s^{\text{test}}}, \text{ with } n_s^{\text{test}} = n_t n_{\text{sim}}^{\text{test}} \tag{2.11}$$

While the validation data is used for hyperparameter tuning during the surrogate modeling process, the test data exclusively serves the purpose of assessing the final results on previously unseen data.

Problem Formulation In practice, we seek to identify a surrogate model $\tilde{\boldsymbol{F}}(t, \boldsymbol{\mu}, \boldsymbol{x}_0)$ from a chosen family of available architectures $\mathcal{F}$ that closely approximates the reference solution (2.7) while achieving computational efficiency given the data $\boldsymbol{X}$. This inherently entails a trade-off between accuracy and efficiency, a classic challenge in surrogate modeling. Hence, we can reformulate our goal as an optimization problem of the form

$$\begin{aligned} \min_{\tilde{\boldsymbol{F}} \in \mathcal{F}} \quad & \frac{1}{n_t n_{\text{sim}}} \sum_{t \in \mathbb{T}} \sum_{\boldsymbol{\mu} \in \mathbb{P}} L(\boldsymbol{x}(t, \boldsymbol{\mu}), \tilde{\boldsymbol{x}}(t, \boldsymbol{\mu})) \\ \text{s.t.} \quad & \boldsymbol{x}(t, \boldsymbol{\mu}) = \boldsymbol{F}(t, \boldsymbol{\mu}, \boldsymbol{x}_0) \\ & \tilde{\boldsymbol{x}}(t, \boldsymbol{\mu}) = \tilde{\boldsymbol{F}}(t, \boldsymbol{\mu}, \boldsymbol{x}_0) \\ & \boldsymbol{x}_0 = \boldsymbol{x}(t_0) \\ & \mathcal{J}(\tilde{\boldsymbol{F}}(t, \boldsymbol{\mu}, \boldsymbol{x}_0)) \ll \mathcal{J}(\boldsymbol{F}(t, \boldsymbol{\mu}, \boldsymbol{x}_0)), \end{aligned} \tag{2.12}$$

where L is an objective function measuring the error between the reference $\boldsymbol{x}$ and approximated solutions $\tilde{\boldsymbol{x}}$, while $\mathcal{J}$ is some measure of the computational costs. It should be noted that the mean of the objective function in (2.12) is merely an illustrative example; depending on the specific problem at hand, an alternative measure may be more appropriate. A bouquet of surrogate families and strategies exist for addressing problem (2.12), prompting the question of which is the most appropriate for a given scenario.

2.4 Intrusive Methods

Intrusive methods in MOR refer to techniques that involve modifications to the original model or its governing equation. They are often a popular choice due to their consistency with physical laws, high accuracy, and the well-developed theoretical foundation, see e.g. [Antoulas05, QuarteroniManzoniNegri16, Benner17]. A more comprehensive outline with details about specific algorithms follows in Section 3.2. Many of the intrusive MOR techniques rely on projection and project the high-dimensional model onto a low-dimensional subspace. For linear systems like (2.5) this results in constant reduced operators. Hence, they can be precomputed and all computations required to solve the reduced system are performed in the low-dimensional subspace making them very efficient.

Considering nonlinear systems like (2.4), on the contrary, leads to complications. In this case, the operators are not constant and thus still need to be evaluated in the full high-dimensional space which severely limits any efficiency gains. This resulted in the development of hyperreduction techniques such as the discrete empiric interpolation method [ChaturantabutSorensen10, PeherstorferEtAl14]. Hyperreduction techniques entail the evaluation of nonlinear functions at a limited set of representative points using derived basis functions instead of full evaluations. To illustrate, only the results for a subset of spatial points that capture the essential dynamics are computed; all other points are interpolated from their results. A more comprehensive list of intrusive methods follows in Section 3.2 as well.

However, the fundamental aspect of intrusive methods is also one of their primary limitations: They require knowledge about the original system description to derive low-dimensional surrogate models. On the one side, this enables consistency with the original model, the possibility to maintain system-theoretic properties, or to deploy error estimators [GreplPatera05, GrunertFehrHaasdonk20, FengEtAl23]. On the other side, the governing equations may not always be fully known or not accessible at all, limiting the usability of such methods. Furthermore, direct manipulation of the original simulation code is frequently required, which is tantamount to additional effort.

2.5 Non-intrusive Methods/Data-driven Modeling

Non-intrusive methods in MOR are techniques that do not require the original model's governing equations or simulation code. Instead, they are concerned with the generation of ROMs solely based on the simulation results of the according high-fidelity model. Consequently, they are also referred to as data-driven methods. Their purpose is to either derive mathematical expressions based on the available observations, as done in Sparse Identification of Nonlinear Dynamics (SINDy) [BruntonProctorKutz16], Dynamic Mode Decomposition (DMD) [Schmid10, Kutz16, Schmid22], and Operator Inference (OI) [PeherstorferWillcox16, KramerPeherstorferWillcox24], or to directly approximate the model's solution, e.g. using black-box Machine Learning (ML) models. For a more comprehensive overview on data-driven modeling refer to [BruntonKutz22] and for a comparison of non-intrusive MOR for finite element models to [KaramoozMahdiabadiEtAl21].

2.5.1 Machine Learning

ML is a branch of Artificial Intelligence (AI) focused on developing algorithms that enable computers to learn from and make predictions or decisions based on data, without being explicitly programmed [HastieTibshiraniFriedman17, Bishop19]. This enables broad applications from classification, to regression tasks, reinforcement learning, or generative

modeling among others. In this thesis, we are mainly interested in predicting continuous quantities based on inputs and consequently focus on regression methods. Of the numerous available ML methods, Neural Networks (NNs) and Gaussian Processes (GPs) are of special importance and are introduced in the following. For other popular techniques such as linear regression [MontgomeryPeckVining13], decision trees [Loh11, BreimanEtAl17], or support vector machines [Steinwart08], the interested reader is directed to the corresponding references.

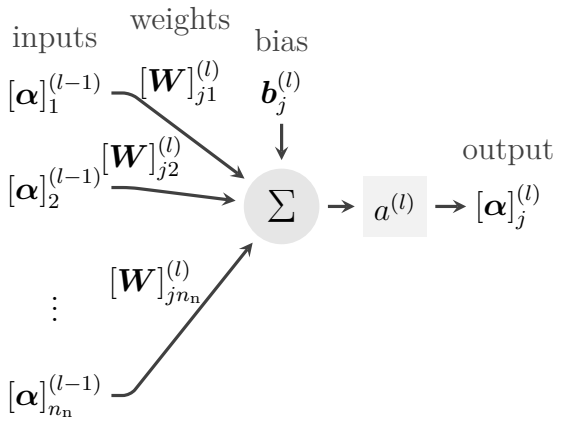
2.5.2 Neural Networks

NNs are the foundation of Deep Learning (DL) [Nielsen18, Aggarwal23] and one of the most popular families of data-driven methods due to their considerable flexibility, which is enabled by a multitude of architectural variations, scalability through parallel processing on GPUs, and expressiveness emphasized by the universal approximation theorems [HornikStinchcombeWhite89]. The fundamental methodology underlying their regression approach entails the linear combination of input variables to generate features, processed by nonlinear functions. The coefficients defining the contributions of the individual features can be modified during the learning process in order to better align with the desired data.

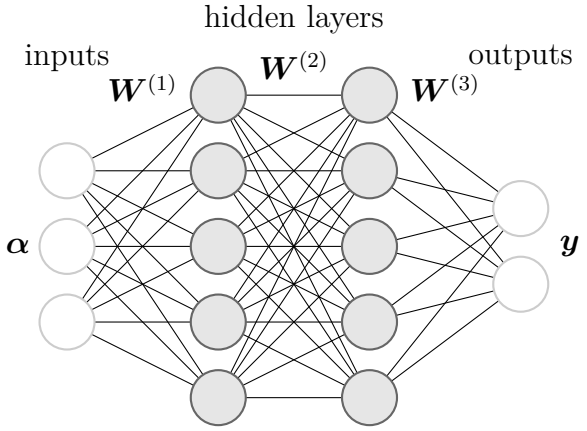
The structure of a NN is composed of successive layers, with any desired number of neurons. Neurons receive signals from other neurons and process these signals, i.e. they are mathematical models that process inputs, apply a transformation, and produce outputs, see Fig. 2.5a. The number of neurons and layers as well as the connections of the layers, and thus the complexity of the network, can be varied and adopted as required. A distinction is made between input, hidden and output layers. The former represents the input data, while the latter represents the predicted output data. The layers in between are not directly observed and are therefore called hidden.

Nowadays, there are a variety of network architectures, for example Recurrent Neural Networks (RNNs) with neurons that receive feedback. Commonly used representatives are Long Short-Term Memorys (LSTMs) [HochreiterSchmidhuber97] and its recent extension xLSTMs [BeckEtAl24]. Moreover, Convolutional Neural Network (CNN) often find application in image recognition, while attention-based architectures [VaswaniEtAl17] led to the recent rise of large language models. Recently, Kolmogorov networks [LiuEtAl24] have attracted a great deal of attention with an approach that replaces learnable weights with learnable activation functions and thus alters the basic structure of NNs.

In order to explain the basic concepts of neural networks, we consider one of its most common and simple representatives: a Multi Layer Perceptron (MLP), i.e a fully connected feedforward NN. An example of such a network is depicted in Fig. 2.5b. It is called fully connected, because each neuron of one layer is connected to all neurons of the following one.



(a) An artificial neuron of a neural network. Each input $[\boldsymbol{\alpha}]_i^{(l-1)}$, $i = 1, \ldots, n_\mathrm{n}$ of the j-th neuron of the l-th layer is weighted by a corresponding weight $[\boldsymbol{W}]_{ji}^{(l)}$ and linearly combined along with the bias $[\boldsymbol{b}]_j^{(l)}$. This sum is passed through a (nonlinear) activation function resulting in the output of the neuron $[\boldsymbol{\alpha}]_i^{(l)} = a^{(l)}\left(\sum_{i=1}^{n_\mathrm{n}} [\boldsymbol{W}]_{ji}^{(l)} [\boldsymbol{\alpha}]_j^{(l-1)} + [\boldsymbol{b}]_j^{(l)}\right)$. Please note that we denote the i-th entry of a vector as $[\square]_i$ and the entry at position ij of a matrix with $[\square]_{ij}$.



(b) Example illustration of a fully connected feedforward multi-layer neural network with three inputs and two outputs. For each neuron in the two hidden layers, a nonlinear activation function is applied to the linear combination of its corresponding inputs. The information is then propagated forward through the network from the input layer to the output layer.

Figure 2.5: Visualization of the functionality a single neuron (a) and of a neural network (b).

In general, a NN $\boldsymbol{\Psi}$ tries to approximate a relationship between an input variable $\boldsymbol{\alpha} \in \mathbb{R}^{n_{\boldsymbol{\alpha}}}$ and the corresponding output variable $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$ based on an available dataset

$$\mathcal{D} \coloneqq \{(\boldsymbol{\alpha}_i, \boldsymbol{y}_i)\}_{i=1}^{n_s}. \tag{2.13}$$

by adapting its weights $\boldsymbol{W}$ so that $\boldsymbol{y} \approx \tilde{\boldsymbol{y}} = \boldsymbol{\Psi}(\boldsymbol{\alpha}; \boldsymbol{W})$. Throughout this thesis, subscripts of the form $\square_i$, where i is a number (excluding 0, which is reserved for initial values) or a lowercase Latin letter, denote the i-th instance of a quantity—e.g. $\boldsymbol{\alpha}_i$ and $\boldsymbol{y}_i$, respectively. In cases where such indexing would conflict with semantic distinctions we instead use Roman numerals (e.g. $\square_\mathrm{I}$, $\square_\mathrm{II}$) to indicate structurally different quantities.

In detail, a NN computes its prediction as a series of successive transformations

$$\tilde{\boldsymbol{y}} = \boldsymbol{\Psi}(\boldsymbol{\alpha}; \boldsymbol{W}) = \boldsymbol{a}^{(n_\mathrm{l})}(\cdot; \boldsymbol{W}^{(n_\mathrm{l})}) \circ \ldots \circ \boldsymbol{a}^{(1)}(\boldsymbol{\alpha}; \boldsymbol{W}^{(1)}) \tag{2.14}$$

of the input variables [HastieTibshiraniFriedman17] where n_l corresponds to the number of layers and the superscripts like $\square^{(l)}$ indicate to which layer each term belongs. The overall weights are composed of $\boldsymbol{W} = \{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\}_{l=1}^{n_\mathrm{l}}$. The transformation in the l-th layer

$$\boldsymbol{\alpha}^{(l)} = \boldsymbol{a}^{(l)}(\boldsymbol{W}^{(l)} \boldsymbol{\alpha}^{(l-1)} + \boldsymbol{b}^{(l)}) \tag{2.15}$$

is constructed out of a (nonlinear) activation function $\boldsymbol{a}$ that receives the linear combination of the outputs of the neurons of the previous layers $\boldsymbol{\alpha}^{(l-1)}$ as input. The linear combination is determined by the weights $\boldsymbol{W}^{(l)}$ and the bias $\boldsymbol{b}^{(l)}$. This information process, i.e the calculation of the output from the input values, is known as forward propagation, whereas backpropagation is used in the conduction of training.

Training During training, the estimated outputs $\tilde{\boldsymbol{y}} = \boldsymbol{\Psi}(\boldsymbol{\alpha}; \boldsymbol{W})$ are predicted in the forward pass with fixed parameters $\boldsymbol{W}$, whereupon the errors are propagated back to adjust the weights with respect to a certain loss function like the Mean Squared Error (MSE)

$$L(\boldsymbol{W}) \coloneqq \frac{1}{n_s} \sum_{i=1}^{n_s} (\boldsymbol{y}_i - \tilde{\boldsymbol{y}}_i)^2. \tag{2.16}$$

One straightforward method for minimizing the loss function is gradient descent optimization, in which the weights are updated in the direction of their negative gradient in every iteration. The i-th iteration results in

$$\boldsymbol{W}^i = \boldsymbol{W}^{i-1} - \gamma_{\text{lr}}^{i-1} \frac{\partial}{\partial \boldsymbol{W}^{i-1}} L(\boldsymbol{W}^{i-1}), \tag{2.17}$$

where the superscript indicates from which iteration a quantity stems from. The learning rate γ_{lr} can be determined before training but also be adjusted during training by the used optimizing scheme or a learning rate scheduler to account for current circumstances. Due to the special structure of a neural network, it is simple to calculate its gradients by the chain rule of differentiation. In general, the weights are not adjusted after every single prediction; rather, they are optimized only after predictions are computed for a certain number of samples. This set of samples is called batch and can include an amount of one sample up to the complete training data set $\mathcal{D}$. Moreover, the complete data set is processed several times during training. Each of these runs is called an epoch. A common optimization algorithm used is ADAM [KingmaBa14]

It is standard practice to design a neural network with more parameters than strictly necessary to approximate the system. This allows the network to learn complex interrelationships, but can lead to the problem of overfitting. Overfitting describes the behavior of a network that only learns the training data by heart and not the underlying relationships. Many methods exist to avoid the risk of overfitting, see [HastieTibshiraniFriedman17]

2.5.3 Gaussian Processes

Another popular methodology for regression tasks are Gaussian Processes. Their core concept is to narrow the space of potential functions that can explain given data by imposing a prior on the functions first and condition them with available observations second. In

doing so, they obtain an approximated posterior probability distribution. For a detailed explanation, please refer to [WilliamsRasmussen06, SchulzSpeekenbrinkKrause18].

Similar to multivariate Gaussian distributions, which are defined by a mean $\boldsymbol{\mu} \in \mathbb{R}^{n_{\boldsymbol{\alpha}}}$ and a kernel matrix $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{n_{\boldsymbol{\alpha}} \times n_{\boldsymbol{\alpha}}}$ (also referred to as covariance matrix), GPs are defined by a mean function $\acute{\boldsymbol{\mu}}(\boldsymbol{\alpha}) \in \mathbb{R}^{n_{\boldsymbol{\alpha}}}$ and kernel function $\kappa(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j)$ (also referred to as covariance function) for inputs $\boldsymbol{\alpha} \in \mathbb{R}^{n_{\boldsymbol{\alpha}}}$. A function $\boldsymbol{f}$ that is distributed as a GP can consequently be denoted as

$$\boldsymbol{f} \sim \mathrm{GP}(\acute{\boldsymbol{\mu}}, \kappa). \tag{2.18}$$

For a finite number of input samples $\boldsymbol{\alpha}$, the corresponding function values $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{\alpha}) \in \mathbb{R}^{n_{\boldsymbol{y}}}$ then follow a normal distribution $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\mathcal{K}})$ where $\boldsymbol{\mathcal{K}}$ is the Kernel matrix obtained by applying the kernel function element-wise to all inputs so that its entries are defined as

$$[\boldsymbol{\mathcal{K}}]_{ij} = \kappa(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j). \tag{2.19}$$

It should be noted that the selected kernel function implies a distribution over functions and gives a higher probability to those that satisfy certain properties, such as smoothness. The so defined GP can serve as a prior in Variational Inference (VI). Recall that the notation $[\square]_i$ is consistently used throughout this thesis to denote the i-th entry of a vector or, in the case of a matrix, the i-th column. Similarly, $[\square]_{ij}$ refers to the j-th entry of the i-th column. For further details, see the notation section.

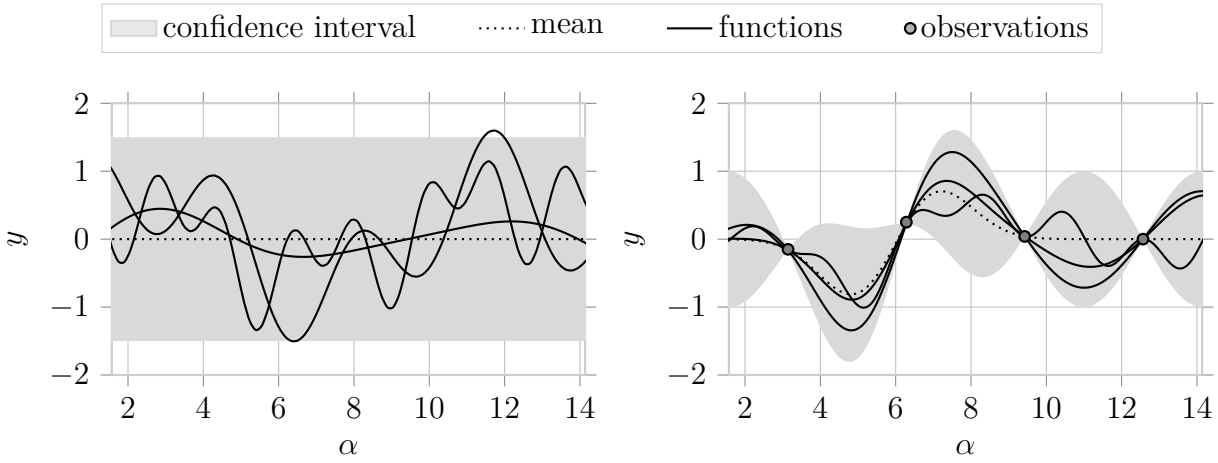
To illustrate the approach how to obtain the corresponding posterior, i.e. how to condition the prior with observations, we follow the explanations from [WilliamsRasmussen06]. Consider a training dataset as in (2.13), where the input samples $\boldsymbol{\alpha}$ have mean values $\boldsymbol{\mu} = \acute{\boldsymbol{\mu}}(\boldsymbol{\alpha}_i)$, $i = 1, \ldots, n_s$ and observed results $\boldsymbol{y}$. Moreover, let $\boldsymbol{\alpha}_*$ be test samples with the means $\boldsymbol{\mu}_*$ for which the output $\boldsymbol{y}_*$ is to be predicted. Suppose $\boldsymbol{y}$ and $\boldsymbol{y}_*$ are jointly Gaussian, then

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\mathcal{K}} & \boldsymbol{\mathcal{K}}_* \\ \boldsymbol{\mathcal{K}}_*^\intercal & \boldsymbol{\mathcal{K}}_{**} \end{bmatrix} \right).$$

In this context, the kernel matrix for the variables in the training input space is represented by $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{n_{\boldsymbol{\alpha}} \times n_{\boldsymbol{\alpha}}}$, while the kernel matrix between the training and test data is represented by $\boldsymbol{\mathcal{K}}_* \in \mathbb{R}^{n_{\boldsymbol{\alpha}} \times n_{\boldsymbol{y}}}$. The kernel matrix among the test variables themselves is represented by $\boldsymbol{\mathcal{K}}_{**} \in \mathbb{R}^{n_{\boldsymbol{y}} \times n_{\boldsymbol{y}}}$. For this specific test set, the posterior can then be calculated as

$$\boldsymbol{y}_* \mid \boldsymbol{\alpha}_*, \boldsymbol{\alpha}, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}_p,\ \boldsymbol{\mathcal{K}}_p)$$

with mean $\boldsymbol{\mu}_p = \boldsymbol{\mu}_* + \boldsymbol{\mathcal{K}}_*^\intercal \boldsymbol{\mathcal{K}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$ and $\boldsymbol{\mathcal{K}}_p = \boldsymbol{\mathcal{K}}_{**} - \boldsymbol{\mathcal{K}}_*^\intercal \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{\mathcal{K}}_*$. An example visualization how a GP prior is conditioned with observations can be seen in Fig. 2.6. For regression, the GP's mean $\boldsymbol{\mu}_p$ serves as predicted output and the corresponding variance can serve as an indicator for the reliability of the prediction. During training, the kernel function's hyperparameters such as length or scale parameters are tuned to match the given data.



(a) GP prior (b) GP posterior

Figure 2.6: In (a), three functions drawn at random from a GP prior with zero mean are shown, whereas subfigure (b) shows three random functions connecting samples drawn from the posterior conditioned with four observations.

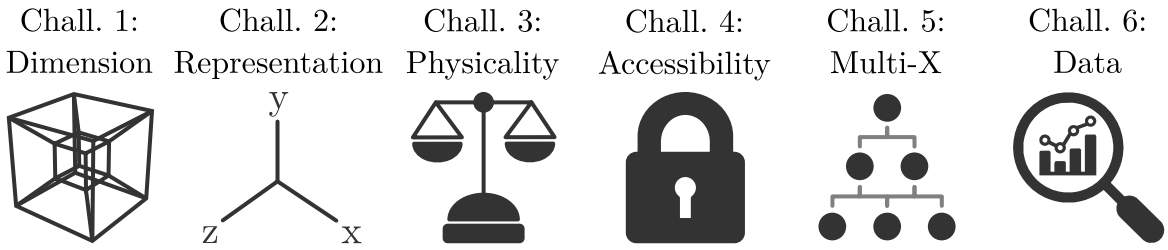
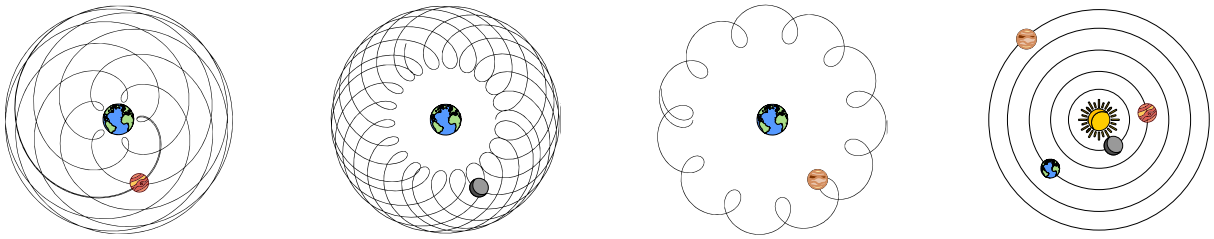


Figure 2.7: Six identified challenges in surrogate modeling of structural dynamical systems.

2.6 Challenges

Despite the plethora of possibilities and algorithms that exist in the present era, the process of surrogate modelling remains a challenging endeavour. There are major challenges that restrict the applicability of existing methodologies, exceed their current capabilities, or impose a considerable computational burden. It is therefore necessary to develop specialized solutions for individual problems. Throughout this thesis, surrogate models are especially developed for structural dynamics problems. Prior to this, we will examine some of the existing challenges in greater detail, particularly those pertinent to this work. A selection of them is depicted in Fig. 2.7.

Challenge 1 (Dimensionality). Very often the present data to derive surrogate models is extremely high-dimensional. This can be the case for simulation data because the HF model relies on a fine numerical discretization. It is not uncommon, for example, that FE models have hundreds of thousands or even million spatial DOFs to accurately represent their physical counterparts. In the case of experimental data, high-dimensionality can



(a) Geocentric Venus orbit (b) Geocentric Mercury orbit (c) Geocentric Jupiter orbit (d) Heliocentric orbits

Figure 2.8: Qualitative illustration of geocentric orbits for a selection of planets (a-c) and heliocentric orbits (d).

occur due to the implemented measurement system. Consider measurements taken on a modern smartphone camera, e.g. a video in full HD resolution. Each frame thus comprises 1920×1080 pixels, with each pixel containing information from three channels corresponding to the respective colors, red, green, and blue. Even in this relatively conservative example, the number of datapoints to be processed for every single frame exceeds six million.
The processing of such high-dimensional data requires significant computational resources, both in terms of processing units and memory. Under such conditions, identifying a system can become an intractable problem due to the difficulty in determining the interactions and relations between that many variables. At the same time, even if a model is found, what is the knowledge we can extract from it when its sheer dimensionality compromises interpretability and generalizability? Fortunately, the manifold hypothesis [FeffermanMitterNarayanan16, GorbanTyukin18] states that a large share of high-dimensional data observed in the real world are in fact embedded in a low-dimensional manifold within this high-dimensional space. This tells us that it is not necessary to operate on the full system state but only on an expressive small number of latent variables. The question that we are left with is how we can find this low-dimensional manifold.

Challenge 2 (Representation). Physical laws are expressed as mathematical relations between state variables and their description is closely tied to the coordinate system in which they are expressed. For a simple mechanical example like a pendulum, the choice of coordinates can already have a significant impact. A description of its dynamics in Cartesian coordinates based on the position of its bob is more complicated than a description in generalized coordinates in the form of its angle (the angle between the vertical and the pendulum rod) and the corresponding velocity.
A comparable situation can be observed in the context of heliocentrism and geocentrism. While the orbits of the planets of our solar system appear complex in a geocentric coordinate system, they can be described as ellipses around the barycenter of our solar system (the center of mass of several point masses) in a heliocentric one. A simplified illustration of the orbits is given in Fig. 2.8. It required the contributions of prominent scientists

such as Kopernikus, Kepler, and Newton to challenge the prevailing geocentric view, which persisted in many ancient and medieval European civilizations, and establish the heliocentric coordinate system.
Another example of the profound impact of coordinate transformations can be found in the context of Einstein's general relativity. Shortly after Einstein presented his field equations in 1915 [Einstein15], Karl Schwarzschild found a solution, designated as the Schwarzschild metric, for a homogeneous mass under a universal cosmological constant of zero in the absence of electric charge and angular momentum [Schwarzschild16]. However, the occurrence of a singularity in his solution at the event horizon of a black hole (the boundary of a black hole where not even light can escape) puzzled mathematicians and physicists. It took almost 50 years until Martin Kruskal [Kruskal60] and György Szekeres [Szekeres] independently revealed that this singularity is not a physical one but a coordinate singularity. Hence, the singularity can be resolved by a transformation of the Schwarzschild metric into new coordinates, which are now known as Kruskal-Szekeres coordinates.
In the context of PDEs, it is possible for some nonlinear differential equation to be linearized by a clever choice of coordinates. Take for example the Burger's equation

$$\boldsymbol{x}_t + \boldsymbol{x}\boldsymbol{x}_{\boldsymbol{\chi}} - \mu \boldsymbol{x}_{\boldsymbol{\chi}\boldsymbol{\chi}} = 0, \tag{2.20}$$

that can be transformed into the linear heat equation (2.2) by substituting $\boldsymbol{x} = -2\mu \frac{\hat{\boldsymbol{x}}_{\boldsymbol{\chi}}}{\hat{\boldsymbol{x}}}$.
All these examples underline how crucial it is to find a suitable representation for a given system by choosing the right coordinate system and emphasize the significant benefits this can offer. Accordingly, the coordinates used to describe a surrogate model must not only be low-dimensional but also represent the dynamics. The first two challenges can thus lead to contradictory objectives as the low-dimensional coordinates are not necessarily suited to describe the dynamics in meaningful manner. As a consequence, both dimensionality and dynamics must be acknowledged to find a fitting description for a system.

Challenge 3 (Physicality). A further challenge inherent to data-driven modelling is the absence of physicality. This naturally constrains the reliability, trustworthiness and interpretability of the models, rendering them unsuitable for deployment in safety-critical applications. Furthermore, models based solely on data are prone to overfitting and exhibit limited extrapolation capabilities outside the training regime, e.g. for long-term predictions. In [KarniadakisEtAl21] three ways in which physics can enter a ML model are described: observational bias, inductive bias, and learning bias. Observational bias describes physicality introduced by the data, i.e. in the observations from physically consistent simulations or real-world measurements. Inductive bias describes prior assumptions like mathematical constraints that are encoded in the model structure and consequently are guaranteed to be fulfilled strictly. In contrast, learning bias refers to the process forcing the training of a model to converge towards solutions that adhere the underlying physics, e.g. by the inclusion of specific terms in loss functions. As a consequence, the used data,

the architecture of the surrogate model and the corresponding training objective all need to be carefully chosen to create reliable, accurate and physically consistent predictions when required.

Challenge 4 (Accessibility). As previously stated, the availability of governing equations enables intrusive methods to derive powerful surrogates. Nevertheless, there are numerous physical processes that remain poorly understood, suffer from erroneous oversimplified assumptions, inaccuracies in recorded measured variables, or other deficiencies in the derivation of system properties. Consequently, it is not always feasible to derive governing equations, or they may not be sufficiently descriptive of the underlying process.
Even in instances where this is not the case, models are often simulated using commercial software. The companies responsible for developing these software solutions often have little interest in sharing their explicit knowledge for various reasons. Obviously, one objective is to maintain competitive advantage and safeguard their intellectual property. Moreover, some companies have security concerns as open-sourcing could expose security vulnerabilities and closed-source software enables them to maintain strict quality control. For a more comprehensive perspective on the debate on closed-source and open-source software refer to [Raymond99, WestGallagher06, Feller07]. Unfortunately, these considerations often result in the inaccessibility of the provided software.
Hence, all that remains for the creation of efficient surrogate models is data and limited information about the system itself. The process of surrogate modeling thus becomes more difficult, especially regarding the correct choice of coordinates (Chall. 2) and physical consistency (Chall. 3). How is it possible to derive expressive, extrapolative and physical-aware data-driven surrogates without knowledge about the explicit description of the underlying system?

Challenge 5 (Multi-X). In practice, numerous physical models are constrained to a singular domain or scale, and thus only facilitate the simulation of isolated effects. Conversely, in the context of real-world problems, a multitude of physical phenomena and coupled processes exists across a spectrum of scales and domains, evolving at varying rates. Multi-physics modeling is concerned with coupled simulations to acknowledge for the interactions occurring between domains, while multi-scale modeling deals with the investigation of phenomena at smaller scales and their seamless integration at larger scale and coupled simulations. Covering multiple spatial scales is one of driving forces in (Chall. 1), as the discretization in mesh-based multi-scale simulations like FE simulations must resolve the smallest features resulting in exceedingly large dimensionalities. The surrogates for multi-x phenomena thus must be capable of capturing and coupling different domains in different time as well as spatial scales.

Challenge 6 (Data). The training of many ML models requires big data, particularly in deep learning. For many scientific problems obtaining that much data is not feasible. This is due to the fact that either the measurements themselves are costly and labour-intensive,

or the phenomenon of interest is inherently difficult to capture in a general sense. For simulation data, the process of obtaining HF simulation results is often very time- and power-consuming and hence expensive. Consequently, the amount of available data to work with is usually severely constrained.
Furthermore, the quality of the data may be compromised by uncertainties and errors introduced at various stages of the computational or measurement workflow. Aleatory uncertainties result from randomness or intrinsic variability inherent to a system. This type of uncertainty is typically irreducible because it arises from natural variability or random processes that cannot be controlled. For instance, slight variations in machine performance of a manufacturing process could lead to varying product quality even when all other conditions are constant. Epistemic uncertainties, on the other hand, result from a lack of knowledge of a system, such as unknown factors, incomplete data, or an incomplete understanding of the phenomena and are consequently linked to Chall. 4. For example, miscalibrated sensors could introduce a wrong bias into a modelling process. The field of Uncertainty Quantification (UQ), see e.g [Smith24], addresses these topics as it deals with characterization of uncertainty and the analysis of its propagation from inputs to model outputs. For a review of UQ in DL refer to [AbdarEtAl21]. With such sources of uncertainties and errors that enter at various stages of a computational workflow, surrogate models must be derived in a high-noise low-data regime. The major task that arises from that is to design reliable systems from unreliable sources.

Other challenges The list of challenges obviously is not comprehensive and there exist many more. However, these are outlined as the most significant throughout this work and are consequently elaborated upon. Besides the aforementioned challenges, more information can be found in [LavinEtAl21], where motifs of simulation intelligence are defined along with the respective challenges. In addition, hyperparameter tuning can be an exhaustive and time-consuming process and a trade-off between physicality and efficiency or between accuracy and interpretability must often be made.
However, it is not always the case that interpretability and physicality must be in contradiction with accuracy. The introduction of knowledge and awareness of constraints can also enhance the quality of models. The most fundamental laws of nature are often described by simple, parsimonious expressions that are applicable across a range of scientific fields. This is a principle that has been established by natural science over the past centuries. In light of the above, the derivation of parsimonious models can be beneficial in the context of surrogate modelling, leading to more robust results and potentially better generalization properties, particularly in the absence of infinite data. For a more comprehensive view on parsimony for physics-informed ML refer to [KutzBrunton22, BruntonKutz23]. All of the aforementioned challenges (1-6) are tackled by some of the introduced methods and surrogate models in this thesis. The following chapter is dedicated to the first two: Dimensionality and Coordinates.

Chapter 3

Shrinking the Problem: Identification of Low-dimensional Representations

Schatten tanzen, die nur wir erkenn'n
Flammend rotes Firmament
Seh', wie du deine Hand aufhältst
Ich greif' nach dir, wenn du fällst

Klangkünstler ft. Obernaur, Die Welt Brennt

As mentioned previously in Challs. 1 and 2, the choice of coordinates is crucial for the effective representation of a system. It is therefore necessary to seek descriptions that are simultaneously low-dimensional–in order to guarantee efficiency–, expressive–in order to generalize to a multitude of circumstances and varying dynamical conditions–, as well as comprehensive–in order to foster interpretability. This chapter is specially dedicated to the problem of dimensionality of systems. Therefore, it examines the reasons behind the dimensionality of classical numerical approximation schemes and the methods that circumvent this issue by identifying low-dimensional representations. One field addressing this topic is Model Order Reduction (MOR). As the term itself indicates, this process entails the reduction of the order of models, with the objective of achieving enhanced computational efficiency.

3.1 Spatial Convergence of Numerical Methods

Recall that it is in general not possible to solve Partial Differential Equations (PDEs) equations exactly. Consequently, discretization-based methods are employed to compute approximate solutions by representing the continuous system using a finite set of points

and basis functions. In system analysis, it is crucial that these approximations capture the actual physical phenomena rather than artifacts introduced by the numerical discretization scheme.

However, the simulation results of discretization methods such as the Finite Element (FE) are determined by the resolution of the spatial discretization. Consequently, increased accuracy comes at the cost of increased dimensionality and thus computational effort. This effect is known as spatial convergence describing the rate of change in accuracy for decreasing element sizes [BabuskaSzabo82]. In consequence of the above, techniques of refinement were introduced in the context of the Finite Element Method (FEM). These include h-refinement, which alters the mesh by dividing elements into smaller ones, p-refinement, which increases the polynomial degree of an element, and the combination of both referred to as h-p-refinement [BabuškaGuo92]. In the end, the discretization must be sufficiently fine that alterations in mesh size have neglectable impact on the results.

In this context, singularities as they can occur at sharp corners particularly pose a major challenge. As a result the discretization must be greatly refined around them. One method to mitigate this phenomenon is adaptive mesh refinement, which dynamically refines the meshes only where needed [Plewa05, PerssonPeraire09]. The concept of adaptive mesh refinement is founded upon the premise that local stresses within a given region are not influenced by stresses in other regions, as postulated by the Saint-Venant's Principle [Toupin65]. Furthermore, multi-scale problems present a significant challenge, as they require the simultaneous capture of both macro- and microscale phenomena, as outlined in Chall. 5. In such scenarios, the underlying mesh must be sufficiently refined to resolve the smallest scales. Multi-grid methods, as described in references [McCormick87, TrottenbergOosterleeSchuller00], address this challenge by gradually refining a coarse-grained model in areas of high inaccuracy, thereby achieving the desired level of accuracy.

In light of the aforementioned considerations, it is not surprising that numerous numerical realizations are of a considerable dimension even if the dynamics of the underlying system reside on a low-dimensional manifold, i.e. even if the system has a low-dimensional intrinsic dimension. That means that, rather than stemming from the inherent properties of a given problem, dimensionality frequently emerges as a consequence of the modeling technique itself. To showcase this statement, let us consider the kart example from Section 2.2.1. The numerical model stems from a discretized parameterized PDE with $n_\mu = 3$ parameters. The intrinsic dimensionality of the resulting solution manifold is, at most, equal to the number of parameters $\boldsymbol{\mu}$ plus one (accounting for time t) under the assumption that the dynamical system has a unique trajectory for each parameter instance [LeeCarlberg20]. Hence, the intrinsic dimension of this parameterization equals at most $r = 4$ but the numerical discretization, however, has over 50000 Degree of Freedoms (DOFs).

A series of FE models of the kart are generated, differing only in their discretization, to investigate the impact of dimension on qualitative results. Subsequently, crash simu-

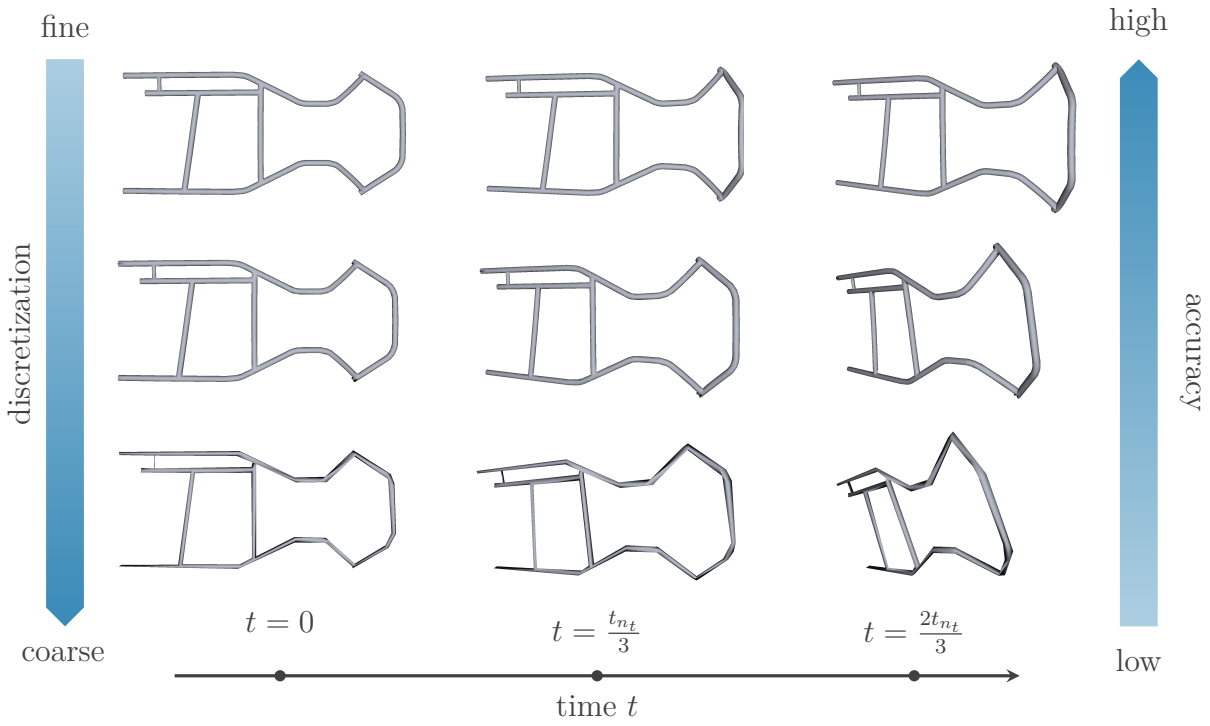


Figure 3.1: Simulations of the FE kart model at varying levels of discretization. The top row depicts the original discretization, comprising over 9,000 nodes. The second row represents a coarse version with 1,169 nodes resulting in a mean node distance error above 13.8 cm, while the last row depicts a very coarse version with 74 nodes resulting in an error above 15.7 cm over all test simulations.

lations of one and the same scenario are conducted for all discretizations. Despite the models' initial conditions being similar, their behavior diverges significantly during the simulations, as illustrated in Fig. 3.1. At the end of the simulations, the discretizations exhibit distinct inclinations, displacements, and deformations, thereby demonstrating qualitatively different dynamic behaviors. This outcome confirms the necessity of a fine resolution using the FEM. It should be noted that the results do not represent a proper convergence study. Furthermore, the coarse models were produced with a downsampling approach [GarlandHeckbert97] that maintains geometrical aspects but does not employ a mesh simplification method that is specifically tailored for FE models. Nevertheless, the results indicate the extent to which conventional methods rely on a high degree of resolution and highlight the inherent trade-off between accuracy and efficiency.

In conclusion, it is not reasonable to expect numerical models to align with the intrinsic dimension of a given problem. To enhance the efficiency of the models used, it is essential to explore alternative methodologies that can effectively reduce the dimensionality.

3.2 Fundamentals of Model Order Reduction

Model Order Reduction is a cornerstone for the efficient simulation and optimization of dynamic models, particularly for those that arise in systems with many degrees of freedom. The primary objective is to simplify those complex, large-scale models while preserving their essential behavior and accuracy. This is often achieved by applying dimensionality reduction.

Addendum 1 (A Brief and Incomplete History of Model Order Reduction)**.** *First approaches following this idea arose in the context of superposition methods in the 19th century [Rayleigh96]. Superposition methods approximate the dynamic response of a system by a superposition of a small number of its eigenmodes determined by generalized degrees of freedom. A subsequent significant development occurred during the 1950s and 1960s. During this period, there were rapid advancements in the fields of linear system theory and control theory, driven by the need for sophisticated technologies in aerospace applications and the race to the moon. Notable developments include the concepts of controllability and observability for state-space models, as proposed by Kalman [Kalman60], which played a pivotal role in the advancement of MOR. Furthermore, these developments had a considerable impact on the following Apollo space missions, as evidenced by [McGeeSchmidt85].*

The limited computing power and memory available in computers of that time, however, meant that large models could not be solved in a single step. One of the earliest approaches to address this problem is Component Mode Synthesis (CMS) [CraigBampton68, Craig00, ParkPark04], introduced by [Hurty65]. In this context, large models are partitioned into smaller substructures, each represented by different ansatz functions (component modes). The individual components are subsequently coupled (synthesized) to approximate the global behavior. In [CraigBampton68], a popular representative of CMS has been proposed in which the ansatz functions are composed of internal eigenmodes of the substructures and static deformation modes with respect to corresponding boundary displacements (correction modes).

In the 1980s, the research was still primarily focused on the input-output behavior of systems, particularly in the context control applications. These efforts led to the development of balanced truncation by [Moore81] for which [PernebоSilverman82] showed that it preserves stability. The fundamental idea is to preserve the most controllable and observable states. In detail, a system is transformed into a balanced realization, in which the states are ordered according to their contribution to the input–output behavior. Then only the most essential states are kept, and the remainder are truncated to obtain the reduced system. For further information, please refer to [GugercinAntoulas04, ReisStykel08].

In the following decade, Krylov subspace methods [Bai02, BeattieGugercin05] were developed to construct Reduced Order Models (ROMs) in such a way that ensures the transfer

function of the ROM to align with that of the original system. For this, the reduction approach focuses on matching the first coefficients (moments) of a so-called moment expansion of the transfer function. Consequently, such methods are also referred to as moment-matching. One example for such an approach was derived in [FeldmannFreund95].

During the 2000s, an increasing emphasis was placed on nonlinear problems with the development of hyperreduction techniques. Approaches to tackle nonlinear systems include the empiric interpolation method [BarraultEtAl04, Gugercin08] and its variant the discrete empiric interpolation method [ChaturantabutSorensen10, PeherstorferEtAl14] that avoid evaluating high-order nonlinear terms by sparsely sampling and evaluating them. In the context of FE models, another popular approach is the energy-conserving sampling and weighting method [FarhatEtAl14, FarhatChapmanAvery15]. This method involves evaluating the nonlinearities only for a small subset of elements.

Although data-driven approaches in MOR have existed since the 1950s in the form of Principal Component Analysis (PCA) and its variants, see [Lorenz56, Sirovich87, BerkoozHolmesLumley93], it was not until the 2010s that their development gained significant momentum. The integration of machine learning, in particular, has been a driving force in this regard, as it enabled new possibilities for the creation of surrogate models from simulation or experimental data. Among the methods employed were neural networks [HesthavenUbbiali18] and Gaussian processes [GuoHesthaven18]. Others inferred operators from data [PeherstorferWillcox16]. It should be noted that this list of methods is not exhaustive and that numerous additional approaches exist. A comparison of MOR methods in structural dynamics can be found in [BesselinkEtAl13].

3.2.1 Dimensionality Reduction

The objective of dimensionality reduction is to derive a low-dimensional yet expressive representation of system states, thereby facilitating the efficient and rapid computation of surrogate models. The objective is therefore to identify reduced coordinates $\boldsymbol{z} \in \mathcal{Z} \subseteq \mathbb{R}^r$, designated as latent states, which are embedded within a so-called *latent space* $\mathcal{Z}$. The latent space is intended to have a considerably lower dimension $r \ll n$ than the original state space $\mathcal{X} \subseteq \mathbb{R}^n$. Despite its lower dimensionality, the latent state must nevertheless be capable of capturing all of the essential characteristics of the original states.

In light of the aforementioned, two coordinate transformations serve as the core of the endeavor to find the latent variable: a *reduction mapping* $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{Z}$ and its reverse, the *reconstruction mapping* $\boldsymbol{\psi} : \mathcal{Z} \to \mathcal{X}$. While the reduction mapping transforms the original state into its latent equivalent $\boldsymbol{z} = \boldsymbol{\phi}(\boldsymbol{x})$, the reconstruction mapping enables the original state to be approximately reconstructed from the low-dimensional approximation. Their function composition $\boldsymbol{\psi} \circ \boldsymbol{\phi}$ yields the reconstructed state $\breve{\boldsymbol{x}} \coloneqq \boldsymbol{\psi} \circ \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\psi}(\boldsymbol{z})$ with $\boldsymbol{x} \approx \breve{\boldsymbol{x}}$. There are a multitude of methodologies for deriving the reduction and

reconstruction mappings in the domain of model order reduction, as evidenced by the literature [Antoulas05, GuoHesthaven19, BennerEtAl21].

In this thesis, we restrict ourselves to data-driven reduction techniques for parameterized systems. Hence, recall the problem formulation outlined in Section 2.3. We obtain data in the form of High-Fidelity (HF) simulations based on a finite number of simulation parameters $\boldsymbol{\mu} \in \mathbb{P}$ within a specified time interval $\mathcal{T}$. The collected data is stored in the snapshot matrix $\boldsymbol{X} \in \mathbb{R}^{n \times n_s}$ where the number of samples n_s corresponds to the sum of samples of all simulations, see (2.8). This data serves as starting point for the dimensionality reduction of the system.

3.2.2 Linear Reduction Techniques

A well-established methodology for identifying reduced coordinates can be found in linear reduced basis methods, as discussed in [QuarteroniManzoniNegri16]. Such methods are based on the mathematical principle of projection. A projection $\boldsymbol{P}$ is defined by two subspaces spanned by the matrices $\boldsymbol{V} \in \mathbb{R}^{n \times r}$ and $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{n \times r}$ respectively. When the projection matrices are bi-orthogonal, i.e. $\boldsymbol{\mathcal{W}}^\intercal \boldsymbol{V} = \boldsymbol{I}$, the projector is defined as $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{\mathcal{W}}^\intercal$. In general, a distinction is made between *Galerkin* projections where $\boldsymbol{V} = \boldsymbol{\mathcal{W}}$, and *Petrov-Galerkin* projections where $\boldsymbol{V} \neq \boldsymbol{\mathcal{W}}$ [Antoulas05]. In this thesis, only the former type of projection is considered.

In the case of the Galerkin projection, the state

$$\boldsymbol{x} \approx \breve{\boldsymbol{x}} \coloneqq \boldsymbol{V}\boldsymbol{V}^\intercal \boldsymbol{x} = \boldsymbol{V}\boldsymbol{z} \tag{3.1}$$

is approximated as a linear combination of a relatively small number of *reduced basis vectors* $\{\boldsymbol{v}_i \in \mathcal{X} \subseteq \mathbb{R}^n\}_{i=1}^r$ stored in the reduction matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r]$. The entries of the reduced state $\{[\boldsymbol{z}]_i\}_{i=1}^r$ serve as coefficients for the corresponding basis vector. In such a case, the reduction mapping is a projection $\boldsymbol{\phi}_{\text{lin}}(\boldsymbol{x}) \coloneqq \boldsymbol{V}^\intercal \boldsymbol{x}$ onto the subspace spanned by $\boldsymbol{V}$ and the reconstruction mapping is the corresponding back projection $\boldsymbol{\psi}_{\text{lin}}(\boldsymbol{z}) \coloneqq \boldsymbol{V}\boldsymbol{z}$.

For a dynamical system described by a set of ODEs as in (2.3), the reduced-order dynamical system is obtained as

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}(t, \boldsymbol{\mu}) = \boldsymbol{V}^\intercal \boldsymbol{f}(t, \boldsymbol{V}\boldsymbol{z}(t, \boldsymbol{\mu}); \boldsymbol{\mu}), \tag{3.2}$$

and evolves in the r-dimensional subspace $\mathcal{Z}$. Of the many existing reduced basis methods, two particular reduced basis methods are discussed in detail in the following.

3.2.3 Principal Component Analysis

Principal Component Analysis attempts to approximate state trajectories of a dynamic system as accurately as possible while projecting them onto an r-dimensional subspace.

It is one of the most popular and at the same time longest existing methods in the field of data-driven reduction techniques. It was pioneered by [Pearson01] and [Hotelling33] in the area of statistics for reducing the dimensionality of data while retaining as much variance as possible. In the context of engineering, it is better known as Proper Orthogonal Decomposition (POD) [Volkwein13]. This term was shaped by the work of Lumley in the 1960s, referring to it as proper orthogonal expansion, see [BerkoozHolmesLumley93], and [Sirovich87]. It has been independently discovered in other domains as well and is therefore also known as Karhunen-Loève or Kosambi-Karhunen-Loève transformation.

The objective of PCA is to identify a set of orthonormal basis vectors that can effectively represent the snapshot matrix $\boldsymbol{X}$. To find the basis vectors, the snapshot matrix is decomposed as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Omega}^\intercal \tag{3.3}$$

using a Singular Value Decomposition (SVD). This yields three matrices: one orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ containing the left-singular vectors as columns, another orthogonal matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n_s\times n_s}$ containing the right-singular vectors as columns, and a rectangular matrix $\boldsymbol{\Sigma} = \operatorname{diag}(\boldsymbol{\varsigma}) \in \mathbb{R}^{n\times n_s}$ with the singular values $\boldsymbol{\varsigma} \in \mathbb{R}^{\min(n,n_s)}$ on its diagonal. Consequently, the left singular vectors can serve as basis vectors, also referred to as principal components, while the matrix product $\boldsymbol{\Sigma}\boldsymbol{\Omega}^\intercal$ yields the corresponding coefficients.

The magnitude of the singular values $\boldsymbol{\varsigma}$ reflects the contribution of the associated basis vector to the overall approximation. Thus, the singular vectors are usually ordered in descending order of their singular values $[\boldsymbol{\varsigma}]_1 \geqslant [\boldsymbol{\varsigma}]_2 \geqslant \ldots \geqslant [\boldsymbol{\varsigma}]_r > [\boldsymbol{\varsigma}]_{r+1} \geqslant \ldots \geqslant [\boldsymbol{\varsigma}]_{\min(n,n_s)}$, where we suppose that the r-th singular value is truly greater than its successor. Hence, if the singular values decrease fast, it is possible to approximate the snapshot matrix with only the r most dominant basis vectors. Moreover, the rank-r matrix approximation $\boldsymbol{X} \approx \tilde{\boldsymbol{X}} = [\boldsymbol{U}]_{1:r}\,[\boldsymbol{\Sigma}]_{1:r\,1:r}\,[\boldsymbol{\Omega}]_{1:r}^\intercal$ obtained from the truncated singular value decomposition with $[\boldsymbol{U}]_{1:r} \in \mathbb{R}^{n\times r}$, $[\boldsymbol{\Sigma}]_{1:r\,1:r} \in \mathbb{R}^{r\times r}$, $[\boldsymbol{\Omega}]_{1:r} \in \mathbb{R}^{n_s\times r}$ is optimal under the Frobenius norm $\left\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\right\|_\mathrm{F}$ following the Eckart–Young–Mirsky theorem originally solved by [Schmidt07]. By exactly using these truncated first r columns of $\boldsymbol{U}$, i.e. the (most important) basis vectors, the reduction matrix for a Galerkin projection

$$\boldsymbol{V}_\mathrm{PCA} \coloneqq [\boldsymbol{U}]_{1:r} \in \mathbb{R}^{n\times r} \tag{3.4}$$

is found. Instead of applying the SVD to the snapshot matrix, an eigenvalue problem can be solved for its Gramian $\boldsymbol{X}^\intercal\boldsymbol{X}$ to obtain the reduced basis vectors as well. The reduction and reconstruction mapping we seek are in the end obtained as $\boldsymbol{\phi}_\mathrm{PCA}(\boldsymbol{x}) \coloneqq \boldsymbol{V}_\mathrm{PCA}^\intercal\boldsymbol{x}$ and $\boldsymbol{\psi}_\mathrm{PCA}(\boldsymbol{z}) \coloneqq \boldsymbol{V}_\mathrm{PCA}\boldsymbol{z}$ respectively.

3.2.4 CUR Decomposition

The CUR Decomposition, as formalized by [MahoneyDrineas09], is a low-rank matrix factorization resulting in a set of three matrices analogous to the SVD. In contrast to the latter, it does not construct the matrices out of abstract bases, but rather explicitly expresses them using a small number of actual columns and rows of the original matrix. This approach offers the advantage of interpretable basis vectors, as they correspond directly to the original data. Consequently, a present sparsity structure is maintained.

In detail, the matrix is decomposed as

$$\boldsymbol{X} \approx \mathbf{C}\mathbf{U}\mathbf{R}, \tag{3.5}$$

where $\mathbf{C} \in \mathbb{R}^{n \times r_\mathbf{C}}$ is composed of a small number of actual columns of $\boldsymbol{X}$ and $\mathbf{R} \in \mathbb{R}^{r_\mathbf{R} \times n_s}$ is composed of a small number of actual rows of $\boldsymbol{X}$. The matrix $\mathbf{U} \in \mathbb{R}^{r_\mathbf{C} \times r_\mathbf{R}}$ is computed in such a way that the product $\mathbf{C}\mathbf{U}\mathbf{R}$ is close to $\boldsymbol{X}$. The approximation quality obviously depends on the selection of rows and columns to construct $\mathbf{C}$ and $\mathbf{R}$.

The first step to select them in a meaningful way is to perform an SVD on the data $\boldsymbol{X}$. By doing so, the j-th column of $\boldsymbol{X}$ can be approximated as a linear combination of the first r left singular vectors

$$[\boldsymbol{X}]_j = \sum_{i=1}^{r} ([\boldsymbol{\varsigma}]_i [\boldsymbol{U}]_i) [\boldsymbol{\Omega}]_{ij}, \tag{3.6}$$

where $[\boldsymbol{\Omega}]_{ij}$ is the j-th entry of the i-th right singular vector $[\boldsymbol{\Omega}]_i$. It should be noted that the columns of $\boldsymbol{X}$ only differ in the corresponding coefficients $[\boldsymbol{\Omega}]_{ij}$ in this representation which are consequently used for the selection of columns. In the following, we want to select columns in $\boldsymbol{X}$ that exhibit a correlation with the first r right singular vectors. For this the normalized statistical leverage score

$$s_j^{\mathrm{CUR}} \coloneqq \frac{1}{r} \sum_{i=1}^{r} [\boldsymbol{\Omega}]_{ij}^2 \tag{3.7}$$

is introduced as importance measure. This score forms a probability distribution over the n_s columns. In the next step, a column of $\boldsymbol{X}$ is kept with the probability $p_j = \min\{1, c s_j^{\mathrm{CUR}}\}$, where the parameter c grows at most proportionally to $c = \mathcal{O}(r \log r / \epsilon^2)$. In this context, r and ϵ modulate the parameter c and affect the number of selected columns. The expectation of the actual number of columns satisfies $r_\mathbf{C} \leqslant c$. The error parameter ϵ modulates the approximation quality in the Frobenius norm that is achieved by at least 99%, see [MahoneyDrineas09]. All selected columns are assembled in the matrix $\mathbf{C}$.

The same procedure is applied to the transposed data matrix $\boldsymbol{X}^\intercal$ yielding $\mathbf{R}$. Once, both matrices are determined, $\mathbf{U}$ is computed as $\mathbf{U} = \mathbf{C}^\dagger \boldsymbol{X} \mathbf{R}^\dagger$, where $\square^\dagger$ is the Moore-Penrose generalized inverse. For the sake of a Galerkin projection, only the $\mathbf{C}$ matrix serves as

projection matrix and a state is projected as $\tilde{\boldsymbol{x}} = \mathbf{C}\mathbf{C}^{\dagger}\boldsymbol{x}$. As $\mathbf{C}$ is in general not orthogonal, its pseudo inverse is used for projection. The random selection of basis vectors according to their normalized statistical leverage score ensures that the selected columns (and rows) form a representative and diverse subset of the data so that not only redundant or nearly collinear columns are chosen. However, following such a procedure results in different matrices of varying sizes every time the method is applied. Moreover, the dimension of the reduced variable can not be chosen in advance. In summary, the reduction mapping using the CUR decomposition results in $\boldsymbol{\phi}_{\mathrm{CUR}}(\boldsymbol{x}) \coloneqq \mathbf{C}^{\dagger}\boldsymbol{x}$, while the reconstruction mapping is computed as $\boldsymbol{\psi}_{\mathrm{CUR}}(\boldsymbol{z}) \coloneqq \mathbf{C}\boldsymbol{z}$.

3.2.5 Limitations of Linear Techniques

Although linear reduction techniques are widely used, they have certain limitations. To assess these limitations, the so-called Kolmogorov n-width

$$\begin{aligned} d^{r}_{\mathrm{Kol}}(\mathcal{S}_{\mathcal{P}}) &\coloneqq \inf_{\substack{\mathcal{X}_r \subseteq \mathcal{X} \\ \dim(\mathcal{X}_r) \leqslant r}} \sup_{\boldsymbol{x} \in \mathcal{S}_{\mathcal{P}}} \inf_{\tilde{\boldsymbol{x}} \in \mathcal{X}_r} \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|, \\ &= \inf_{\substack{\mathcal{X}_r \subseteq \mathcal{X} \\ \dim(\mathcal{X}_r) \leqslant r}} d_{\mathrm{Kol}}(\mathcal{X}_r, \mathcal{S}_{\mathcal{P}}) \end{aligned} \tag{3.8}$$

is often used to quantify the optimal linear trial subspace ([LeeCarlberg20], [UngerGugercin19]). The first infimum in (3.8) is taken over all r-dimensional linear subspaces $\mathcal{X}_r$ of the state space $\mathcal{X}$. The function $d_{\mathrm{Kol}}(\mathcal{X}_r, \mathcal{S}_{\mathcal{P}})$ then determines the largest distance between the subspace $\mathcal{X}_r$ and any point $\boldsymbol{x}$ in $\mathcal{S}_{\mathcal{P}}$, denoting the manifold of solutions to all valid time and parameter, see (2.9). Please note that in much of the literature the dimension of the subspace is denoted as n, which is the reason for the designation n-width. In (3.8), it is chosen to be r to match the notation of this thesis.

If a problem exhibits a rapidly decaying Kolmogorov n-width, linear trial subspaces are well-founded and have been successfully applied. This is for example often the case for diffusion-dominated problems. If the Kolmogorov n-width is, however, slowly decaying, linear trial subspaces often fail to achieve sufficient accuracy, requiring significantly larger dimensions to produce reliable results [OhlbergerRave16]. Slowly decaying n-widths often occur in advection-dominated problems.

In order to investigate the complexity of the aforementioned kart model from Section 2.2.1 in this regard, let us consider the course of the singular values of the high-fidelity simulation results $\boldsymbol{X}$ as an indicator of the Kolmogorov n-width. As previously stated, the magnitude of each singular value reflects the importance of the corresponding reduced basis vector for describing the data. Consequently, the rate of decay of these values provides insight into the extent to which the space can be approximated using only the subspace spanned by the selected basis vectors. When the singular values decay rapidly, it indicates that the

dataset (which serves as a proxy for the solution manifold) can be well-approximated in low dimensions, resulting in a smaller n-width. As previously stated, the intrinsic dimension for the kart parameterization is at most $r = 4$.

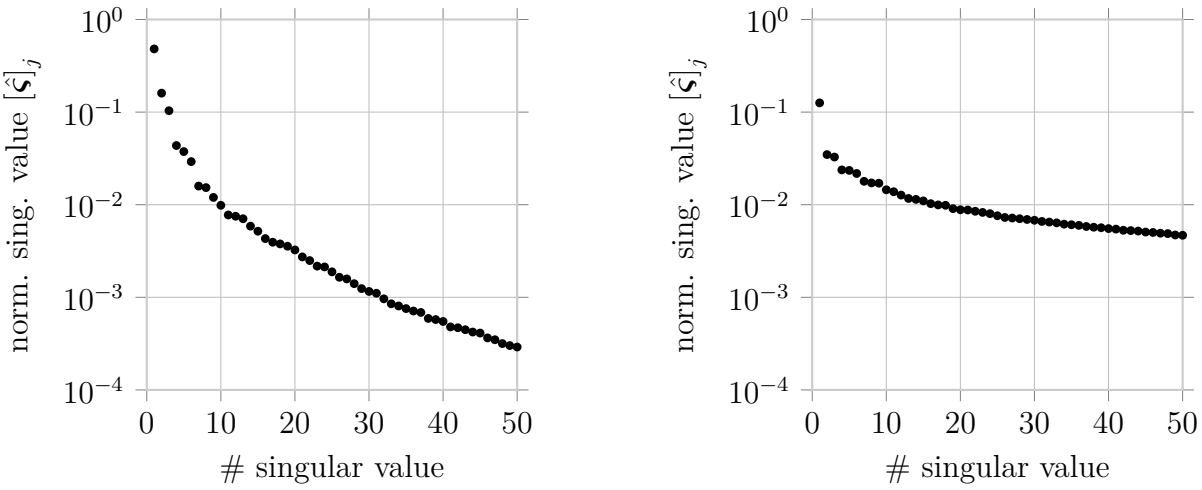
The course of normalized singular values $[\hat{\boldsymbol{\varsigma}}]_j = [\boldsymbol{\varsigma}]_j / \sum_{i=1}^{\min\{n,n_s\}} [\boldsymbol{\varsigma}]_i$ for the kart model is presented in Fig. 3.2 once obtained for displacement and once obtained for stress data. With regard to the displacements, it can be observed that, while the first four singular values make significant contributions of approximately 79 %, the subsequent singular values still exert a notable influence. Typically, one anticipates an elbow in the course, after which truncation does not introduce significant errors. However, such behavior is not observed for the displacements, and the course only flattens out slowly.

For the stress data, this becomes even more apparent and the individual components have less individual impact leading to a very flat course. The contribution of the first four singular values to the overall approximation only adds up to about 21 %. In consequence, the stress data presents a more significant challenge for linear reduction techniques than the displacements. A comparable phenomenon has also been observed in a previous study [KneiflEtAl23]. One potential explanation for this phenomenon can be found in the order of operations in which these quantities are solved for. In FE computations, the system is first solved for nodal displacements. Strains and stresses are then derived from displacement gradients according to the specified material model. Since accuracy and convergence rates decrease with each differentiation, displacements are subject to continuity criteria, whereas the stresses are not. As a result, stress discontinuities or singularities may occur at element edges, requiring localized mesh refinement to mitigate these effects. Even in less severe cases, it can be assumed that the stresses are less well distributed than the displacements and thus are more complicated to capture.

Therefore, it can be assumed that linear reduction methods like the PCA, lead to considerable errors when reducing the data to the intrinsic dimension and that a larger number of basis vectors is necessary for satisfactory approximations. Thus, nonlinear alternatives to the linear projection-based reduction methods are discussed in the following.

3.2.6 Nonlinear Reduction

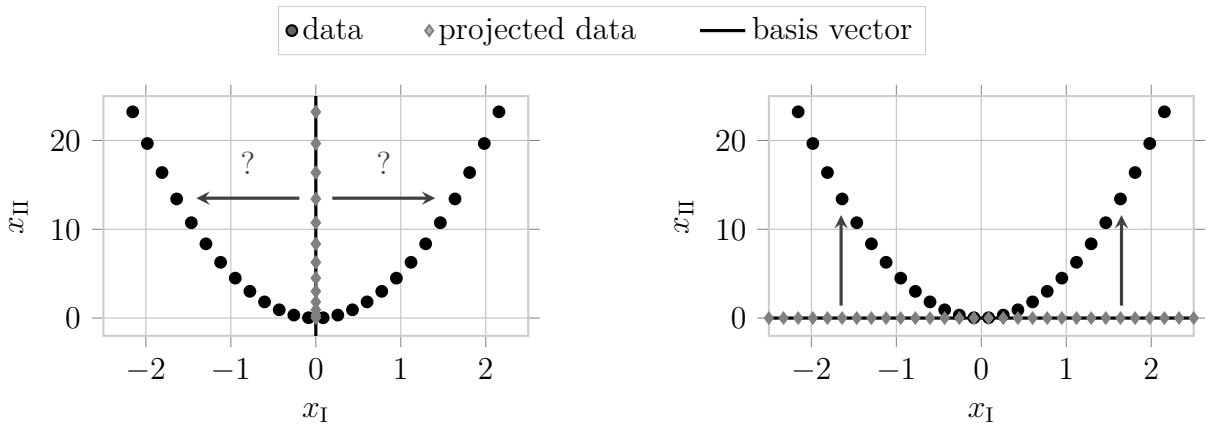
Nonlinear reduction techniques provide powerful tools for capturing complex relationships in data that linear methods may miss. Unlike linear approaches, which assume a direct relationship between features, these methods are able to model intricate, nonlinear dependencies, providing a richer representation of the underlying structure. To introduce the advantages of nonlinear reduction methods, let us examine a thought experiment presented in Ex. 1.



(a) displacements

(b) stress

Figure 3.2: The first 50 normalized singular values of the training data for a snapshot matrix composed out of displacement data (a) and stress data (b) as indicator of the Kolmogorov n-width.



(a) Projection of the data points onto the most dominant basis vector.

(b) Projection of the data points onto the second basis vector.

Figure 3.3: Projection of data onto the first and second principal component.

Example 1 (Nonlinear Decoding of Parabola)**.** *Consider a two-dimensional dataset* $\mathcal{D} = \{(x_{\mathrm{I}i}, x_{\mathrm{II}i})\}_{i=1}^{n_s}$ *where the first coordinate* $x_{\mathrm{I}} = \mu$ *corresponds to a parameter and the second* $x_{\mathrm{II}} = \mu^2$ *to its square. In this thought experiment, we want to reduce that data to only one dimension from which we can reconstruct the original data. Applying a linear reduction in the form of PCA to this dataset, will return the* x_{II}*-axis as basis vector as the most variance occurs in the second variable. A visualization of this can be found in Fig. 3.3a. Given the limitations of reconstructing the data as a linear combination of the basis vector, this is the optimal choice, but it still leaves us unsatisfied as the problem does not appear to be that complicated.*

The use of a nonlinear reconstruction, even in conjunction with a linear reduction, represents a fundamental shift to the previously described problem. Consider the linear projection onto the second principal component, i.e. the x_{I}*-axis as shown in Fig. 3.3b. While it is not feasible to reconstruct the original data from the projection onto the* x_{II}*-axis (Fig. 3.3a), it is evident that the same data can be perfectly reconstructed as the square of the projection onto the* x_{I}*-axis (Fig. 3.3a). The results demonstrate two key points. First, enabling nonlinearity in the reduction and reconstruction mapping can enhance the approximation outcomes. Second, the features that may be optimal for linear reduction techniques may not be the most suitable for alternative approaches.*

3.2.7 Kernel Principal Component Analysis

Kernel Principal Component Analysis (KPCA) was introduced in [SchölkopfSmolaMüller97] as a nonlinear extension of PCA. The fundamental concept is to transform the state onto a higher-dimensional feature space $\acute{\mathcal{X}} \subseteq \mathbb{R}^{\acute{n}}$ with $\acute{n} \gg n$, where it is more likely to achieve linear separability. In a next step, PCA is performed on the transformed data while expensive computations in the high-dimensional feature space are circumvented by employing the *kernel trick*.

In detail, a priori unknown nonlinear map $\boldsymbol{\varphi} : \boldsymbol{x} \mapsto \boldsymbol{\varphi}(\boldsymbol{x}) \in \acute{\mathcal{X}}$ transforms the states into the feature space, resulting in the transformed snapshot matrix $\acute{\boldsymbol{X}} = \begin{bmatrix}\boldsymbol{\varphi}(\boldsymbol{x}_1)\ \boldsymbol{\varphi}(\boldsymbol{x}_2)\ \ldots\ \boldsymbol{\varphi}(\boldsymbol{x}_{n_s})\end{bmatrix} \in \mathbb{R}^{\acute{n} \times n_s}$. The goal is to apply PCA to this transformed data without performing it explicitly. For this, recall that a transformed state can be approximated as a linear combination of the reduced basis vectors or principal components obtained via PCA as $\boldsymbol{\varphi}(\boldsymbol{x}_i) \approx \sum_{l=1}^{r} [\boldsymbol{z}_i]_l \, \boldsymbol{v}_j$ with $\boldsymbol{z}_i = [\boldsymbol{\varsigma}]_i \, [\boldsymbol{\Omega}]_i$. Similarly, there exist coefficients $\boldsymbol{\xi}_i$ so that the principal components can be written as linear combinations of the transformed states as

$$\boldsymbol{v}_l = \sum_{i=1}^{n_s} [\boldsymbol{\xi}_i]_l \, \boldsymbol{\varphi}(\boldsymbol{x}_i), \quad l = 1, ..., r, \tag{3.9}$$

since all solutions lie in the span of $\boldsymbol{\varphi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\varphi}(\boldsymbol{x}_{n_s})$. In KPCA, however, the principal components themselves are never calculated explicitly but only the projections of the

transformed states onto those components

$$\boldsymbol{v}_l^\mathsf{T}\boldsymbol{\varphi}(\boldsymbol{x}) = \left(\sum_{i=1}^{n_s} [\boldsymbol{\xi}_i]_l \, \boldsymbol{\varphi}(\boldsymbol{x}_i)\right)^\mathsf{T} \boldsymbol{\varphi}(\boldsymbol{x}). \tag{3.10}$$

In this equation, all vectors $\boldsymbol{\varphi}(\boldsymbol{x})$ appear only within scalar products. This enables the use of the kernel trick that replaces scalar products $\boldsymbol{\varphi}(\boldsymbol{x}_i)^\mathsf{T}\boldsymbol{\varphi}(\boldsymbol{x}_j)$ with an appropriate kernel function

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\varphi}(\boldsymbol{x}_i)^\mathsf{T}\boldsymbol{\varphi}(\boldsymbol{x}_j). \tag{3.11}$$

As a consequence, computations in the high-dimensional feature space dimension (right side of (3.11)) are replaced with computations in the state space (left side of (3.11)). Please note that the function $\boldsymbol{\varphi}$ is accordingly not explicitly formulated but only implicitly defined via the selected kernel function.

The kernel function can be applied to all samples, so that the scalar product $\boldsymbol{\varphi}(\boldsymbol{x}_i)^\mathsf{T}\boldsymbol{\varphi}(\boldsymbol{x}_j)$ from (3.10) corresponds to the elements $[\boldsymbol{\mathcal{K}}]_{ij}$ of the kernel matrix $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{n_s \times n_s}$. The kernel matrix usually needs to be centered, as a chosen kernel function $\kappa(\cdot,\cdot)$ generally does not result in a centered matrix. The coefficients $\boldsymbol{\xi}_i, i = 1, \ldots, n_s$ can then be obtained by solving an eigenvector equation of the kernel matrix $\boldsymbol{\mathcal{K}}$. With the prescribed approach, the reduction mapping of the KPCA yields

$$\boldsymbol{\phi}_{\mathrm{KPCA}}(\boldsymbol{x}) = \sum_{l=1}^{r} \boldsymbol{v}_l^\mathsf{T}\boldsymbol{\varphi}(\boldsymbol{x}) \tag{3.12}$$

where the projections $\boldsymbol{v}_l^\mathsf{T}\boldsymbol{\varphi}(\boldsymbol{x})$, see (3.10), are computed using the kernel trick (3.11). Following this procedure, it is not possible to obtain a backprojection matrix, as the projection matrix itself is never explicitly computed. Hence, the reconstruction mapping $\boldsymbol{\psi}_{\mathrm{KPCA}}(\boldsymbol{z})$ is usually obtained via kernel ridge regression [BakırWestonSchölkopf04]. A recent digest on KPCA can be found in [GarcíaGonzálezEtAl20].

3.2.8 Autoencoder

Autoencoders (AEs) and their variants are arguably the most prevalent nonlinear methods designed to learn efficient encodings of high-dimensional input data. They are a type of neural network that attempts to approximate the identity mapping, $\boldsymbol{\Psi}_{\mathrm{ae}} : \mathcal{X} \to \mathcal{X}$. However, they feature a bottleneck in the middle of their architecture, as illustrated in Fig. 3.4. This bottleneck forces the autoencoder to learn a low-dimensional latent representation of the input data.

Therefore, AEs can be split at their narrowest point into two functional parts: the *encoder* $\boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}) : \mathcal{X} \to \mathcal{Z}$ mapping the full to the latent state and the *decoder* $\boldsymbol{\psi}_{\mathrm{d}}(\boldsymbol{z}; \boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}) : \mathcal{Z} \to \mathcal{X}$ reconstructing the full state from this latent state. In this

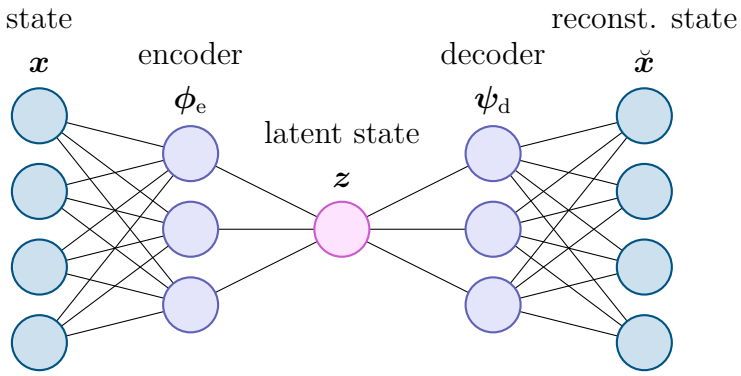


Figure 3.4: Schematic representation of an autoencoder.

context, $\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}$ and $\boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}$ are the respective weights of the individual networks. Consequently, the autoencoder

$$\breve{\boldsymbol{x}} = \boldsymbol{\Psi}_\mathrm{ae}(\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}) : x \mapsto \boldsymbol{\psi}_\mathrm{d} \circ \boldsymbol{\phi}_\mathrm{e}(\boldsymbol{x}) \tag{3.13}$$

is the result of the function composition of the encoder and decoder. The reduction mapping accordingly corresponds to the encoder $\boldsymbol{\phi}_\mathrm{AE}(\boldsymbol{x}) \coloneqq \boldsymbol{\phi}_\mathrm{e}(\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}})$, while the decoder represents the reconstruction mapping $\boldsymbol{\psi}_\mathrm{AE}(\boldsymbol{z}) \coloneqq \boldsymbol{\psi}_\mathrm{d}(\boldsymbol{z}; \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}})$. The AE is trained to optimize the reconstruction loss

$$L_\mathrm{rec}(\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}) \coloneqq \frac{1}{n_s} \sum_{i=1}^{n_s} \left\| \boldsymbol{x}_i - \boldsymbol{\Psi}_\mathrm{ae}(\boldsymbol{x}_i; \boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}) \right\|^2 = \frac{1}{n_s} \sum_{i=1}^{n_s} \left\| \boldsymbol{x}_i - \breve{\boldsymbol{x}}_i \right\|^2 \tag{3.14}$$

that measures the difference between the input and its reconstruction.

The concept of autoencoders as a nonlinear alternative to PCA was first introduced in the early 1990s [Kramer91, HintonZemel93]. Since then their development has been closely tied to advancements in machine learning and neural network research. Accordingly, they play a crucial role in deep learning applications such as anomaly detection, image denoising, data compression, and feature learning. Moreover, one of the most successful generative models of the 2010s, the Variational Autoencoder (VAE), is based on a similar structure. Other successful variants include denoising, convolutional, or graph convolutional autoencoders. Both graph convolutional autoencoders and variational play a significant role in this thesis, and detailed explanations are provided in the corresponding Sections 6.1.1 and 8.1.

3.3 Comparison of the Reduction Techniques

Before presenting additional details on aspects of surrogate modeling beyond the reduction, which will be discussed in the subsequent chapter, the presented reduction algorithms are evaluated in comparison to one another. This evaluation is conducted with respect to the reconstruction capabilities of the algorithms on the kart example, as well as with regard

to the meaning of the reduced variables in capturing characteristics of the HF model. For these considerations, the matrix including all reduced training samples

$$\begin{aligned} \boldsymbol{Z} &\coloneqq \left[\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_s}\right] \in \mathbb{R}^{r \times n_s} \\ &\text{with } \boldsymbol{z}_j = \boldsymbol{\phi}(\boldsymbol{x}_j),\ \boldsymbol{x}_j \in \boldsymbol{X}\ \forall j = 1, \ldots, n_s \end{aligned} \tag{3.15}$$

is used, where the matrix $\boldsymbol{X}$ contains the high-fidelity training snapshots as defined in (2.8). Implementation details on the individual reduction techniques are given in Table 3.1.

Table 3.1: Comparison of hyperparameters for PCA, CUR Decomposition (CUR) decomposition, Autoencoder, and KPCA approaches.

Approach	**Hyperparameter Category**	**Details**
PCA	-	
CUR	Sampling strategy	Random
	Error parameter ϵ	0.4
AE	Encoder layer size	$n \times 300 \times 150 \times 75 \times r$
	Decoder layer size	$r \times 75 \times 150 \times 300 \times n$
	Optimizer	Adam
		Learning Rate: $1e-3$
	Training	Epochs: 400
		Batchsize: 128
	Data preprocessing	Normalization
	Regularization	Select best weights w.r.t. validation data
KPCA	Kernel	Polynomial $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i{}^\intercal \boldsymbol{x}_j + 350)^3$
	Data preprocessing	Centering

Sampling the Latent Space In order to provide a preliminary comparison of the reduction algorithms, the impact that variations in the latent space have on the original state space are presented. This is an approach that is often performed in generative modeling, particularly in deep generative models like VAEs. However, this tool is rarely considered in surrogate modeling. Nonetheless, as demonstrated in our publication [KneiflEtAl23], this approach provides meaningful insights. Therefore, we extend its application to the kart model for the presented reduction techniques. For this purpose, the value of the i-th entry of the reduced state $[\boldsymbol{z}]_i$ is varied, while all other entries remain constant. Given the objective of sampling the latent variable within a range that correlates with the variables

that are observed, we sample it around the mean values

$$\bar{\boldsymbol{z}} = \frac{1}{n_s} \sum_{\boldsymbol{z}_j \in \boldsymbol{Z}} \boldsymbol{z}_j \tag{3.16}$$

occurring in the data. The component-wise calculated standard deviation

$$[\boldsymbol{z}_{\text{std}}]_i = \sqrt{\frac{1}{n_s} \sum_{\boldsymbol{z}_j \in \boldsymbol{Z}} ([\boldsymbol{z}_j]_i - [\bar{\boldsymbol{z}}]_i)^2}, \; i = 1, \ldots, r \tag{3.17}$$

is used to define the upper and lower bound of the sampled area of the i-th entry as $[\bar{\boldsymbol{z}}]_i \pm [\boldsymbol{z}_{\text{std}}]_i$. It should be noted that the mean and standard deviation vary for different reduction algorithms.

For PCA, CUR, and KPCA, all remaining entries except of the i-th one are set to zero to observe an isolated view of effects. Accordingly, the lower and upper bound of the sampled area are defined as

$$([\bar{\boldsymbol{z}}]_i \pm [\boldsymbol{z}_{\text{std}}]_i)\, \hat{\boldsymbol{e}}_i, \tag{3.18}$$

where $\hat{\boldsymbol{e}}_i$ is the standard basis, e.g. $\hat{\boldsymbol{e}}_1 = [1, 0, \ldots, 0]^\intercal$. In contrast, an alternative sampling methodology is being pursued for the AE because in that case zero entries in the latent state are not equivalent to no contribution of that variable for the reconstruction. Hence, all entries of the latent variable are set to their mean value but only the i-th entry is varied with its standard deviation following

$$\bar{\boldsymbol{z}} \pm [\boldsymbol{z}_{\text{std}}]_i\, \hat{\boldsymbol{e}}_i. \tag{3.19}$$

The sampled reduced states are then reconstructed and the results of each reduction algorithm are shown in Fig. 3.5.

In the case of linear reduction methods, the sampling of individual entries within the reduced space is analogous to the excitation of the corresponding reduced basis vector, which we call *modes* in the following for all reduction approaches. For PCA, it can be observed that the individual modes represent isolated movements such as forward movements with tilting (mode 1), lateral displacement (mode 2), or deflection of the front (mode 3). The CUR modes just represent scaled displacement vectors that occurred in the data itself. Hence, we can observe that all modes include tilting as well as forward movements but with different levels of deformations.

The interpretation of the modes for nonlinear methods is not as straightforward as it is for linear methods. Rather than representing the state as a superposition of individual basis vectors, these methods reconstruct it as a nonlinear function where variables interact with each other. Therefore, the separation of effects is harder to observe, particularly in the case of autoencoders. Regarding KPCA, multiple motions can be observed within a

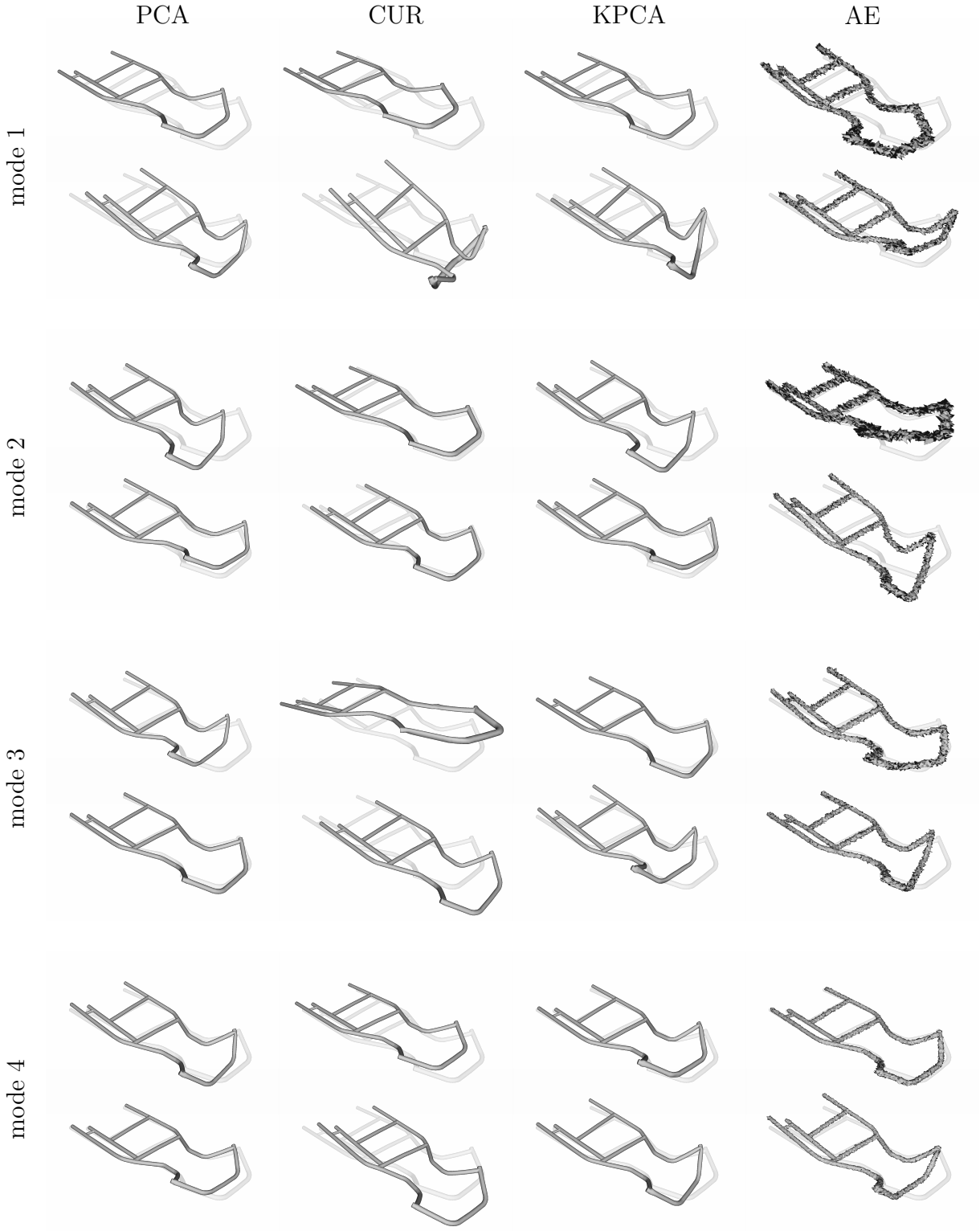


Figure 3.5: The first modes for the various reduction algorithms are illustrated for the displacement of the kart example. The reference configuration is depicted in light gray behind each deformed configuration for reference. For each mode, two states are presented: one obtained at the lower bound of the corresponding sampling area and one at the upper bound.

single mode. For instance, mode 1 combines forward motion with tilting and strong lateral deformations. It is not surprising that the forward motion and tilting occur so often for the reduction algorithms, as they are among the most dominant motions.

For the autoencoder, the modes have no direct physical interpretation and manifest as motions with superposed bizarre deformations. Please note that even if individual modes appear discontinuous, this does not imply that the autoencoder reconstructs the kart geometry in a similarly discontinuous manner. On the contrary, the overall prediction—formed by a nonlinear combination of these modes—results in a smooth surface approximation, as expected. The individual motions of the AE occur less isolated. This is not surprising as the decoder represents a chain of nonlinear deformations and all latent states interact with each other. Nevertheless, it can be observed that certain motions such as lateral deformations (mode 2 and 3), folding of the front (mode 1 and 4) or tilting (mode 1 and 4) are still dominant in individual modes. This effect can be partially enhanced by VAEs, where the latent variables are forced to exhibit greater disentanglement meaning that isolated effects in the state space correspond to isolated changes in the latent space. It should be noted that the modes for CUR and AEs are different every time the reduction methods are fitted and that the modes shown are consequently only examples. Moreover, the modes are not ordered by importance in contrast to PCA.

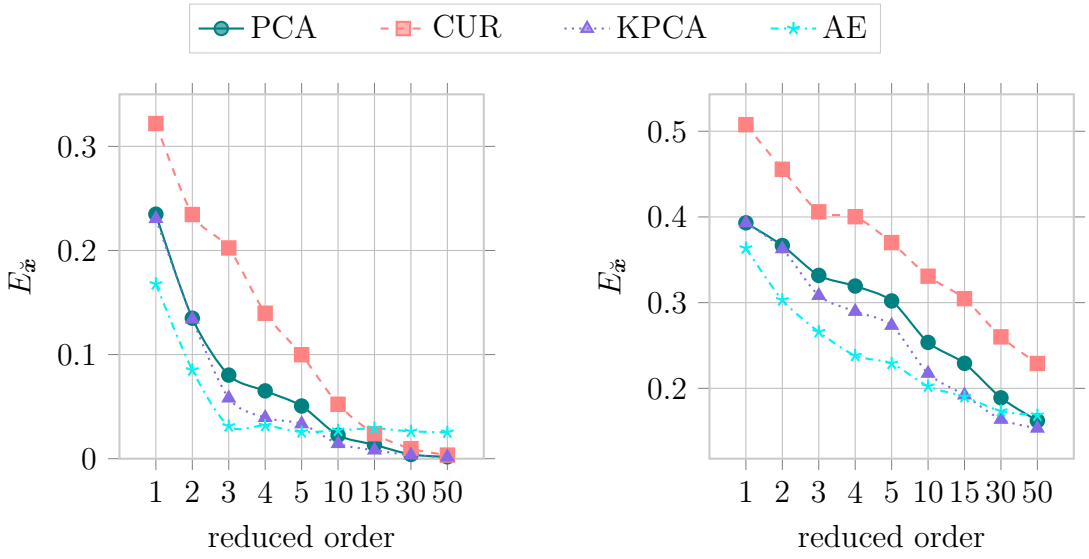
Reconstruction Error For the sake of validating the reduction performance, we consider a relative reconstruction error over a set of states

$$E_{\breve{\boldsymbol{x}}}(\boldsymbol{X}, \breve{\boldsymbol{X}}) \coloneqq \frac{\left\| \boldsymbol{X} - \breve{\boldsymbol{X}} \right\|_2}{\|\boldsymbol{X}\|_2} = \frac{\|\boldsymbol{X} - \boldsymbol{\phi}(\boldsymbol{\psi}(\boldsymbol{X}))\|_2}{\|\boldsymbol{X}\|_2} \tag{3.20}$$

that serves as an indicator for the approximation quality for a whole dataset. The results over a sweep of reduced orders $r \in \{1, 2, 3, 4, 5, 10, 15, 30, 50\}$ are given in Fig. 3.6.

Notably, the nonlinear reduction algorithms especially pay off in very low dimensions. This effect is balanced out as soon as more and more latent dimensions are added. Not surprisingly, the reduction based on the CUR decomposition performs worst as it is a linear technique that is in contrast to PCA suboptimal. In return, it offers interpretable and physical modes. Regarding the nonlinear methods, AE clearly performs best when the reduced dimensionality is close to the intrinsic dimension. However, if the reduced dimension is increased beyond a certain threshold, the other methods prove to be more efficacious. The reconstruction abilities of KPCA settles between the results of PCA and AE for a wide range of reduced orders. For higher ones, it can even achieve the best results.

Only considering the results obtained for the displacement data presented in Fig. 3.6a, reveals that increasing the reduced dimension beyond a certain point does not offer any advantage for the autoencoder, and the performance remains stagnant. This phenomenon



(a) displacement test data

(b) stress test data

Figure 3.6: Relative reconstruction error on displacement (a) and stress test data (b). Please note that the results for the CUR decomposition and AEs vary for each fit. Hence, five runs are conducted for each method and the displayed result is the mean reconstruction error over the runs.

may be attributed to the increased likelihood of the autoencoder becoming trapped in local minima with an increased reduced dimension. Furthermore, convergence behavior can be observed for all algorithms, with only minor performance enhancements occurring in higher dimensions.

Conversely, the stress data demonstrates a divergent pattern, see Fig. 3.6b. It is evident that even when the reduced dimension is set to higher values, the reconstruction error undergoes a substantial decrease as the dimensionality increases. Respectively, the superior performance of nonlinear over linear methods is sustained for a wider range of reduced dimensions than previously observed for the displacements. A notable observation of the present results is that the reduction of stress data poses a substantially more substantial challenge in comparison to that of displacements data. This finding is consistent with the previous observations regarding the course of the corresponding singular values discussed in Section 3.2.5.

It can be stated that, in general, the linear reduction techniques yields satisfactory reconstruction results for the kart example, provided that a latent space of a sufficiently high dimension is considered. It has been demonstrated that nonlinear methods can clearly outperform their linear counterparts in reducing the system to very low dimensions. Furthermore, it is possible that they yield superior results with a more exhaustive optimization of hyperparameters.

3.4 Further Considerations of Desired Attributes of the Coordinates

A low-dimensional system representation is of significant benefit. However, as stated in Chall. 2, this is not the sole attribute that the latent coordinates should possess. For example, many applications favor a linear system description over a nonlinear one to apply linear control theory or linear system analysis. The entire research field of Koopman theory [BruntonEtAl22], for instance, is centered around this topic and represents the dynamics of nonlinear systems with the linear Koopman operator, named after Bernard Osgood Koopman [Koopman31]. This is accomplished by analyzing functions of the states rather than the states themselves.

Moreover, the optimality statement of the PCA is only valid regarding the reconstruction. It does not acknowledge the dynamics of a system itself and thus evolving the state in such a subspace may lead to larger errors in the overall approximation of a surrogate model. Hence, in certain scenarios, it is beneficial to consider more than just the reconstruction quality of the states when constructing the latent space. In light of these considerations, AEs have a significant advantage over their competitors, as they can be easily customized to accommodate further aspects beyond the mere reconstruction. This can, for example, be achieved by enriching the reconstruction loss (3.14) by additional terms. A more detailed examination of this procedure will be provided in subsequent chapters. For the time being, we have derived methods capable of efficiently reducing the dimensionality of a system from an isolated perspective. In the subsequent chapter, we discuss how dynamics can be captured in an isolated process, before moving on to combining those two pillars of surrogate modeling.

Chapter 4

Dynamics Decoded: From Black-box Approaches to System Identification

So many people telling me one way
So many people telling me to stay
Never time to have my mind made up
Caught in a motion that I don't wanna stop

The Whitest Boy Alive, Burning

The reduction methods that have been considered thus far are capable of identifying low-dimensional variables to represent high-dimensional states. However, these methods do neither account for time nor dynamics. Consequently, introducing them in a surrogate modeling setting represents only the first step in the process of approximating a High-Fidelity (HF) model. The second step must involve a strategy for capturing the dynamics and evolving them over time. It is evident that there are numerous possibilities that can be employed to address this issue. One may endeavor to approximate the solution for a given setup directly, to evolve the dynamics in a sequential fashion, or to identify an explicit description of the dynamics. In this thesis, a distinction is made between *black-box modeling* and *system identification* as well as a distinction between methods that aim to *capture the state evolution* (e.g. the flow map (2.7)) directly, and methods that aim to *match the dynamics equations*, i.e. that mimic the first order time derivatives of an Ordinary Differential Equation (ODE) or Partial Differential Equation (PDE) defined by the right hand sides of (2.1) and (2.3).

In the context of dynamical systems, black-box approaches are designed to directly model the state evolution, often without explicit assumptions about physical laws or governing equations. These methods address the challenge of approximating the dynamics as input-

output mappings or state transition relationships and predict states based solely on past observations or input parameters. In consequence, rather than representing the evolution of a system in terms of differential equations or operators, these methods treat the system as a functional mapping that relates the time t, the current state $\boldsymbol{x}(t_i)$, or the parameters $\boldsymbol{\mu}$ to a specified state $\boldsymbol{x}(t, \boldsymbol{\mu})$. Classically, regression methods such as Neural Networks (NNs) or Gaussian Processes (GPs) are employed for these tasks. While black-box approaches offer valuable predictive capabilities, they are not without limitations. By focusing exclusively on input-output relationships, they frequently lack the interpretability and generalizability that are characteristic of physics-based methods. Consequently, their performance may decline when extrapolated beyond the training data or when confronted with scenarios that fall outside the observed data.

System identification methods [Tangirala18, BruntonKutz22], on the contrary, aim to model a system by discovering or approximating the mathematical relationships that govern its behavior. Unlike black-box approaches, these approaches aim to represent the underlying dynamics explicitly, often in the form of ODEs or PDEs. Therefore, they attempt to reconstruct the governing equations from data, enabling not only state prediction but also interpretability, generalization, and insights into the system's behavior. While these methods are powerful, they are often constrained by the assumptions made about the system's structure or the availability of noise-free data. Additionally, highly complex or chaotic systems where the true dynamics cannot be easily parameterized represent a major challenge.

Addendum 2 (Approaches to Data-Driven Modeling of System Dynamics)**.** *A plethora of methodologies have been proposed in the existing literature that can be placed in one or the other category or somewhere in between. Physics-Informed Neural Networks (PINNs) [RaissiPerdikarisKarniadakis19], for example, pose a compromise of the mentioned categories. They directly approximate the solution of an ODE or PDE but penalize deviations from the solution's time derivative of a known governing equation. In addition, although they compute the solution of a PDE as a mere black-box, they can also be used for parameter identification. They are employed in a multitude of applications in the approximation of PDEs and beyond. One such example can be found in our own published work, where we utilized them in the context of model predictive control for robotic manipulators [NicodemusEtAl22].*

Hamiltonian neural networks [GreydanusDzambaYosinski19], on the contrary, learn time derivatives of coordinates as state derivatives of a Hamiltonian parameterized by a neural network. The so computed time derivatives can then be integrated in time to evolve the system. Hence, they are a black-box modeling method (as no explicit dynamics equations are derived) approximating first order time derivatives that take conservation laws into account. Neural ODEs [ChenEtAl18] are a class of deep learning models formulating the transformation of data as a continuous-time dynamical system. They parameterize the

right hand side $\boldsymbol{f}$ of an ODE as neural network and evolve this state using numerical solvers. This renders them inherently suited to approximating dynamical systems. Neural Operators [KovachkiEtAl23] learn mappings between infinite-dimensional function spaces, i.e. their in- and outputs are functions rather than finite-dimensional vectors. Hence, they can approximate solution operators of ODEs and PDEs (i.e. solutions of a PDEs for any boundary condition or parameter setting) instead of a specific instance of the solution. Fourier Neural Operators [LiEtAl20a] leverages the Fourier transform to parameterize operators in the frequency domain and Deep Operator Networks [LuEtAl21] model function-to-function mappings using distinct networks to encode the input function and the desired output locations.

In the context of system identification, Sparse Identification of Nonlinear Dynamics (SINDy) [BruntonProctorKutz16] is among the most popular methods. This method involves formulating an explicit dynamics equation as a linear combination of function candidates from a preselected library of functions, with each function weighted by a learnable coefficient. Dynamic Mode Decomposition (DMD) [Schmid10, Kutz16], on the contrary, identifies a state-space model and can be used to analyze spatiotemporal modes (dynamic modes). Operator inference [PeherstorferWillcox16, KramerPeherstorferWillcox24] is another method that identifies operators (e.g., linear or quadratic ones) that govern the dynamics of a system.

An alternative approach is system identification using symbolic regression, which not only determines the parameters of the governing equations, as in standard regression, but also discovers their mathematical form. Instead of assuming a predefined equation structure (as in linear or polynomial regression), symbolic regression explores various mathematical expressions to find the best-fitting model. Such approaches have been employed, for example, in [BongardLipson07, SchmidtLipson09]. In contrast to predefined model structures, symbolic regression generates the unknown dynamics as a combination of elementary mathematical operations and functions. This process is frequently driven by evolutionary algorithms, such as genetic programming, which iteratively explore and refine potential equations by searching through combinations of terms and operations. A categorization of the mentioned methods into the distinctive modeling approaches is given in Fig. 4.1.

4.1 Black-box Modeling

As mentioned, black-box modeling techniques do not attempt to interpret the underlying physics of a system, instead they seek to directly represent the state evolution of a system. In this thesis, it is focused on black-box methods that do not approximate the dynamics equation but the solution directly.

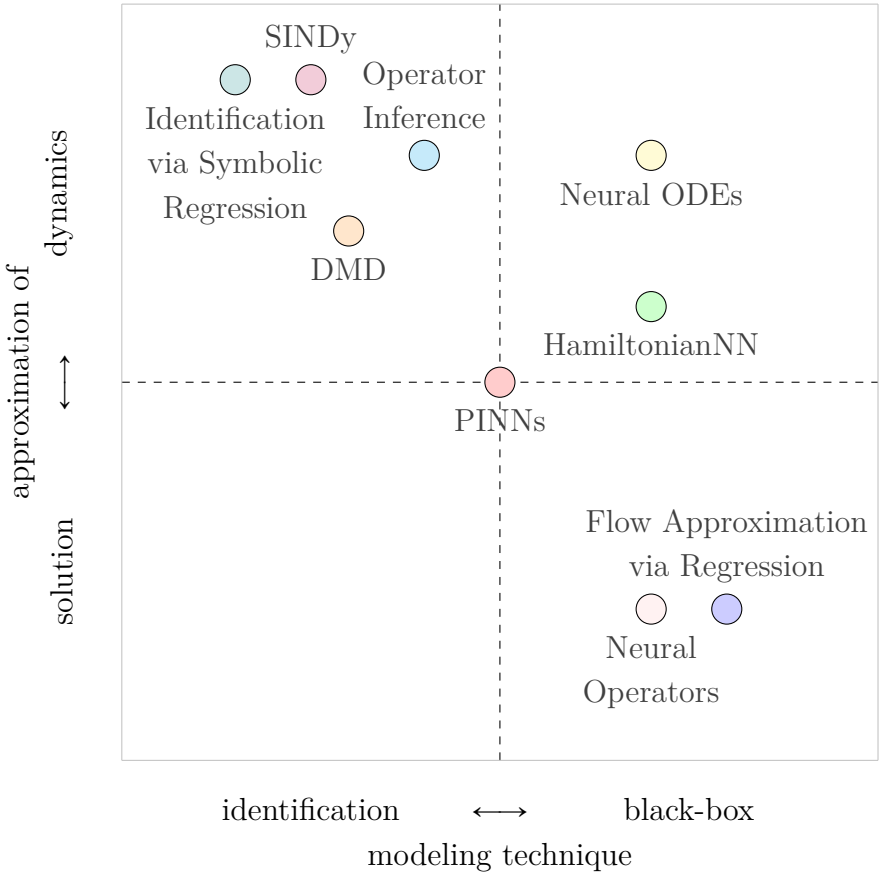


Figure 4.1: Qualitative categorization of methods for approximating a dynamical system into the categories of *identification* and *black-box* modeling, and direct approximation of the solution or approximation of the dynamics.

4.1.1 State Evolution Modeling

One way to approach the task of creating models to approximate state evolutions, is to predict the state of a system for a given time instance, initial condition, and parameters using a regression algorithm $\boldsymbol{\theta} : \mathcal{T} \times \mathcal{P} \times \mathcal{X} \to \mathcal{X}$. Hence, they are a direct proxy of the flow map (2.7). In order to train the model, we first need to build a corresponding dataset out of the available quantities. In this dataset, each state is assigned to the corresponding input as a tuple following

$$\mathcal{D}^{\text{flow}} \coloneqq \left\{ \left(\underbrace{\begin{bmatrix} t_i \\ \boldsymbol{\mu}_i \\ \boldsymbol{x}_{0i} \end{bmatrix}}_{\text{inputs } \boldsymbol{\alpha}_i}, \underbrace{\boldsymbol{x}(t_i, \boldsymbol{\mu}_i, \boldsymbol{x}_{0i})}_{\text{outputs } \boldsymbol{y}_i} \right) \right\}_{i=1}^{n_s}, \tag{4.1}$$

where the n_s samples cover the data of all simulations. It should be noted that the initial value and time-invariant parameters do not vary throughout the course of one simulation. In that case, the initial value samples $\boldsymbol{x}_{0l} = \boldsymbol{x}_{0l+1} = \cdots = \boldsymbol{x}_{0l+n_t}$ as well as the parameter

samples $\boldsymbol{\mu}_l = \boldsymbol{\mu}_{l+1} = \cdots = \boldsymbol{\mu}_{l+n_t}$ remain constant for n_t samples assuming that l is the first sample of a simulation and the samples are ordered. A regression model, such as an NN or GP, can be directly trained on the dataset to obtain the desired surrogate model. However, models trained in this manner may be prone to overfitting, and their predictions may not satisfy the physical constraints. To circumvent this issue, we can incorporate regularization that accounts for the conditions associated with dynamic systems.

There exist different ways to regularize a regression model $\tilde{\boldsymbol{F}}$ with dynamics information. For example, one might know that the time derivatives are only positive, that the rate of change is limited by some physical constraints, or that the solution fulfills smoothness constraints. In such cases, it is possible to penalize the model if it violates these constraints. The dataset (4.1) is constructed in such a way that the time is an explicit input for the model. Hence, it is straightforward to derive the temporal derivative of the predicted state

$$\dot{\tilde{\boldsymbol{x}}} = \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{x}} = \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{F}}(t, \boldsymbol{\mu}, \boldsymbol{x}_0) \tag{4.2}$$

by the surrogate $\tilde{\boldsymbol{F}}$. When, in addition to the state data, the state time derivatives $\dot{\boldsymbol{x}}$ are available as data they can be directly used for regularization. If that is not the case, the time derivatives can still be computed using numerical derivation methods assuming that the resolution in time is fine enough and the data is not too noisy. In either way, the deviation of the time derivative of the regression model and the respective time derivative data can be used to fit the model in a way that respects the dynamics.

Let's consider a neural network that approximates the state from time, parameters, and initial values, i.e the input-output mapping defined in (4.1) as $\tilde{\boldsymbol{x}} = \boldsymbol{\Psi}(t, \boldsymbol{\mu}, \boldsymbol{x}_0; \boldsymbol{W})$. We can formulate a loss function that pushes the model closer to the actual state and at the same time takes the time derivative evolution into account as

$$L(\boldsymbol{W}) = \frac{1}{n_s}\sum_{i=1}^{n_s}(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^2 + \lambda_{\mathrm{dyn}}(\dot{\boldsymbol{x}}_i - \dot{\tilde{\boldsymbol{x}}}_i)^2, \tag{4.3}$$

where λ_{dyn} is a positive weighting factor to balance the impact of the individual loss terms.

4.1.2 Exploitation of Sequential Information

The aforementioned approach encounters an issue when time-varying parameters are introduced. Only considering the parameter $\boldsymbol{\mu}_i$ associated with the time t_i does not provide sufficient information to predict the corresponding state $\boldsymbol{x}_i$, as the trajectory that led to this point is not uniquely defined by this single input parameter but rather by the sequence of past parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_i$ since the initial state. One potential solution is to pass the entire parameter sequence pertinent to the current scenario as input to the model. However, this approach is not practical for long sequences and is unable to accommodate

varying time lengths. Fortunately, autoregressive models offer a better suited way of exploiting temporal dependencies inherent in the data by sequentially evolving states.

Autoregressive models generate sequences in which the value of a variable at a given point in time is expressed as a function of its past values. Hence, they are especially tailored to cope with sequential data as it often occurs in text but also in the time evolution of dynamical systems. In contrast to the aforementioned approach, these methods evolve the dynamics in a step-by-step manner and thus are unable to directly predict a state for a selected time. However, they can yield superior predictive power as they are capable of utilizing the complete information encoded in a history of states. Popular models include Recurrent Neural Networks (RNNs) such as Long Short-Term Memorys (LSTMs) or autoregressive transformers [VaswaniEtAl17] like GPT [RadfordEtAl18] or BERT [DevlinEtAl18]. Other models like temporal convolutional networks exploit information stored over a temporal sequence by moving a filter around a state sequence.

In our setting, we aim for sequence-to-sequence learning from an input sequence representing the time-varying parameters to an output sequence representing the system states. Accordingly, the input-output mapping we want to learn is defined by the dataset

$$\mathcal{D}^{\text{seq}} \coloneqq \left\{ \left(\underbrace{\left[\begin{bmatrix} \boldsymbol{\mu}_{0,l} \\ \boldsymbol{x}_{0,l} \end{bmatrix}, \ldots, \begin{bmatrix} \boldsymbol{\mu}_{n_t,l} \\ \boldsymbol{x}_{0,l} \end{bmatrix} \right]}_{\text{input sequence } \boldsymbol{\alpha}_{1,l},\ldots,\boldsymbol{\alpha}_{n_t,l}}, \underbrace{[\boldsymbol{x}_{1,l}, \ldots, \boldsymbol{x}_{n_t,l}]}_{\text{output sequence } \boldsymbol{y}_{1,l},\ldots,\boldsymbol{y}_{n_t,l}} \right) \right\}_{l=1}^{n_{\text{sim}}}, \tag{4.4}$$

that covers all n_{sim} trajectories each obtained from a single simulation. Please note that the i-th state, i.e. the state at time t_i, of the l-th simulation/trajectory is abbreviated as $\boldsymbol{x}_{i,l} = \boldsymbol{x}(t_{i,l}, \boldsymbol{\mu}_{i,l}, \boldsymbol{x}_{0,l})$. A analogue notation is used for the other quantities.

Recurrent Neural Networks Consider sequential data in the form of an ordered time series $\boldsymbol{y}_{1,l}, \ldots, \boldsymbol{y}_{n_t,l}$ along with a corresponding input sequence $\boldsymbol{\alpha}_{1,l}, \ldots, \boldsymbol{\alpha}_{n_t,l}$ as in (4.4). Recurrent neural networks, including LSTMs, store temporal information from previous samples of a sequence in a hidden state $\boldsymbol{h} \in \mathbb{R}^{n_\text{h}}$ using feedback connections, see e.g. [Aggarwal23]. In a standard recurrent network, the hidden state $\boldsymbol{h}_j$ and the output $\boldsymbol{y}_j \in \mathbb{R}^{n_{\boldsymbol{y}}}$ for the j-th prediction in a sequence are defined as

$$\boldsymbol{h}_j = \boldsymbol{a}^{\boldsymbol{h}}(\boldsymbol{W}^{\boldsymbol{h}\boldsymbol{\alpha}}\boldsymbol{\alpha}_j + \boldsymbol{W}^{\boldsymbol{h}\boldsymbol{h}}\boldsymbol{h}_{j-1} + \boldsymbol{b}^{\boldsymbol{h}}) =: \boldsymbol{f}^{\boldsymbol{h}}(\boldsymbol{\alpha}_j, \boldsymbol{h}_{j-1}) \tag{4.5}$$

$$\boldsymbol{y}_j = \boldsymbol{a}^{\boldsymbol{y}}(\boldsymbol{W}^{\boldsymbol{y}\boldsymbol{h}}\boldsymbol{h}_j + \boldsymbol{b}^{\boldsymbol{y}}) =: \boldsymbol{f}^{\boldsymbol{y}}(\boldsymbol{\alpha}_j, \boldsymbol{h}_{j-1}), \tag{4.6}$$

where $\boldsymbol{\alpha}_j \in \mathbb{R}^{n_{\boldsymbol{\alpha}}}$ represents the j-th input, $\boldsymbol{W}^{\boldsymbol{h}\boldsymbol{\alpha}} \in \mathbb{R}^{n_\text{h} \times n_{\boldsymbol{\alpha}}}$ is the input-to-hidden weight matrix, $\boldsymbol{W}^{\boldsymbol{h}\boldsymbol{h}} \in \mathbb{R}^{n_\text{h} \times n_\text{h}}$ is the hidden-to-hidden weight matrix, and $\boldsymbol{W}^{\boldsymbol{y}\boldsymbol{h}} \in \mathbb{R}^{n_{\boldsymbol{y}} \times n_\text{h}}$ is the hidden-to-output weight matrix. The biases are given by $\boldsymbol{b}^{\boldsymbol{h}} \in \mathbb{R}^{n_\text{h}}$ and $\boldsymbol{b}^{\boldsymbol{y}} \in \mathbb{R}^{n_{\boldsymbol{y}}}$, while $\boldsymbol{a}^{\boldsymbol{h}}$ and $\boldsymbol{a}^{\boldsymbol{y}}$ denote the activation functions of the hidden and output states, respectively. Accordingly, the same weights are shared across all predictions, ensuring that the same underlying functions are applied to all elements in the sequence. Additionally, due to the recursive nature of (4.5), RNNs can process sequences of varying lengths.

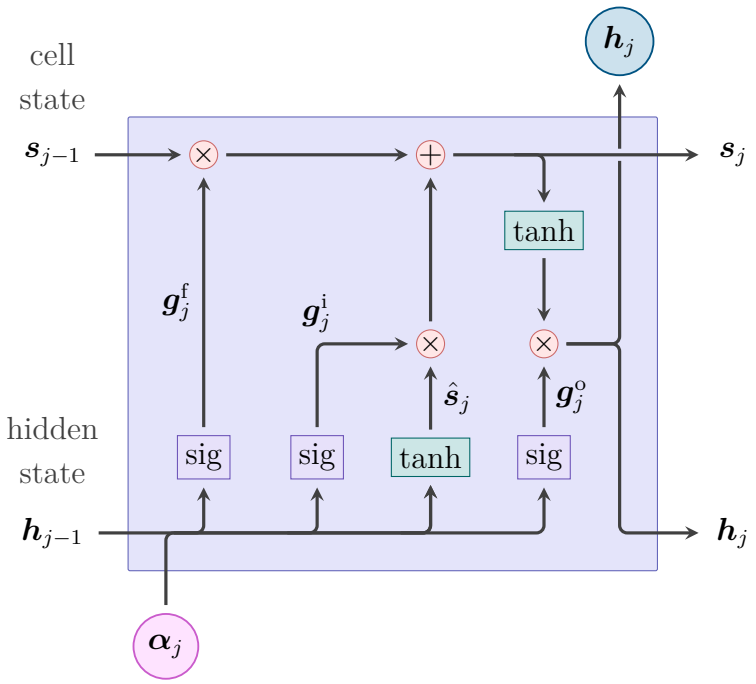


Figure 4.2: Structure of an LSTM block with pointwise operations $\odot$ and neural network layers sig for the gates with sigmoid activation functions restricting their output to $(0,\ 1)$, which corresponds to either a closed or open gate. Additionally, tanh represent layers with a tanh activation function. The gate units $\boldsymbol{g}_j^{\mathrm{f}}$, $\boldsymbol{g}_j^{\mathrm{i}}$ and $\boldsymbol{g}_j^{\mathrm{o}}$ control the data flow. Time-dependencies are propagated through the hidden state.

Long Short-Term Memory Networks Classical RNNs suffer from vanishing or exploding gradients. To address this limitation, [HochreiterSchmidhuber97] proposed to use gates to maintain a constant error flow through internal states. Their proposed networks are referred to as long short-term memory networks. The structure of an LSTM block is shown in Fig. 4.2 and can be described by

$$\begin{aligned} \boldsymbol{g}_j^{\mathrm{f}} &= \mathrm{sig}(\boldsymbol{W}^{\mathrm{f}} \left[\boldsymbol{h}_{j-1}^{\intercal},\, \boldsymbol{\alpha}_j^{\intercal}\right]^{\intercal} + \boldsymbol{b}^{\mathrm{f}}) & \hat{\boldsymbol{s}}_j &= \tanh(\boldsymbol{W}^{c} \left[\boldsymbol{h}_{j-1}^{\intercal},\, \boldsymbol{\alpha}_j^{\intercal}\right]^{\intercal} + \boldsymbol{b}_c) \\ \boldsymbol{g}_j^{\mathrm{i}} &= \mathrm{sig}(\boldsymbol{W}^{\mathrm{i}} \left[\boldsymbol{h}_{j-1}^{\intercal},\, \boldsymbol{\alpha}_j^{\intercal}\right]^{\intercal} + \boldsymbol{b}^{\mathrm{i}}) & \boldsymbol{s}_j &= \boldsymbol{g}_j^{\mathrm{f}} \boldsymbol{s}_{j-1} + \boldsymbol{g}_j^{\mathrm{i}} \hat{\boldsymbol{s}}_j \\ \boldsymbol{g}_j^{\mathrm{o}} &= \mathrm{sig}(\boldsymbol{W}^{\mathrm{o}} \left[\boldsymbol{h}_{j-1}^{\intercal},\, \boldsymbol{\alpha}_j^{\intercal}\right]^{\intercal} + \boldsymbol{b}^{\mathrm{o}}) & \boldsymbol{h}_j &= \boldsymbol{g}_j^{\mathrm{o}} \tanh(\boldsymbol{s}_j). \end{aligned} \tag{4.7}$$

The cell state $\boldsymbol{s} \in \mathbb{R}^{n_{\mathrm{h}}}$ provides long-term memory capabilities, while the hidden state $\boldsymbol{h} \in \mathbb{R}^{n_{\mathrm{h}}}$ acts as the cell's output. The weight matrices $\boldsymbol{W}^{\mathrm{f}}, \boldsymbol{W}^{\mathrm{i}}, \boldsymbol{W}^{c}, \boldsymbol{W}^{\mathrm{o}} \in \mathbb{R}^{n_{\mathrm{h}} \times (n_{\mathrm{h}} + n_{\boldsymbol{\alpha}})}$ and biases $\boldsymbol{b}^{\mathrm{f}}, \boldsymbol{b}^{\mathrm{i}}, \boldsymbol{b}^{c}, \boldsymbol{b}^{\mathrm{o}} \in \mathbb{R}^{n_{\mathrm{h}}}$ govern the internal computations. In this framework

- the forget gate $\boldsymbol{g}_j^{\mathrm{f}} \in \mathbb{R}^{n_{\mathrm{h}}}$ determines whether specific information is retained in the cell,
- the input gate $\boldsymbol{g}_j^{\mathrm{i}} \in \mathbb{R}^{n_{\mathrm{h}}}$ controls the incorporation of new information into the cell,
- the output gate $\boldsymbol{g}_j^{\mathrm{o}} \in \mathbb{R}^{n_{\mathrm{h}}}$ specifies which information is passed from the cell to the next layer or time step.

All gates utilize a sigmoid activation function $\mathrm{sig}(\cdot)$. For a detailed discussion of LSTMs, refer to [HochreiterSchmidhuber97].

Typically, a single LSTM block, as defined in (4.7), is reused recursively across all time steps to form an LSTM layer. Multiple such layers can be stacked in series to construct a deep LSTM network. In the final layer of the network, the hidden states correspond to the network's output, i.e. $\boldsymbol{y}_j = \boldsymbol{h}_j$. Consequently, the multi-time-step dependency is also reflected in the output, and the size of the last layer must match the output dimension $\mathbb{R}^{n_{\boldsymbol{y}}}$. For the first time sample of a sequence, the hidden and cell state are initialized often as zeros since there is no previous track of them.

4.2 System Identification

A mathematical description of a system should be expressive enough to cope the dynamics present in the data, yet simple enough to be interpretable. Although these objectives appear to be in opposition, they are not necessarily incompatible in practice. A parsimonious identified model, i.e. a model in a sparse representation that constitutes a minimal set of terms and parameters, has frequently been shown to be more effective than a model with an arbitrary number of terms and parameters. It employs the fewest coefficients or basis functions necessary to fit the data, often resulting in greater robustness, superior generalizability and reduced risk of overfitting. Additionally, it is more accessible for analysis and interpretation. In light of the aforementioned points, it can be concluded that parsimony can be employed as an effective regularizer for dynamical systems [KutzBrunton22, BruntonKutz23]

One method for enforcing parsimony is sparse regression, which aims to identify a model with a minimal number of non-zero coefficients by enforcing sparsity in the solution. The fundamental concept is to fit a model to data while minimizing the number of terms included in the model. The method that transfers this idea to system identification and that is adopted throughout this thesis is SINDy.

4.2.1 Sparse Identification of Nonlinear Dynamical Systems

SINDy [BruntonProctorKutz16] is designed to approximate the dynamics of a system as an ODE of the form (2.3). To achieve this, the right-hand side of the equation is approximated as a linear combination of terms drawn from a library $\boldsymbol{\Theta}(\boldsymbol{x}, \boldsymbol{\mu}) \in \mathbb{R}^{n_{\boldsymbol{\Theta}}}$ of $n_{\boldsymbol{\Theta}}$ candidate functions of the states and parameters. This library is often composed of constant and polynomial functions, trigonometric terms, or combinations of these, e.g.

$$\boldsymbol{\Theta}(\boldsymbol{x}, \boldsymbol{\mu}) = \left[1, [\boldsymbol{x}]_1, \ldots, [\boldsymbol{x}]_n, [\boldsymbol{x}]_1^2, \ldots, [\boldsymbol{x}]_2 [\boldsymbol{\mu}]_1, \ldots, \sin([\boldsymbol{x}]_1), [\boldsymbol{\mu}]_1 \cos([\boldsymbol{x}]_2), \ldots\right]^{\intercal}. \quad (4.8)$$

The individual terms in the library are weighted by corresponding learnable coefficients $\boldsymbol{\Xi} \in \mathbb{R}^{n \times n_{\boldsymbol{\Theta}}}$, and the dynamics are approximated as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\mu}) \approx \boldsymbol{\Xi}\boldsymbol{\Theta}(\boldsymbol{x}, \boldsymbol{\mu}). \tag{4.9}$$

The selection of candidate functions is typically guided by prior knowledge of the physical system, such as known dependencies on state variables or parameters, to ensure both accuracy and interpretability. As previously mentioned, SINDy uses sparse regression to estimate the coefficients in $\boldsymbol{\Xi}$, identifying only the most relevant terms from $\boldsymbol{\Theta}$ to approximate $\boldsymbol{f}$. This sparsity-enforced approach yields parsimonious models that strike a balance between accuracy and simplicity, preventing overfitting and enhancing generalizability.

In order to ascertain the most suitable coefficients, we construct a dataset in accordance to

$$\mathcal{D}^{\text{SINDy}} \coloneqq \left\{ \left(\underbrace{\begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{\mu}_i \end{bmatrix}}_{\text{inputs } \boldsymbol{\alpha}_i}, \underbrace{\dot{\boldsymbol{x}}_i}_{\text{outputs } \boldsymbol{y}_i} \right) \right\}_{i=1}^{n_s}. \tag{4.10}$$

In contrast to the preceding methodologies, the objective is not to disclose the corresponding input-output mapping, but rather to utilize the data to determine the optimal coefficients $\boldsymbol{\Xi}^*$. One approach to this problem is to employ least-squares with ℓ_1 or ℓ_2 regularization so that

$$\boldsymbol{\Xi}^* = \arg\min_{\boldsymbol{\Xi}} \frac{1}{2} \sum_{i=1}^{n_s} \left\| \dot{\boldsymbol{x}}_i - \boldsymbol{\Xi}\boldsymbol{\Theta}(\boldsymbol{x}_i, \boldsymbol{\mu}_i) \right\|_2^2 + \lambda \left\| \boldsymbol{\Xi} \right\|_{1/2}^2. \tag{4.11}$$

Often Sequentially Thresholded Least Squares (STLSQ) is applied where coefficients below a threshold value are iteratively removed to promote sparsity. Once the system is identified, it can be evolved in time using numerical time-stepping schemes.

SINDy is particularly effective for systems with low-dimensional dynamics, where the governing equations are dominated by a small number of key terms boosting the importance of dimensionality reduction. Moreover, the choice of the library plays a crucial role in the success of the method. A rich library improves expressiveness but may increase computational cost and risk of overfitting, while a sparse library may limit the model's capacity to capture complex behaviors. Extensions of SINDy also exist for low-data and high-noise regimes, e.g. using ensembles [FaselEtAl22], or for high-dimensional systems utilizing autoencoders [ChampionEtAl19].

4.3 Discussion and Comparison

The aforementioned methods all have their justification and should be selected according to the circumstances defined by the model that is to be approximated. Take for example

the kart frame crash simulation example from Section 2.2.1. It is a contact problem that exhibits non-smooth dynamics due to the sudden change in forces as the impact occurs. While the standard libraries in SINDy typically include smooth functions and cannot represent such certain changes in dynamics, extending the library to include discontinues terms can complicate the sparse regression process and introduces numerical challenges. Hence, such identification methods might not be suitable for use in this particular context. However, they may be the best choice when the analysis of the identified system is of crucial importance or when evaluation of the model must occur beyond the parameters of the training regime. Additionally, identified models can be used for tasks like continuation, enabling analysis of system behavior under extended conditions.

Black-box approaches, on the contrary, can be used for the kart example, see [KneiflGrunertFehr21, KneiflEtAl24]. These methods excel in modeling high-dimensional systems and can be computationally more efficient in such cases. However, black-box models often lack transparency, providing limited insights into the underlying system dynamics. They also tend to perform poorly when extrapolated outside the training data. As a result, the choice of approach for a specific problem must balance several factors, including the desired depth of analysis, the availability and quality of data, the system's properties, and the intended deployment use case.

4.3.1 Numerical Experiments

To provide a preliminary assessment of the aforementioned approaches and validate the aforementioned statements, a series of three preliminary numerical experiments is conducted, offering insights into strengths and limitations in controlled environments. Two of the example are represented by the already known kart and occupant example, see Section 2.2.1 and Section 2.2.2. For these examples, the surrogate modeling task is to approximate time trajectories of single nodes as we do not want to cope with dimensionality reduction at this point. For the kart, a node in the front of the model, where the greatest deformations occur during the collision, is selected. For the occupant, a node in his head, which undergoes the most movements, is selected. The third model is represented by a simple ODE

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}x(t,\mu) = \mu + 5x(t,\mu) - 0.25x(t,\mu)^2 \text{ with } t \in \mathcal{T} \\ x_0(\mu) = x(0,\mu) = 0 \end{aligned} \tag{4.12}$$

where the parameter μ is sampled randomly from a uniform distribution over $[0, 3)$, the time interval is chosen to be $\mathcal{T} = [0, 3]\,\mathrm{s}$, and the initial condition is set to zero. In total, $n_{\mathrm{sim}} = 100$ simulations are conducted with varying parameters, from which 50 are used for training, 25 for validation, and 25 for testing. The training data is truncated after 1.5 s so that the models can be tested regarding their time extrapolation qualities.

For all three models, surrogates are created in accordance with the descriptions provided in Sections 4.1.1, 4.1.2 and 4.2.1. Specifically, a black-box model is employed to directly mimic the flow map in the form of an Multi Layer Perceptron (MLP), an LSTM is utilized for sequential modeling, and SINDy for system identification. The corresponding choice of hyperparameters is given in Table 4.1. Please note that the first two examples are governed by second-order ODEs. Hence, this structure is enforced in the identified dynamics by SINDy as well, by augmenting the states with their first order time-derivatives following

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{x}\\ \dot{\boldsymbol{x}}\end{bmatrix} = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{\mu}) \approx \begin{bmatrix}\dot{\boldsymbol{x}}\\ \boldsymbol{\Xi}\boldsymbol{\Theta}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{\mu})\end{bmatrix}. \tag{4.13}$$

Table 4.1: Comparison of hyperparameters for Neural Network, LSTM, and SINDy approaches.

Hyperparameter Category	**Black-box**	**Sequential**	**System Identification**
Model	MLP	LSTM	SINDy
Model Structure	Number of layers: 3, Neurons per layer: $128 \times 128 \times 128$	Number of layers: 3, Hidden units per layer: $64 \times 64 \times 64$	Library functions: polynomial functions up to degree of 2 with interactions
Trainable Parameters	Kart: 34,051 Occupant: 34,051 Simple ODE: 33,537	Kart: 83,907 Occupant: 83,907 Simple ODE: 83,521	Kart: 27 Occupant: 18 Simple ODE: 6
Training Regularization		Dropout rate: 0.2	ℓ_2-regularization parameter: 0.05, Threshold for STLSQ: 0.1
Optimization	Adam optimizer, Learning rate: 10^{-3}, Batch size: 32	Adam optimizer, Learning rate: 10^{-3}, Batch size: 32	STLSQ
Data Handling	Epochs: 2500, minmax scaling	Epochs: 10000	Numerical derivation of time derivatives
Output Control	Activation: selu, Loss: Mean Squared Error (MSE)	Activation: tanh and sigmoid, Loss: MSE	

For the performance comparison, a sample-wise normalized error

$$\bar{e}_{\boldsymbol{x}}(t, \boldsymbol{\mu}) \coloneqq \frac{\|\boldsymbol{x}(t, \boldsymbol{\mu}) - \tilde{\boldsymbol{x}}(t, \boldsymbol{\mu})\|_2}{\frac{1}{n_t} \sum_{i=1}^{n_t} \|\boldsymbol{x}(t_i, \boldsymbol{\mu})\|_2} \tag{4.14}$$

is used, where $\tilde{\boldsymbol{x}}(t, \boldsymbol{\mu})$ represents the state approximation of the surrogates. In addition, an overall relative error for the test dataset

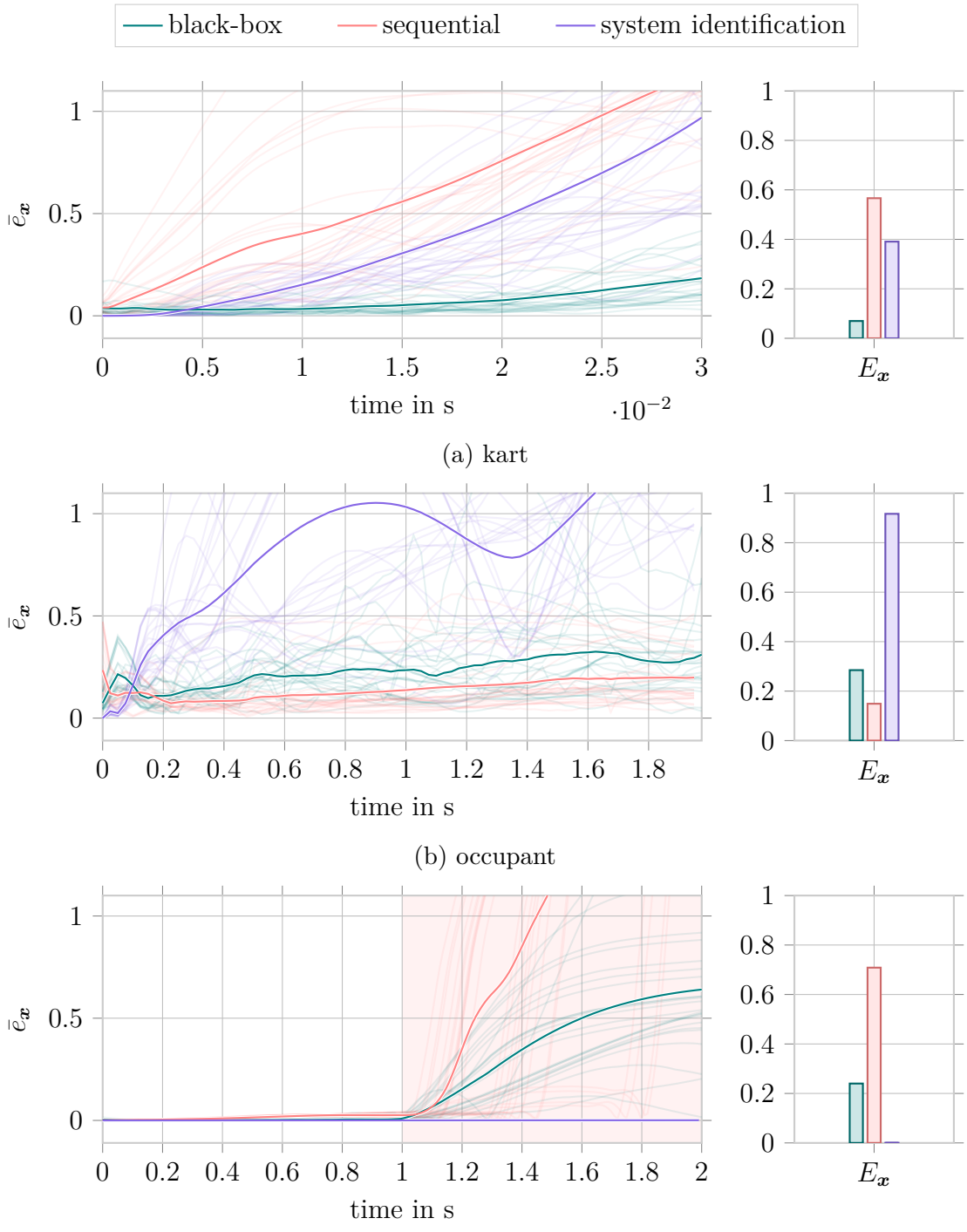
$$E_{\boldsymbol{x}}(\boldsymbol{X}, \tilde{\boldsymbol{X}}) \coloneqq \frac{\left\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\right\|_2}{\|\boldsymbol{X}\|_2} \tag{4.15}$$

serves as scalar performance measure. The results are presented in Fig. 4.3. It should be noted that the models are implemented for the sole purpose of comparing the different approaches under varying problem settings. Thus, they represent rather basic models without extensive hyperparameter optimization, as opposed to very sophisticated ones. Evidently, the individual approaches possess the capacity to be extended and optimized to yield superior results. However, the objective of this preliminary comparison is merely to provide a rudimentary comparison and more specialized approaches will be addressed in the course of this thesis.

As anticipated, system identification (SINDy) encounters challenges in its attempt to replicate the dynamic behavior of the kart crash simulation, resulting in a fast increasing error over time. In contrast, the black-box model capturing the flow aligns with the dynamics throughout the whole simulation time with a notable degree of accuracy. The sequential model (LSTM) demonstrates a comparable initial error but subsequently fails to capture the dynamics. This deficiency can be attributed to the input sequence for the kart consisting solely of constant parameters not providing any useful historical insights. Additionally, the hidden state sequences themselves are unable to induce information regarding the time point of the crash and the subsequent deformations.

The tides turn, considering the occupant example. The performance of the black-box model, which operates within the confines of a specific time frame without incorporating historical data, is found to be suboptimal. In contrast, the sequential model, which incorporates the information about the entirety of the data, yields the best outcome in this comparison. It is noteworthy that the system identification approach does not yield a practical model. This can be attributed to several factors. Second-order problems present a particular challenge, as second-order derivatives are often more prone to noise, and errors can propagate rapidly. Additionally, the very simple library employed for model creation is evidently insufficient in terms of expressiveness. However, more sophisticated libraries led to unstable dynamics. It should be noted that, in theory, the stabilization of the system through the incorporation of a stable higher-order polynomial term is possible. However, this is not the focus of the present investigation.

In the case of the simple ODE example, both the black-box and sequential approaches fail to extrapolate over time. While they demonstrate exceptional performance within the



(a) kart

(b) occupant

(c) simple ODE (time extrapolation highlighted as red background)

Figure 4.3: Normalized error for the displacements of one node over time (left) and overall error (right) for the kart model (a), the occupant model (b), and the simple ODE example (c). The individual test trajectories are drawn transparently, while the mean value of all test simulations is shown opaquely.

training regime, they entirely fail to capture data beyond the observed range, leading to a dramatic increase in error. In contrast, the system identification approach effectively addresses this challenge by successfully identifying the underlying governing equations. This highlights scenarios in which system identification outperforms conventional data-driven methods, demonstrating its robustness and predictive capabilities beyond the training domain.

It is evident that the presented examples are deliberately designed to expose the limitations of the aforementioned numerical approaches, demonstrating their inability to effectively address all scenarios. However, this is the whole intense of the presented results: varying problem settings necessitate diverse approaches, and numerous methodologies exist for addressing these problems. This shows that capturing the dynamics themselves is already a challenging endeavour. Nonetheless, the subsequent sections of this thesis will demonstrate the feasibility of achieving this, even for high-dimensional systems, across a range of scenarios.

Part II

Latent Approximation of Dynamics

Chapter 5

Learning in Reduced Coordinates: Model Reduction Meets Machine Learning

Es una obsesión.

Aventura ft. Judy Santos, Obsesión

We have previously explored two fundamental pillars of data-driven surrogate modeling: the efficient identification of low-dimensional coordinates for high-dimensional systems, and the approximation of system dynamics purely from data. In the following, these two components are blended together to construct surrogate models that are both efficient and accurate in capturing the behavior of high-dimensional structural dynamical systems.

It is important to note that we consider scenarios in which the High-Fidelity (HF) models used for data generation are embedded within commercial simulation software, which conceals their internal structure and renders the system operators inaccessible to the user. As a result, the proposed subsequent methods are placed in a context where they must address the challenges outlined in Challs. 1, 2 and 4, namely, managing high dimensionality, identifying appropriate coordinate systems for representing the system's dynamics, and operating without access to the underlying physical models. While the methods presented in this chapter focus on these challenges, subsequent chapters will focus on tailoring specialized approaches to address additional emerging challenges.

At the core of all methodologies discussed here lies the shared principle of (i) discovering a low-dimensional representation of the system states and (ii) approximating the corresponding dynamics in this latent space. We distinguish between two overarching strategies for modeling these dynamics: black-box modeling and system identification. In the current

chapters of Part II, we focus on black-box approaches for learning latent dynamics, which we collectively refer to as Latent Approximation of Dynamics (LADy). These methods typically model the latent dynamics by learning the mapping from some input variables to the latent state representation. In contrast, the chapters in Part III address the explicit derivation of interpretable, analytic equations for the latent dynamics, which we denote as Latent Discovery of Dynamics (LDD).

Early efforts that can be placed within the LADy framework leveraged linear dimensionality reduction techniques in the form of Principal Component Analysis (PCA), coupled with cubic spline interpolation to capture parametric variations [BuiThanhDamodaranWillcox03]. However, with the increasing availability of data and the rapid advancement of machine learning, the field has experienced substantial growth over the past decade. Influential representative examples of this evolution can be found in [HesthavenUbbiali18, WangHesthavenRay19], where PCA is used to extract a low-dimensional subspace, and neural networks are employed to model the reduced dynamics of parameterized Partial Differential Equations (PDEs). Extensions of this concept to time-dependent problems have also been explored, for instance in [GuoHesthaven19]. Other works, such as [Le GuennecEtAl18], rely on regression methods in combination with linear reduction via the CUR Decomposition (CUR) decomposition.

Nonlinear dimensionality reduction methods have also gained significant attention. For example, Kernel Principal Component Analysis (KPCA) combined with neural networks as used in [SalvadorDedèManzoni21]. Moreover, Autoencoders (AEs) are widely applied in both intrusive and non-intrusive settings: they are used to construct Reduced Order Models (ROMs) from governing equations [LeeCarlberg20], and in purely data-driven contexts to learn latent dynamics via neural networks [FrescaDedeManzoni21]. Recent works, such as [RegazzoniEtAl24], further extend this idea by using latent dynamics models with neural Ordinary Differential Equations (ODEs). Numerous hybrid approaches have also emerged, combining various linear and nonlinear reduction techniques. For example, autoencoders have been used to complement linear subspaces [ShenEtAl21], or stacked in a two-stage reduction process, where a first-stage linear reduction simplifies the system to an intermediate level, followed by further nonlinear refinement via a second reduction method. Examples of such approaches can be found in [FrescaManzoni22, ContiEtAl23b, ContiEtAl24, RettbergEtAl24].

In the field of structural dynamics, these techniques have been successfully applied in several application domains. For example, in the modeling of Micro-Electromechanical Systems (MEMs) systems [FrescaManzoni22, ContiEtAl23b], and in crash simulation scenarios [Le GuennecEtAl18, KneiflGrunertFehr21, KneiflHayFehr22b, KapsCzechDuddeck22], where traditional modeling techniques are often too computationally expensive for real-time or iterative use.

The existing literature demonstrates that surrogate models are generally effective at approximating displacement fields, i.e. the spatial configuration of structural systems. However, other physically relevant quantities such as stress have often been neglected or only estimated in a post-processing step using conventional finite element tools, as seen for example in [FrescaManzoni22]. Although such surrogate modeling techniques are frequently presented as universally applicable, in practice, significant differences and specific challenges arise depending on the physical quantity being approximated.

To address this gap, we previously conducted a detailed investigation into the approximation of both displacement and stress in a continuum-mechanical musculoskeletal model of the human upper arm [KneiflEtAl23]. However, that study was restricted to quasi-static system responses, and the methods were not extended for dynamic scenarios. Consequently, the kart crash example introduced in Section 2.2.1 serves as the benchmark scenario for the upcoming surrogate modeling tasks. Here, we aim to investigate the performance of surrogates to capture the kart's dynamic response during a crash event not only for its deformation field but also it's internal stress distributions.

In addition, most existing studies focus on specific combinations of a single dimensionality reduction method with a single regression technique. This overlooks the wide range of possible method combinations and their potentially differing suitability for specific quantities and applications. Therefore, one of the key goals of this investigation is to evaluate which combinations of reduction and regression methods are most appropriate for various types of physical quantities and structural dynamics use cases.

Through this study, we seek not only to demonstrate the applicability of surrogate models in structural dynamics but also to provide a broad algorithmic comparison along with practical design guidance. In doing so, we aim to support and inform future research in this field. Specifically, we investigate the strengths and limitations of both linear and nonlinear dimensionality reduction techniques in their ability to approximate different physical quantities in dynamic structural systems. The dimensionality reduction methods under consideration include the CUR decomposition, the widely used PCA, its nonlinear extension, KPCA, and fully-connected AEs as a further nonlinear alternative.

In one of our previous publications [KneiflGrunertFehr21], we conducted a comparative study of various regression algorithms for their suitability within the LADy framework, in combination with PCA, applied to a simplified version of the kart crash scenario considered here. It is important to note that the approach proposed in that earlier work differs fundamentally from the methodology presented in this chapter: it followed an autoregressive scheme, where each state is predicted based on the previous one. In contrast, the methodology introduced here directly captures the explicit dependency on time, enabling predictions at arbitrary time instances without requiring the full temporal history. This formulation increases flexibility and efficiency, especially in applications requiring sparse or selectively sampled time evaluations and for scenarios with constant

simulation parameters. For time-varying parameter sequences, however, autoregressive approaches can be beneficial as discussed in the second part of this chapter.

Despite the difference in modeling strategy, the earlier study identified both Gaussian Processes (GPs) and Neural Networks (NNs) as particularly promising regression models. Consequently, these two model types are also employed in the present work. Each dimensionality reduction technique considered is thus combined with two regression variants: Multi Layer Perceptrons (MLPs), as representative of neural network-based models, and GPs, known for their flexibility and uncertainty quantification capabilities. This results in a diverse set of surrogate modeling pipelines.

The content of this chapter focuses on two key aspects: First, the investigation aims to explore the synergies between the mentioned dimensionality reduction and regression methods in order to identify combinations that demonstrate particularly favorable performance in terms of data requirements, reduction capabilities, accuracy, and computational efficiency. This comparative study provides insight into the strengths and limitations of each approach within the LADy framework. Second, building on these insights, the chapter proceeds to examine surrogate modeling for scenarios with time-dependent input signals, which introduces additional complexity due to the temporal structure of the inputs. For this purpose, the human body model representing an occupant in a pre-crash scenario, introduced in Section 2.2.2, serves as an elaborate and realistic numerical example. To effectively handle such sequential data within the surrogate modeling pipeline, we incorporate Recurrent Neural Networks (RNNs), which are well-suited for capturing temporal dependencies.

The main contributions of this investigation can be summarized as follows:

1. We apply the LADy framework to the often overlooked yet physically significant quantity of stress, providing a detailed analysis of its strengths and limitations in this context.

2. We investigate and compare a diverse set of dimensionality reduction techniques, including CUR and KPCA, rarely employed alternatives to PCA and autoencoders, all embedded within the LADy framework. This enables a comprehensive assessment across multiple reduction strategies.

3. We offer practical guidance for selecting appropriate reduction methods, grounded in their accuracy and computational requirements for different physical quantities.

4. We develop surrogate models capable of handling sequential input data within the LADy framework to accommodate dynamically evolving system behavior.

In the following sections, we first detail the procedure for combining dimensionality reduction with dynamic approximation under constant parameter inputs. A range of

method combinations is systematically evaluated and applied to the kart example, enabling a rigorous comparative study. Subsequently, we present an approach for modeling systems with sequential inputs, as exemplified by the occupant scenario.

5.1 Surrogate Modeling Approach

As this is the first section in this thesis in which a complete surrogate modeling framework is described, we take this opportunity to thoroughly examine the entire process chain in detail. This section provides a comprehensive, one-time explanation of all steps involved, laying a clear foundation for the approaches presented in the following chapters. The whole workflow is prescribed by three major steps visually sketched in Fig. 5.1. The first two steps build the offline phase, while the third represents the online phase. Step I represents the computational expensive data-generation, where high-fidelity Finite Element (FE) simulations are conducted to gather simulation data. In Step II, the surrogate model itself is defined and trained. As the final Step III, the surrogate is evaluated and its predictions are validated against theory or constraints. Alternatively, the surrogate can be directly deployed within an application, such as optimization loops, control frameworks, or real-time simulations, where its performance can be assessed based on task-specific criteria.

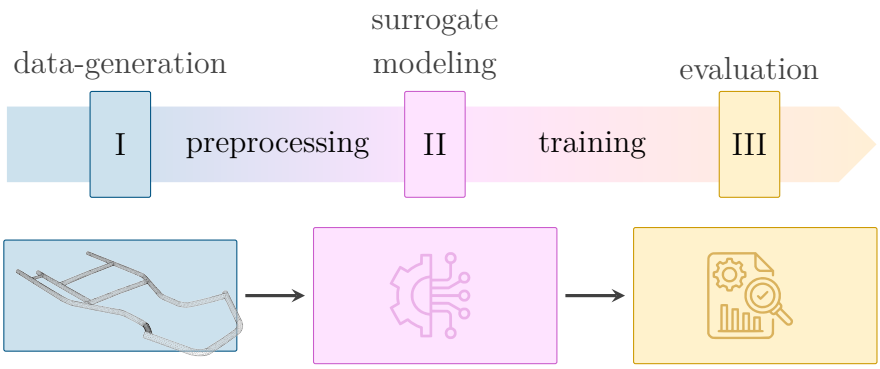


Figure 5.1: Schematic overview of the three steps comprising the presented surrogate modeling pipeline.

In order to obtain a dataset that captures a broad spectrum of the system's behavior, the high-fidelity model is parameterized, and multiple simulations are carried out using the FE software. To ensure comprehensive coverage of the parameter space, a diverse set of simulation parameter combinations is sampled grounded on expert knowledge or (quasi-random) sampling schemes. The resulting simulation outputs—i.e. the FE simulation results—are treated as the reference data and contain detailed information about both nodal quantities, such as displacements, and element quantities, such as stress fields. Subsequently, the raw simulation results are extracted and preprocessed, which

may include operations such as filtering, or normalization, depending on the quality and resolution of the data. All individual simulation trajectories, spanning the full set of nodes or elements, are then organized into a snapshot matrix, which serves as the foundational input for the surrogate modeling step.

The second step involves the formulation of the surrogate model itself. As previously mentioned, the surrogate modeling approach is built upon two fundamental parts. The first part entails identifying a suitable coordinate transformation $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{Z}$ that maps the high-dimensional system state x to its low-dimensional counterpart $\boldsymbol{z} \in \mathcal{Z} \subseteq \mathbb{R}^r$, with $r \ll n$. A corresponding back transformation $\boldsymbol{\psi} : \mathcal{Z} \to \mathcal{X}$ enables the recovery of full-order states from the reduced representation, thereby ensuring the physical interpretability of the results. This step is explained in detail in Section 3.2.

The second part involves modeling the system's behavior within the reduced space, i.e. modeling the temporal evolution of the reduced coordinates $\boldsymbol{z}$. For this purpose, we define a function $\boldsymbol{\theta} : \mathcal{T} \times \mathcal{P} \to \mathcal{Z}$ that approximates the latent dynamics via $\boldsymbol{z} \approx \tilde{\boldsymbol{z}} = \boldsymbol{\theta}(t, \boldsymbol{\mu})$, capturing the dependency on time t and parameters $\boldsymbol{\mu} \in \mathcal{P}$. This follows the methodology outlined in Section 4.1.1. It is essential to note that beyond predictive accuracy, a primary goal of the surrogate model is computational efficiency. Accordingly, the computational costs of the surrogate $\tilde{\boldsymbol{F}}(t, \boldsymbol{\mu})$, denoted by $\mathcal{J}(\tilde{\boldsymbol{F}}(t, \boldsymbol{\mu}))$, should be significantly lower than that of the full-order model $\boldsymbol{F}(t, \boldsymbol{\mu})$, $\mathcal{J}(\boldsymbol{F}(t, \boldsymbol{\mu}))$, thereby enabling efficient deployment in time- or resource-constrained environments.

In the following, dimensionality reduction and dynamics approximation are treated as two separate subproblems, addressed in a decoupled fashion rather than through a joint optimization loop. It is important to note that in the case of autoencoders, a simultaneous training of the autoencoder and the dynamics approximation model can offer notable benefits, as discussed in [FrescaDedeManzoni21, Remark 1]. This potential advantage is leveraged in later chapters of this thesis, where certain approaches significantly benefit from end-to-end training. However, for the purposes of the present comparison, we refrain from implementing joint optimization—even for the autoencoder-based surrogates. This decision ensures fairness across all methods, since joint optimization is not feasible for the other dimensionality reduction techniques considered (e.g., PCA, KPCA, or CUR). By keeping the optimization process consistent across all method combinations, a balanced and meaningful comparison of their respective performances is ensured.

The composition of the trained reconstruction mapping $\boldsymbol{\psi}$ with the learned regression model $\boldsymbol{\theta}$ yields the complete surrogate model $\tilde{\boldsymbol{F}}(t, \boldsymbol{\mu})$, formally defined as

$$\tilde{\boldsymbol{x}}(t, \boldsymbol{\mu}) = \tilde{\boldsymbol{F}}(t, \boldsymbol{\mu}) = \boldsymbol{\psi} \circ \boldsymbol{\theta}(t, \boldsymbol{\mu}).$$

Accordingly, the offline training phase consists of the following subsequent steps

1. Fitting the dimensionality reduction, as described in Section 3.2, yielding the reduced coordinates $\boldsymbol{z} = \boldsymbol{\phi}(\boldsymbol{x})$,
2. Construction of the regression dataset as outlined in Section 4.1.1, using the reduced system states in the latent space,
3. Training of the regression model to approximate the system dynamics in the reduced space.

Once the offline phase is finished, the surrogate can be evaluated in the online phase comprising Step III to efficiently predict system behavior. Specifically, for a new set of (unseen) simulation parameters $\boldsymbol{\mu}$, the regression model provides an approximation of the reduced state trajectory, which is then mapped back to the high-dimensional physical space via the reconstruction mapping $\boldsymbol{\psi}$. Throughout the following investigations in this chapter, we consider scenarios in which all simulations begin from the same zero initial condition. Therefore, the dependency on the initial condition of the surrogate model is omitted without loss of generality. Nevertheless, it is worth pointing out that the initial condition can be incorporated into the parameter vector $\boldsymbol{\mu}$ if needed, enabling the model to accommodate varying initial conditions within the same framework.

Taking all aforementioned considerations into account, the overall optimization goal for constructing a surrogate model, originally introduced in (5.1), can now be refined to reflect the specific surrogate architecture presented. This leads to the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{\phi},\boldsymbol{\psi},\boldsymbol{\theta}} \quad & \frac{1}{n_t n_{\text{sim}}} \sum_{t\in\mathbb{T}} \sum_{\boldsymbol{\mu}\in\mathbb{P}} L(\boldsymbol{x}(t,\boldsymbol{\mu}), \tilde{\boldsymbol{x}}(t,\boldsymbol{\mu})) \\ \text{s.t.} \quad & \boldsymbol{x}(t,\boldsymbol{\mu}) = \boldsymbol{F}(t,\boldsymbol{\mu}) \\ & \tilde{\boldsymbol{x}}(t,\boldsymbol{\mu}) = \tilde{\boldsymbol{F}}(t,\boldsymbol{\mu}) = \boldsymbol{\psi}(\tilde{\boldsymbol{z}}(t,\boldsymbol{\mu})) \\ & \tilde{\boldsymbol{z}}(t,\boldsymbol{\mu}) = \boldsymbol{\theta}(t,\boldsymbol{\mu}) \\ & \mathcal{J}(\tilde{\boldsymbol{F}}(t,\boldsymbol{\mu})) \ll \mathcal{J}(\boldsymbol{F}(t,\boldsymbol{\mu})). \end{aligned} \tag{5.1}$$

Recall that the function L denotes a chosen loss metric that quantifies the discrepancy between the surrogate-predicted states $\tilde{\boldsymbol{x}}$ and the reference states $\boldsymbol{x}$ from the HF model. The optimization is carried out over the coordinate transformation components $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$, which define the reduced latent space, as well as over the regression model $\boldsymbol{\theta}$, which captures the system dynamics in that latent space.

By incorporating simulation parameters $\boldsymbol{\mu} \in \mathbb{P}$ and sampling across the time domain $\mathbb{T}$, the model aims to achieve generalizable and accurate predictions throughout the entire parameter-time space. Moreover, an explicit constraint on computational cost ensures

that the surrogate model $\tilde{\boldsymbol{F}} = \tilde{\boldsymbol{F}}$ is not only accurate but also substantially more efficient than evaluating the original model $\boldsymbol{F}$. An overview of the concrete pipeline is provided in Fig. 5.2.

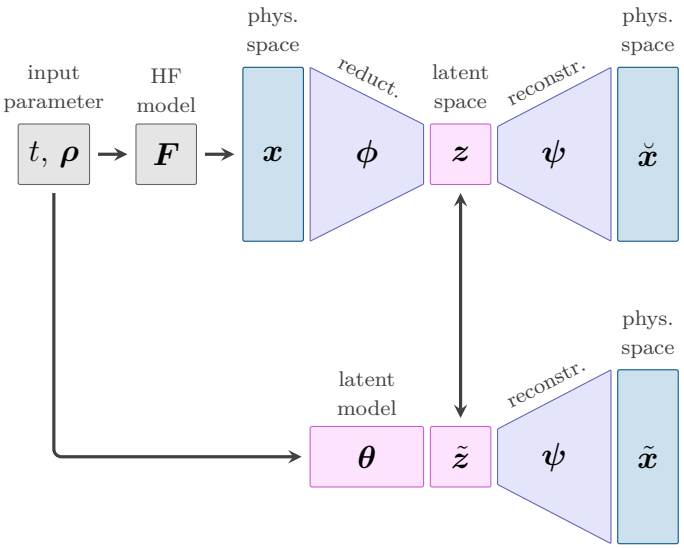


Figure 5.2: Schematic overview of the surrogate modeling pipeline combining dimensionality reduction and regression.

5.1.1 Error Measures

Before validating the surrogate models, we introduce error measures used to assess the performance of the proposed methods. Specifically, we employ sample-wise normalized relative errors, weighted by the inverse of the mean state magnitude within each individual simulation. This weighting is performed for all states associated with a given parameter instance $\boldsymbol{\mu}$ and over the corresponding time series $\mathbb{T}$. Normalized error quantities are denoted with an overbar, i.e. $\bar{e}$. Although some of these metrics have been presented earlier in this work, they are reiterated here for completeness and clarity.

In particular, the *reconstruction error* on the state space

$$\bar{e}_{\breve{\boldsymbol{x}}}(t,\boldsymbol{\mu}) \coloneqq \frac{\|\boldsymbol{x}(t,\boldsymbol{\mu}) - \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{x}(t,\boldsymbol{\mu})))\|_2}{\frac{1}{n_t}\sum_{i=1}^{n_t}\|\boldsymbol{x}(t_i,\boldsymbol{\mu})\|_2} = \frac{\|\boldsymbol{x}(t,\boldsymbol{\mu}) - \breve{\boldsymbol{x}}(t,\boldsymbol{\mu})\|_2}{\frac{1}{n_t}\sum_{i=1}^{n_t}\|\boldsymbol{x}(t_i,\boldsymbol{\mu})\|_2} \tag{5.2}$$

serves as indicator for the ability of the reduction algorithms to embed a state in the latent space with marginal information loss. The *approximation error* on the state space

$$\bar{e}_{\boldsymbol{x}}(t,\boldsymbol{\mu}) \coloneqq \frac{\|\boldsymbol{x}(t,\boldsymbol{\mu}) - \boldsymbol{\psi}(\tilde{\boldsymbol{z}}(t,\boldsymbol{\mu}))\|_2}{\frac{1}{n_t}\sum_{i=1}^{n_t}\|\boldsymbol{x}(t_i,\boldsymbol{\mu})\|_2} = \frac{\|\boldsymbol{x}(t,\boldsymbol{\mu}) - \tilde{\boldsymbol{x}}(t,\boldsymbol{\mu})\|_2}{\frac{1}{n_t}\sum_{i=1}^{n_t}\|\boldsymbol{x}(t_i,\boldsymbol{\mu})\|_2} \tag{5.3}$$

tells us how performant a surrogate model is to approximate the time evolution of full system states, while the *latent approximation error*

$$\bar{e}_{\boldsymbol{z}}(t,\boldsymbol{\mu}) \coloneqq \frac{\|\boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x}(t,\boldsymbol{\mu})) - \tilde{\boldsymbol{z}}\|_2}{\frac{1}{n_t}\sum_{i=1}^{n_t}\|\boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x}(t_i,\boldsymbol{\mu}))\|_2} = \frac{\|\boldsymbol{z}(t,\boldsymbol{\mu}) - \tilde{\boldsymbol{z}}(t,\boldsymbol{\mu})\|_2}{\frac{1}{n_t}\sum_{i=1}^{n_t}\|\boldsymbol{z}(t_i,\boldsymbol{\mu})\|_2} \tag{5.4}$$

does the same but on the latent level.

In addition to these sample-wise evaluated errors, scalar relative error quantities

$$E_{\breve{\boldsymbol{x}}} \coloneqq \frac{\left\|\boldsymbol{X} - \breve{\boldsymbol{X}}\right\|_2}{\|\boldsymbol{X}\|_2},\ E_{\boldsymbol{x}} \coloneqq \frac{\left\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\right\|_2}{\|\boldsymbol{X}\|_2},\ \text{and}\ E_{\boldsymbol{z}} \coloneqq \frac{\left\|\boldsymbol{Z} - \tilde{\boldsymbol{Z}}\right\|_2}{\|\boldsymbol{Z}\|_2} \tag{5.5}$$

are used to validate the performance of entire simulations or data sets at once. Please recall that $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_s}]$ is the snapshot matrix, $\breve{\boldsymbol{X}} = [\breve{\boldsymbol{x}}_1, \ldots, \breve{\boldsymbol{x}}_{n_s}] = [\boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{x}_1)), \ldots, \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{x}_{n_s}))]$ its reconstruction, $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_{n_s}] = [\boldsymbol{\psi}(\tilde{\boldsymbol{z}}_1), \ldots, \boldsymbol{\psi}(\tilde{\boldsymbol{z}}_{n_s})]$ the matrix containing the corresponding approximation of a surrogate model, $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_s}] = [\boldsymbol{\phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_{n_s})]$ the reduced states and $\tilde{\boldsymbol{Z}} = [\tilde{\boldsymbol{z}}_1, \ldots, \tilde{\boldsymbol{z}}_{n_s}]$ the corresponding approximations in the latent space.

Moreover, we introduce errors in the physical domain to provide more tangible measures. In particular, the mean Euclidean distance

$$e_{\boldsymbol{d}}^{\mathrm{mean}}(t,\boldsymbol{\mu}) \coloneqq \frac{1}{n_{\mathrm{node}}}\sum_{j=1}^{n_{\mathrm{node}}}\left\|[\boldsymbol{d}(t,\boldsymbol{\mu})]_{j:} - \left[\tilde{\boldsymbol{d}}(t,\boldsymbol{\mu})\right]_{j:}\right\|_2 \tag{5.6}$$

and the maximum Euclidean distance

$$e_{\boldsymbol{d}}^{\max}(t,\boldsymbol{\mu}) \coloneqq \max_{j\in\{1,\ldots n_{\mathrm{node}}\}}\left\|[\boldsymbol{d}(t,\boldsymbol{\mu})]_{j:} - \left[\tilde{\boldsymbol{d}}(t,\boldsymbol{\mu})\right]_{j:}\right\|_2 \tag{5.7}$$

occurring among all nodes between the reference FE simulation and their approximation are utilized. In that context, $[\boldsymbol{d}(t,\boldsymbol{\mu})]_{j:} \in \mathbb{R}^3$ represents the displacement of the j-th node of the FE model at time t and for parameter $\boldsymbol{\mu}$ and $\left[\tilde{\boldsymbol{d}}(t,\boldsymbol{\mu})\right]_{j:} \in \mathbb{R}^3$ the corresponding prediction of a surrogate model. Additionally, the corresponding averaged values over time and all test simulations

$$E_{\boldsymbol{d}}^{\mathrm{mean}} \coloneqq \frac{1}{n_{\mathrm{sim}} n_t}\sum_{\boldsymbol{\mu}\in\mathbb{P}}\sum_{t\in\mathbb{T}} e_{\boldsymbol{d}}^{\mathrm{mean}}(t,\boldsymbol{\mu}),\quad E_{\boldsymbol{d}}^{\max} \coloneqq \frac{1}{n_{\mathrm{sim}} n_t}\sum_{\boldsymbol{\mu}\in\mathbb{P}}\sum_{t\in\mathbb{T}} e_{\boldsymbol{d}}^{\max}(t,\boldsymbol{\mu}) \tag{5.8}$$

are used to represent the approximation quality for the complete test data. A similar error measure is used for stress

$$E_{\boldsymbol{\sigma}}^{\mathrm{mean}} \coloneqq \frac{1}{n_{\mathrm{sim}} n_t n_{\mathrm{elem}}}\sum_{\boldsymbol{\mu}\in\mathbb{P}}\sum_{t\in\mathbb{T}}\sum_{j=1}^{n_{\mathrm{elem}}}\left\|[\boldsymbol{\sigma}(t,\boldsymbol{\mu})]_j - [\tilde{\boldsymbol{\sigma}}(t,\boldsymbol{\mu})]_j\right\|_2, \tag{5.9}$$

where $[\boldsymbol{\sigma}(t,\boldsymbol{\mu})]_j$ and $[\tilde{\boldsymbol{\sigma}}(t,\boldsymbol{\mu})]_j$ denote the reference and predicted stress values of the j-th element at time t and for parameter $\boldsymbol{\mu}$, respectively.

5.1.2 Results and Discussion

In this section, we present the results concerning the approximation quality of the full-field displacement and stress responses for the kart example. To this end, multiple surrogate models are constructed based on the methodology described in Section 5.1. Each surrogate is denoted by the specific combination of dimensionality reduction and regression algorithms used. For instance, a surrogate employing PCA for reduction and a NN for learning the dynamic parameter dependencies is referred to as a PCANN surrogate. The hyperparameter settings used to configure and train the various models are summarized in Table 5.1.

Table 5.1: Model architectures, hyperparameters, and data properties.

Category	**Hyperparameter**	**Details**
All NNs	Optimizer	Adam, Learning Rate: $1e-3$
	Training	Epochs: 1000, Batchsize: 128
	Activation fcn.	Elu
	Output act. fcn.	Linear
	Regularization	Select best weights w.r.t. validation data
PCA	-	
CUR	Sampling strategy	Random
	Error parameter ϵ	0.4
KPCA	Kernel	Polynomial $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^\intercal \boldsymbol{x}_j + 350)^3$
	Data preprocessing	Centering
AE	Encoder layer size	$n \times 300 \times 150 \times 75 \times r$
	Decoder layer size	$r \times 75 \times 150 \times 300 \times n$
	Data preprocessing	Normalization
MLP	Layer size	$(1 + n_\mu) \times 64 \times 64 \times 64 \times r$
	Data preprocessing	Standardization
GP	Kernel	Matern
	Smoothness parameter	2.5
	Data preprocessing	Normalization
Data	Train-Test split	$n_{\text{sim}}^{\text{train}} = 96$, $n_{\text{sim}}^{\text{test}} = 48$, $n_{\text{sim}}^{\text{val}} = 48$
	Timesteps and time	$n_t = 102$, $t_{\text{end}} = 0.03\,\text{s}$

Data Requirements In the field of model reduction, a distinction can be made between two principal application types. On the one hand, there are scenarios where the offline phase—involving data generation and preprocessing—is time- and resource-critical. On the other hand, there are applications primarily focused on reducing computational effort during the online phase, where evaluation speed is paramount. For the former case, minimizing the number of full-order simulations is essential. This raises an important question: how much training data is sufficient before additional simulations yield diminishing returns in terms of predictive performance?

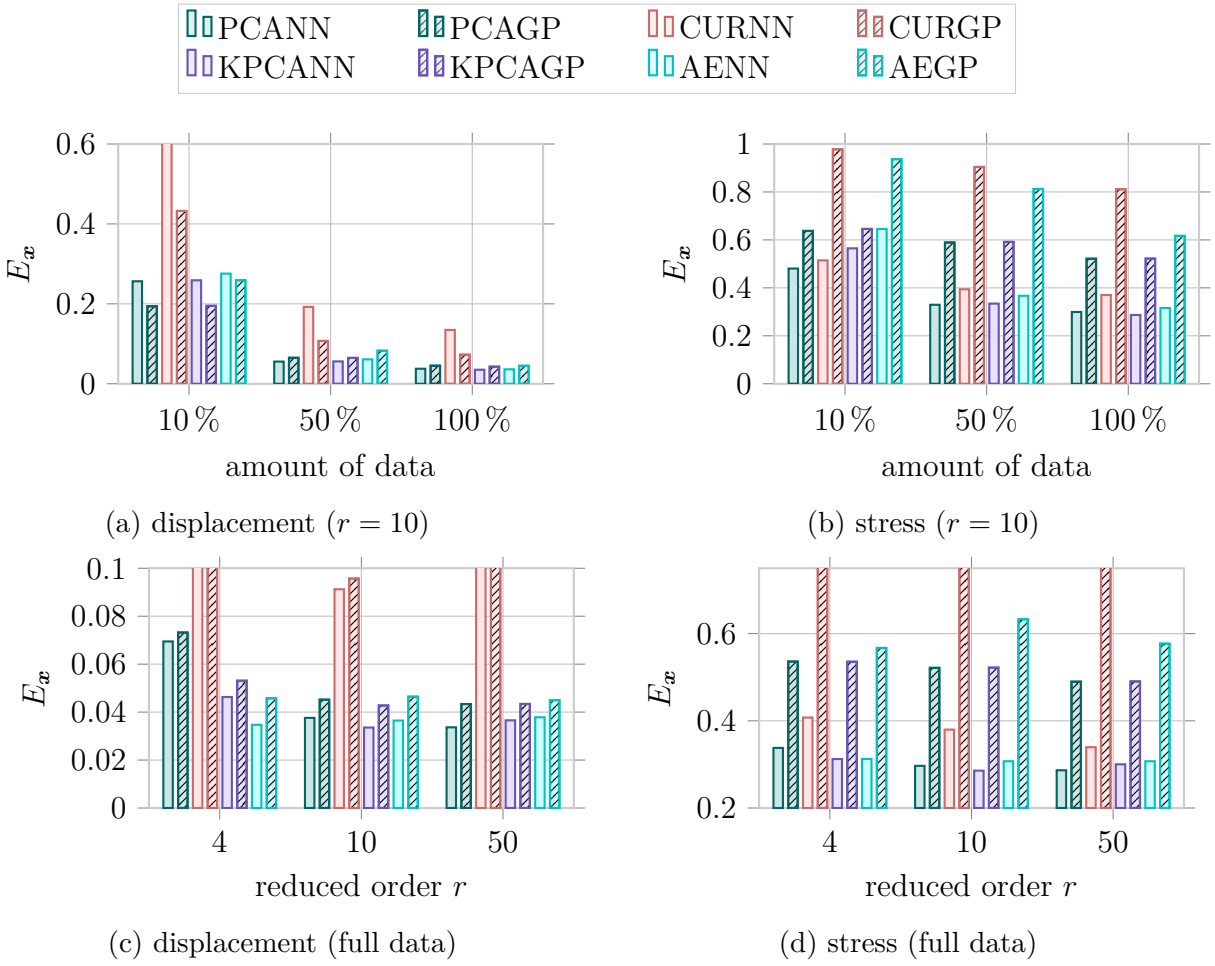
To explore the impact of training data availability, we conduct a series of experiments using surrogate models trained on varying numbers of snapshots, i.e. full simulation results. Specifically, we compare the performance of the surrogate models across three different data regimes: using the full training dataset comprising 96 simulations, a reduced set of 50 % corresponding to 48 simulations, and a low-data scenario with only 10 % of the original dataset, amounting to 9 simulations. The corresponding results for the state approximation using a latent dimension of $r = 10$ are presented in Fig. 5.3a and Fig. 5.3b for displacement and stress data, respectively.

As expected, increasing the amount of training data leads to improved approximation quality across nearly all surrogate model configurations. When sufficient data is available, all surrogates achieve satisfying accuracy in approximating the displacement field. However, the stress data presents a greater challenge, with all surrogate models exhibiting higher errors—even when trained on the full dataset.

GPs are often regarded as particularly effective in low-data regimes, a trend that is also reflected in the displacement results using only 10 % of the training data. In this setting, surrogate models employing GPs consistently outperform those based on NNs. However, this advantage vanishes quickly: already at 50 % data availability, NN-based surrogates yield superior results. Moreover, this benefit of GPs does not extend to the stress predictions in the current investigation. Instead, a consistent pattern emerges across most data availability scenarios in which NNs outperform GPs, resulting in lower approximation errors.

These findings indicate that GPs, despite their theoretical appeal for data-scarce settings, struggle to capture the complexity of the latent space dynamics encountered here, whereas NNs demonstrate greater flexibility and expressiveness in modeling such behavior. Moreover, in the very low data regime, a slight trend can be observed indicating that autoencoder-based surrogates tend to struggle somewhat more with accurately capturing a low-dimensional latent representation

More generally, the transition from 10,% to 50,% of the training data has a substantial positive impact on surrogate performance, whereas the additional step from 50,% to 100,% yields only marginal further improvements. This indicates a diminishing return on



(a) displacement ($r = 10$)

(b) stress ($r = 10$)

(c) displacement (full data)

(d) stress (full data)

Figure 5.3: Normalized errors for varying amounts of used training data ((a) and (b)) and varying reduced orders ((c) and (d)) for the displacements and the stress. The results represent the mean performance of three independently trained surrogate models for each combination of settings, in order to account for effects due to random initialization and stochastic variability during training.

additional data beyond a certain threshold. Among the different dimensionality reduction techniques employed, CUR decomposition consistently exhibits the worst performance, highlighting its limitations in preserving the essential features of complex physical quantities such as stress within the reduced-order models. For the succeeding results, the full training dataset is utilized.

Reduced Order One of the most crucial hyperparameters in ROMs is the latent dimensionality r. Reducing a system to the minimum can especially pay off in scenarios, where data must be transferred with a limited bandwidth or where storage is heavily limited. At the same time, the computational efficiency of the ROM usually scales with its dimensionality. Consequently, one often aims to reduce a system as much as possible.

This, however, usually comes at the cost of accuracy. Hence, in the following experiment, we create surrogate models for varying reduced orders in the set of $r \in \{4, 10, 50\}$.

The results can be seen in Fig. 5.3c for the displacements and in Fig. 5.3d for the stress data. As expected, the lowest latent dimension results in the highest approximation error for the displacement field, and for most surrogate models, also for the stress field—highlighting the fundamental trade-off between model compactness and representational capacity. Notably, surrogate models that employ nonlinear reduction techniques—such as KPCA and AE—demonstrate superior performance at low latent dimensions, as they are better able to capture complex system behavior within a more constrained space. In contrast, PCA-based surrogates begin to catch up at higher latent dimensions, eventually achieving comparable results as the dimensionality increases. In comparison, surrogate models based on CUR decomposition consistently yield substantially poorer results across all considered latent dimensions, underlining its limitations for this application. As before, surrogates that use neural networks to model the latent dynamics generally outperform those based on GPs, exhibiting a slight but consistent advantage for displacements and a pronounced performance benefit when approximating stress fields.

Counterintuitively, for almost all surrogate models, the performance does not significantly improve when increasing the latent dimension from $r = 10$ to $r = 50$. In contrast to the plain reconstruction error presented in Section 3.3, a higher latent dimension does consequently not necessarily lead to a notably better performance. What sounds surprising first, appears logic when one considers the complex interplay between dimensionality reduction and latent dynamics approximation.

To illustrate this effect, we break down the overall approximation error $E_{\boldsymbol{x}}$ into its sources, i.e. the reconstruction error $E_{\check{\boldsymbol{x}}}$ and the error in approximating the latent states $E_{\boldsymbol{z}}$ for the extreme cases of $r = 4$ and $r = 50$. The corresponding results are visualized in Fig. 5.4. The reconstruction error provides a natural barrier how close one can get in approximating the reference values. In alignment to the previous results from Section 3.3, the nonlinear reduction methods, KPCA and AE, particularly outperform their linear counterparts for low latent dimensions with respect to the reconstruction $E_{\check{\boldsymbol{x}}}$ error. For the high latent dimension, the reconstruction error almost becomes neglectable compared to the overall approximation performance for the displacements. However, at the same time, the error in capturing the latent state $\boldsymbol{z}$ increases significantly across all methods. A similar trend is observed for the stress approximations: While the reconstruction improves with increasing latent dimensionality, the accuracy of the latent dynamics approximation deteriorates significantly.

This can be explained by the fact that the complexity of the latent behavior usually grows with the dimension. Take for example PCA. The first latent coordinates describe the most dominant effects occurring in the data, while the later ones represent only marginal changes that take place in the data. Simultaneously, the higher order latent states are

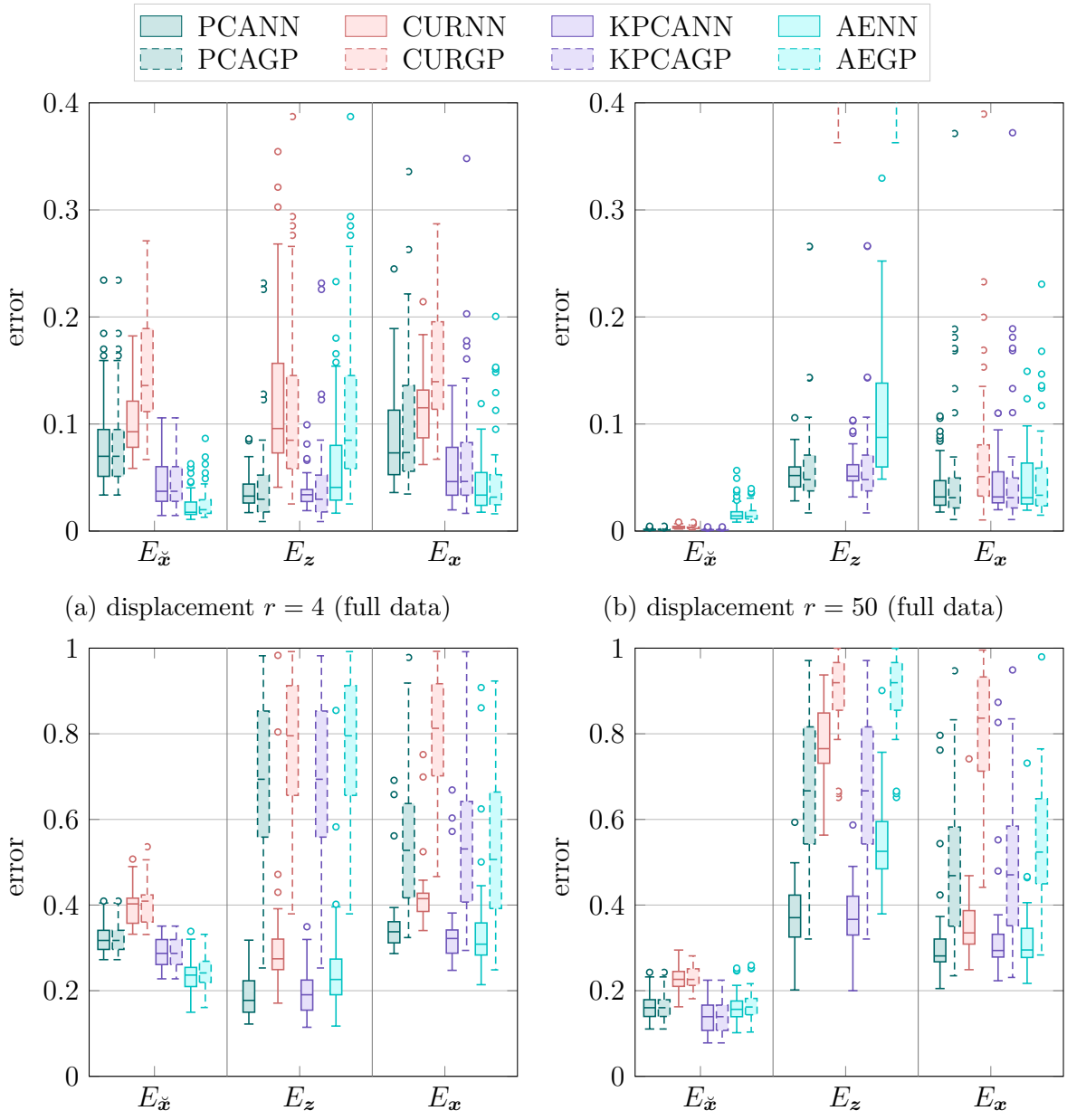
of a smaller magnitude and correspondingly implicitly weighted to a lesser extent in the learning procedure. This can of course be compensated with scaling, but is in general not wished as it is most important to capture the most dominant modes. The same can be said about the CUR decomposition, the more basis you add to the reduction matrix, the more complex and rare behavior will be represented in the corresponding latent coordinates and the harder it gets to capture those effects. Noticeably, the latent errors of PCA and KPCA vary in a very similar range supposing that the corresponding latent dynamics behave similarly.

For the autoencoder, one can not simply transfer these properties as the individual contribution of single latent coordinates are not as distinguishable as it is the case for the linear reduction techniques. Nevertheless, the capacity that comes with a growing latent dimension leads to features with more complexity. Hence, in general one can say that a larger latent space leads on the one hand to a better reconstruction, but on the other hand to latent dynamics that are more difficult to approximate and the choice of the optimal dimension is an act of balance.

Errors in physical space In addition to the normalized error measures presented thus far, Fig. 5.5a and Fig. 5.5b report absolute errors in the physical space for surrogate models with an intrinsic latent dimension of $r = 4$. Remarkably, even at this low-dimensional setting, surrogates employing nonlinear reduction techniques achieve a mean Euclidean displacement error of less than one centimeter for the displacements—despite the highly dynamic nature and complexity of the underlying crash simulation, where nodal displacements can reach up to 150 cm. Surrogates employing PCA also demonstrate comparably low errors, remaining below two centimeters. Only the surrogates using CUR decomposition exhibit notable deviations. Considering the stress approximation, on the contrary, it becomes evident once again that the surrogates struggle to accurately replicate the reference. They exhibit mean errors ranging from $3 \cdot 10^7$ Pa to $9 \cdot 10^7$ Pa, even though the maximum reference stress does not exceed approximately $7.52 \cdot 10^7$ Pa.

Computational times Beyond pure approximation accuracy, the associated computational times for both training and inference are critical metrics when selecting a suitable surrogate modeling architecture. To this end, we report the training times as well as the evaluation (inference) times for the various surrogate models in Fig. 5.5c. Although the reported times correspond to a specific choice of latent dimensionality and are limited to the displacement data, the observed trends are representative and can be expected to transfer to other configurations and physical quantities.

From these results, it is evident that autoencoder-based surrogates require by far the most computational effort during training, resulting in significantly higher training times. This is expected due to the larger number of trainable parameters and the iterative optimization



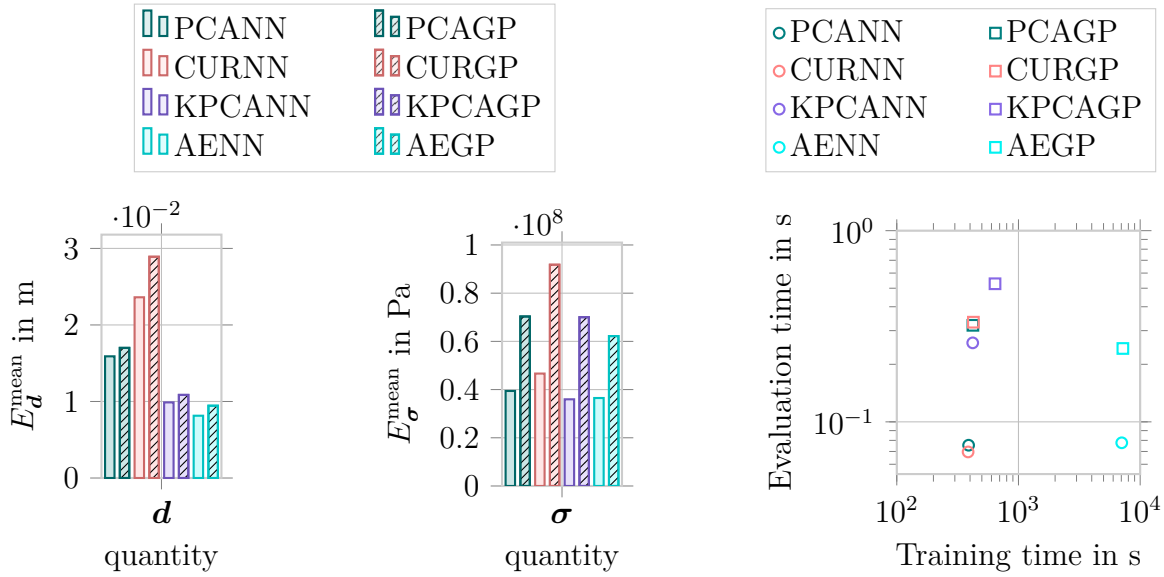
(a) displacement $r = 4$ (full data)

(b) displacement $r = 50$ (full data)

(c) stress $r = 4$ (full data)

(d) stress $r = 50$ (full data)

Figure 5.4: Boxplot of the normalized reconstruction error $E_{\breve{x}}$, latent error E_z, and state error E_x for the displacements and stress values of the kart model. The shown results show the error ranges obtained over all test simulations and are given for two different latent dimensions.



(a) displacement error

(b) stress error

(c) Computational times predicting the displacement values.

Figure 5.5: Absolute errors for the displacements (a) and stress (b) as well as the associated computational times (c) achieved by surrogates with a latent dimension of $r = 4$ for the kart model.

required for both encoder and decoder networks. It should be noted, however, that the absolute training time is highly dependent on factors such as batch size and number of training epochs, meaning it may vary considerably under different hyperparameter settings. Nonetheless, the observed trend remains consistent across comparable configurations. In contrast, the other reduction methods—such as PCA, KPCA, and CUR—exhibit training times within a similar order of magnitude. The surrogates relying on GPs for the latent approximation generally require longer optimization times, owing to the cost of kernel matrix computation and hyperparameter optimization.

When analyzing evaluation times, the picture shifts further: GPs again perform worse than their neural network counterparts, showing significantly higher inference times. Moreover, surrogate models based on KPCA for dimensionality reduction also demonstrate notably longer evaluation times, likely due to the cost of nonlinear kernel evaluations during projection and reconstruction. Interestingly, the surrogate models PCANN, CURNN, and AENN all achieve similar inference times of approximately 70 ms. This is attributed to the efficient implementation of the autoencoder, which compensates for the more complex operations through optimized computation. In a unified and performance-optimized deployment environment, linear reduction techniques—such as PCA—result in even faster evaluation times, further improving the real-time applicability of these surrogate models.

5.1.3 Discussion

The presented results show that the performance of different surrogate models varies significantly across scenarios, with each combination exhibiting distinct strengths and computational demands. In general, nonlinear reduction schemes (e.g., KPCA, AE) are preferable for low latent dimensions, as they better capture complex system behavior in compact representations. In contrast, linear methods (such as PCA) stand out due to their low computational cost, simple hyperparameter tuning, and competitive performance at higher latent dimensions, making them attractive for efficient surrogate modeling in less complex settings.

Both NNs and GPs demonstrate strong capabilities in capturing latent dynamics for the displacements, with a slight performance advantage observed for neural networks. In addition, GPs consistently failed to capture the stress dynamics. Nevertheless, they offer the additional benefit of predictive uncertainty estimates (in the form of standard deviations), which can be valuable in applications requiring confidence quantification. That said, GPs also come with higher memory requirements and may become computationally limiting for large-scale datasets.

Regarding the reduction techniques, KPCA and PCA tend to produce similar latent representations, resulting in comparable latent errors $E_{\boldsymbol{z}}$. The autoencoder-based surrogates generally deliver satisfactory results, despite their computationally expensive training phase. Interestingly, even though a fully connected autoencoder was used—without architectural specialization for spatial structures—it still achieved comparable performance to other methods, even in data-scarce regimes, which is somewhat contrary to expectations. This underlines the robustness of the autoencoder architecture and its potential for surrogate modeling when properly regularized and tuned.

All surrogate models can be evaluated within milliseconds, in stark contrast to the 15–20 minutes required by the HF model per simulation. This results in a speedup of 4 to 5 orders of magnitude, demonstrating the immense computational efficiency achieved by the surrogate approaches. Although this comparison has its limitations—as the FE simulations typically involve the computation of additional physical quantities and the storage of large datasets—it nevertheless highlights the practical advantage of surrogate models. Their rapid evaluation capability makes them particularly well-suited for multi-query scenarios (e.g., optimization, uncertainty quantification) as well as for real-time applications, where fast response times are critical.

However, while the surrogate models provide satisfactory approximation quality for displacements, the stress field poses a distinct and significantly more challenging problem. In general, the approximation quality for stress does not reach a satisfying level, clearly illustrating the limitations of the surrogate modeling schemes considered. Capturing the stress field in a meaningful and compact latent representation proves particularly difficult

in the investigated scenario. Since the quality of dimensionality reduction acts as a natural upper bound for the overall surrogate accuracy, this step already introduces a bottleneck. Moreover, the dynamics of the stress-related latent variables are inherently more complex and nonlinear, making them harder to approximate accurately using standard regression algorithms.

As a result, the direct approximation of the stress field may be suitable only in scenarios with relatively low accuracy requirements. In applications demanding high-fidelity stress predictions, alternative strategies are necessary. These include the development of new, tailored surrogate modeling techniques specifically designed to handle stress fields, or the use of indirect reconstruction, where the stress field is computed from the predicted displacements using standard FE post-processing tools. This latter approach leverages the reliable displacement approximation and established physical relationships to ensure consistency and physical plausibility in the stress predictions.

Beyond approximation quality, soft factors also play an important role when choosing a suitable dimensionality reduction algorithm. These factors pertain to practical considerations during implementation and deployment—especially in the offline phase and in terms of computational resource consumption. For instance, PCA is generally straightforward to apply and often works out-of-the-box without the need for elaborate tuning. In contrast, KPCA and AEs require more extensive hyperparameter tuning.

Moreover, autoencoders typically require a computationally expensive training phase, especially when large datasets or deep architectures are used. On the other hand, KPCA—while computationally lighter to train—tends to be slower during inference, due to its reliance on kernel matrix evaluations involving the full training dataset. In summary, while nonlinear methods may offer improved approximation quality in certain scenarios, they often come at the cost of higher implementation effort, training time, and resource demands. These aspects should be weighed carefully, particularly when targeting real-time applications, resource-constrained environments, or rapid prototyping.

We summarize the main characteristics of the reduction and regression techniques used:

i) **Principal Component Analysis (PCA)**: A simple, efficient method with cheap training or tuning required. Its linearity, however, limits performance on systems with nonlinear dynamics or at low latent dimensions.

ii) **CUR Decomposition (CUR)**: Provides interpretable modes and low evaluation cost, but suffers from randomness and consistently lower accuracy.

iii) **Kernel Principal Component Analysis (KPCA)**: Captures nonlinear structure effectively but is computationally and memory intensive, with more costly online evaluation.

iv) **Autoencoder (AE)**: Delivers high-quality approximation quality with fast inference,

though at the cost of expensive training and more complex model design, benefits least from increased latent dimensionality.

v) **Neural Networks (NNs)**: Flexible and efficient at runtime, NNs showed robustness across the used combinations.

vi) **Gaussian Processes (GPs)**: Offer strong predictions and built-in uncertainty quantification, especially for small data, but are memory-heavy and failed to capture stress latent dynamics.

5.2 Surrogate Modeling Approach for Sequential Input Data

The previous section focused on predicting system behavior for fixed parameters, such as physical or geometrical properties—e.g., material constants, initial conditions, or boundary values—a strategy also commonly adopted by many existing data-driven surrogate modeling approaches for structural dynamical systems. However, in many practical applications, sequential parameter input series must be considered, and in such cases, those methods often reach their limits in approximation quality as exemplified in Section 4.3.

One representative example for such a scenario is the occupant model introduced in Section 2.2.2. It is influenced by time-varying accelerations arising from pre-crash driving scenarios. Such applications require methods capable of accurately modeling the system's dynamic response under continuously evolving input conditions. In these cases, the current system state is not solely determined by the present input, but also by its temporal context, i.e. past input values. Consequently, surrogate models must be equipped to capture these temporal dependencies to achieve meaningful and accurate predictions.

To address this, a methodology is presented in the following that is tailored to handle time-dependent parameters. We proposed this approach first in [KneiflHayFehr22b]. In particular, an autoregressive surrogate model is presented, which employs a Long Short-Term Memory (LSTM) network to sequentially predict the evolution of latent states. These latent states are obtained by applying PCA to the high-dimensional system trajectories. The LSTM is conditioned on the sequence of parameter inputs that influence the system dynamics over time as well as the history of latent states. In contrast to conventional methods, we implement a skip-connection to the last layer of the LSTM so that it only needs to model the difference from one latent state to another. The objective of this study is to develop a surrogate that can accurately and efficiently model the occupants' response under dynamic inputs, with the overarching goal of achieving real-time capability.

An alternative strategy for addressing time-dependent parameter dependencies is proposed by [ZhuangEtAl21], who combine Model Order Reduction (MOR) with a Runge-Kutta

neural network. In this approach, the neural network is trained to approximate the derivative of the system state, while the prediction of future states is carried out mimicking the structure of classical Runge-Kutta methods. Additionally, [WilliamsZahnKutz24] present a method that integrates RNNs with a decoder to reconstruct complex spatio-temporal dynamics from sparse sensor measurements. Although their approach is not explicitly situated within the field of MOR, the conceptual parallels—particularly the use of sequence modeling and full-field reconstruction—underscore its relevance. These similarities highlight a broader trend in the literature toward leveraging temporal learning architectures for efficient and expressive surrogate modeling.

5.2.1 Methodology

The first step of the methodology follows procedures similar to those discussed previously, namely the reduction of the high-dimensional system state into a low-dimensional latent representation using PCA. The use of this linear reduction technique is deemed sufficient for the occupant model, as it provides satisfactory reconstruction quality with a relatively small number of modes, while maintaining low computational cost.

Once a reduced representation of the system states has been obtained, the latent dynamics can be formulated as a discrete state transition of the form

$$\boldsymbol{z}_{i+1}(\boldsymbol{\mu}_i) \coloneqq \boldsymbol{z}_i(\boldsymbol{\mu}_i) + \Delta \boldsymbol{z}_i(\boldsymbol{\mu}_i), \tag{5.10}$$

where the state at the next time step is computed as the current state plus a state increment. We use the subscript i as shorthand notation to indicate that the corresponding quantity is associated with the time step t_i, e.g., $\boldsymbol{z}_i(\boldsymbol{\mu}_i) = \boldsymbol{z}(t_i, \boldsymbol{\mu}(t_i))$. We leverage this formulation by designing the surrogate model to explicitly mimic the structure of (5.10). In particular, the latent dynamics are modeled as

$$\tilde{\boldsymbol{z}}_i(\boldsymbol{\mu}_{i+1}) \coloneqq \tilde{\boldsymbol{z}}_i(\boldsymbol{\mu}_i) + \Delta \tilde{\boldsymbol{z}}_i(\boldsymbol{\mu}_i) = \tilde{\boldsymbol{z}}_i(\boldsymbol{\mu}_i) + \boldsymbol{\Psi}(\boldsymbol{\alpha}_i; \boldsymbol{W}), \tag{5.11}$$

where $\boldsymbol{\Psi}$ denotes an LSTM network with learnable weights $\boldsymbol{W}$, and $\boldsymbol{\alpha}_i$ is the input sequence at time step i. The LSTM predicts the state increment $\Delta\tilde{\boldsymbol{z}}$, and a skip connection is used to propagate the latent state forward in time, starting from an initial condition $\tilde{\boldsymbol{z}}_0(\boldsymbol{\mu}_0) \coloneqq \boldsymbol{z}_0(\boldsymbol{\mu}_0)$. Accordingly, the full surrogate model—including the reconstruction $\boldsymbol{\psi}$ into physical space—takes the form

$$\tilde{\boldsymbol{x}}_{i+1} = \tilde{\boldsymbol{F}}(\boldsymbol{\mu}_i) = \boldsymbol{\psi}(\tilde{\boldsymbol{z}}_i(\boldsymbol{\mu}_i) + \boldsymbol{\Psi}(\boldsymbol{\alpha}_i; \boldsymbol{W})). \tag{5.12}$$

The LSTM is trained and optimized as described in Section 4.1.2, such that it best approximates the true latent state difference $\Delta\boldsymbol{z}$ with respect to a chosen loss function, ensuring accurate rollout of the dynamic trajectory.

For training, we construct a dataset comprising latent state increments $\Delta \boldsymbol{z}(t_i, \boldsymbol{\mu}_i)$ and the corresponding input sequences $\boldsymbol{\alpha}_i$. Each input sequence is composed of the preceding n_w latent states $\boldsymbol{z}$ along with their associated parameters $\boldsymbol{\mu}$. This results in the following dataset structure:

$$\mathcal{D} = \left(\underbrace{\begin{bmatrix} \boldsymbol{z}_{i-n_w}(\mu_{i-n_w,j}) \dots \boldsymbol{z}_{i-1}(\boldsymbol{\mu}_{i-1,j}) \\ \boldsymbol{\mu}_{i-n_w,j} \cdots \boldsymbol{\mu}_{i-1,j} \end{bmatrix}}_{\boldsymbol{\alpha}_{i,j} \in \mathbb{R}^{(r+n_p)\times n_w}}, \quad \underbrace{\left[\Delta \boldsymbol{z}_i(\boldsymbol{\mu}_{i,j})\right]}_{\boldsymbol{y}_{i,j} \in \mathbb{R}^r} \right)_{i \in \mathcal{T}}^{j \in \{1,\dots,n_{\text{sim}}\}}, \tag{5.13}$$

where j distinguishes the different simulations. We note that the first n_w samples of each simulation lack sufficient predecessors and are therefore excluded from training.

All these components are integrated into a unified surrogate modeling approach that proceeds in two phases:

1. The offline phase (training): The high-fidelity model is evaluated to generate data, and the reduction matrix $\boldsymbol{V}$ is computed via PCA. Based on the reduced trajectories, the dataset (5.13) is assembled. Finally, the LSTM network is trained on this dataset utilizing formulation (5.12) to learn the temporal dynamics in latent space.

2. The Online phase (inference): For a new sequence of time-dependent parameters, the model starts from the reduced initial state $\boldsymbol{z}_0$ and its associated parameter $\boldsymbol{\mu}_0$. At each time step, the LSTM predicts the state increment $\Delta\tilde{\boldsymbol{z}}$, from which the updated latent state is computed. This newly obtained state is then used as part of the input for the next prediction and is projected back into the physical space.

Importantly, since the first few predictions lack full-length histories, the ability of LSTM networks to handle input sequences of variable length becomes crucial for ensuring accurate early-time predictions and a smooth rollout of the full trajectory.

Implementation The HF data is reduced to a latent space of dimension $r = 30$. The LSTM network architecture begins with a masking layer followed by stacked LSTM layers, as illustrated in Fig. 5.6. The masking layer ensures compatibility with sequences of variable length, which is particularly important in the early time steps when full input histories are not yet available. The final LSTM layer consists of $r = 30$ units, ensuring consistency with the dimensionality of the latent space.

All LSTM layers operate on a temporal input horizon of $n_w = 8$ time steps, meaning each prediction is conditioned on the eight previous states and parameters. The key hyperparameter settings are summarized in Table 5.2. These include architectural and training-specific choices, such as the number of hidden units, optimizer settings, and batch size. All hyperparameters—including r, the number of neurons n_{n}, the time horizon n_w, and the learning rate γ_{lr}—are selected using a grid search based on performance on the validation set.

Table 5.2: Hyperparameters for the surrogate modeling of the occupant example.

Hyperparameter Category	**Details**
Dimensions	Latent State: $r = 30$
Time horizon	$n_w = 8$
Optimizer	RMSprop, Learning Rate: $1e-3$
Training	Epochs: 150, Batchsize: 5
Hidden Layer sizes	$256 \times 256 \times 256 \times 30$
Regularization	Select best weights w.r.t. validation data
Data	$n_{\text{sim}}^{\text{train}} = 90$, $n_{\text{sim}}^{\text{val}} = 11$, $n_{\text{sim}}^{\text{test}} = 6$, $n_t = 50$–221, $t_{\text{end}} = 1.225$–$5.5\,\text{s}$

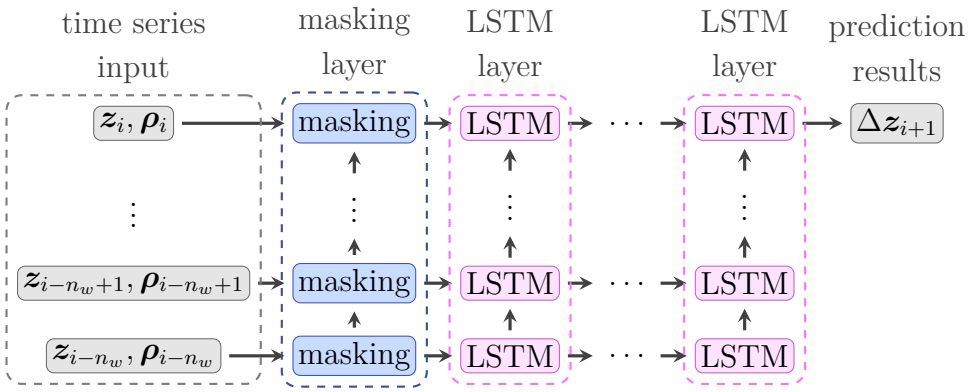


Figure 5.6: Network architecture for sequential surrogate modeling in the latent space.

5.2.2 Results

In this section, the performance of the surrogate model is validated on the test simulations. All high-fidelity and surrogate evaluations are carried out on a machine equipped with an Intel Core i7-6700 CPU and 32 GB of RAM. The considered test scenarios vary in their temporal lengths. As a result, the test dataset comprises simulation trajectories of varying lengths, ranging from 74 to 164 time steps, leading to a total of 600 individual test samples. A comprehensive summary of the performance across these test scenarios is presented in Table 5.3.

It is noteworthy that the reconstruction error remains consistently below 1% across all test simulations, which confirms the suitability of a linear dimensionality reduction for the occupant model. This observation justifies the use of PCA as an efficient and sufficiently

Table 5.3: Error quantities of six test simulations.

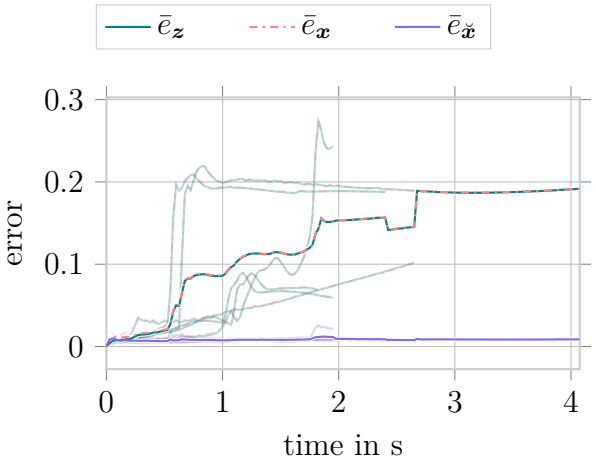
	sim 1	sim 2	sim 3	sim 4	sim 5	sim 6	mean
$E_{\boldsymbol{z}}$	0.052	0.037	0.144	0.161	0.039	0.045	0.080
$E_{\boldsymbol{x}}$	0.053	0.039	0.145	0.162	0.040	0.047	0.081
$E_{\breve{\boldsymbol{x}}}$	0.009	0.007	0.010	0.009	0.008	0.008	0.008
$E_{\boldsymbol{d}}^{\max}$in cm	0.285	0.407	0.642	0.659	0.382	0.989	0.561
${}^{\Delta t^{\tilde{\boldsymbol{F}}}}/_{t_{\text{end}}}$ (-)	0.982	0.989	0.984	0.975	0.990	0.990	0.985

expressive reduction method for this application. Furthermore, the latent regression error $\bar{e}_{\boldsymbol{z}}$ and the overall approximation error in the physical state space $\bar{e}_{\boldsymbol{x}}$ are nearly identical. This indicates that the quality of the latent dynamics approximation—as learned by the autoregressive surrogate—is the dominant factor determining the overall prediction accuracy.

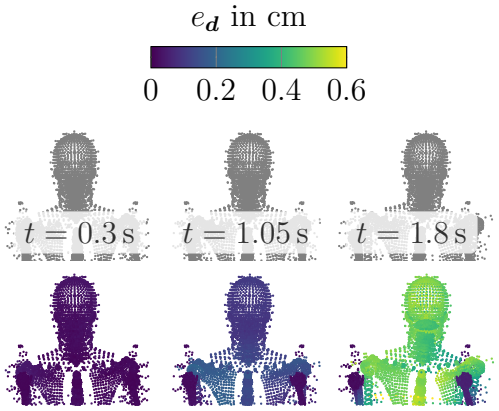
To verify this observation, we closely examine the results shown in Fig. 5.7a, where the latent error, reconstruction error, and state-space approximation error are plotted over time for all test simulations. The analysis clearly indicates that the state-space approximation error $\bar{e}_{\boldsymbol{x}}$ is predominantly governed by the latent approximation error $\bar{e}_{\boldsymbol{z}}$. For the majority of the simulation duration, the two curves closely coincide, demonstrating that the quality of the latent dynamics prediction is the key factor determining the surrogate's overall performance. Meanwhile, the reconstruction error remains consistently low and near optimal throughout the simulations, contributing only marginally to the total error.

In general, the model maintains a high level of accuracy. The maximum node-wise displacement error $E_{\boldsymbol{d}}^{\max}$ observed across all test simulations remains below 1 cm, which demonstrates the robustness of the surrogate even for longer rollouts. Moreover, the surrogate exhibits a significant advantage in terms of computational efficiency. Comparing the ratio between the computation time Δt and the simulated time t_{end}—where a value below one indicates real-time capability—highlights the efficiency gap between the surrogate's results ${}^{\Delta t^{\tilde{\boldsymbol{F}}}}/_{t_{\text{end}}} = 0.985$ and the HF results ${}^{\Delta t^{\text{HF}}}/_{t_{\text{end}}} = 2271.70$.

As a final comparison, a visualization of the occupant's motion at selected time points is provided in Fig. 5.7b, using the test simulation with the highest dynamics. During the initial phase of the motion, almost no noticeable differences can be observed between the high-fidelity and surrogate predictions. Only towards the end of the simulation does a quantitative deviation emerge, particularly in the upper body region. Nevertheless, the overall motion sequence is captured accurately throughout the entire trajectory. This visual assessment confirms that the surrogate model closely approximates the high-fidelity occupant simulation, even in dynamic and nonlinear regimes—while requiring only a fraction of the computational effort.



(a) State, latent, and reconstruction errors over time across all test simulations. Transparent lines show individual trajectories; the opaque line shows the mean. Only available trajectories are used to compute the mean at each time step.



(b) Motion of the occupant's upper body at certain time points for an example test simulation. The reference solution is shown in gray in the upper half, whereas the nodes of the approximation in the bottom half are colored with respect to their individual node distance error.

Figure 5.7: Relative error for the displacements of one node over time (a) and overall error (b) for the occupant model (b).

5.2.3 Discussion

The presented combination of PCA and LSTMs demonstrates the capability to construct an efficient and accurate surrogate model for an accelerated human occupant simulation. For simulations of limited duration, the method effectively captures even complex time-dependent behaviors, while achieving a remarkable reduction in computation time. However, it is also observed that the approximation error tends to increase over time and mainly stems from the approximation of the dynamic behavior in the latent space.

This error source, however, cannot be solely attributed to the learning capacity of the LSTM. It is also inherently linked to the autoregressive structure of the approach, which introduces a characteristic trade-off: while enabling sequential modeling, it also causes error accumulation over time. That is, once a deviation occurs at any time step, it is propagated throughout the remainder of the simulation, continuously contributing to further inaccuracies. Hence, non-negligible deviations from the high-fidelity reference model, especially in long-term predictions can occur. Addressing this limitation is a key direction for future work.

Possible strategies include periodic updates of the surrogate model using real-time sensor data to correct deviations or enhancing the robustness of the regression algorithm by introducing artificial noise in the training process to better withstand accumulated prediction errors. Improving these aspects will be essential for ensuring the reliability and applicability of such surrogate models in long-term and safety-critical simulations.

Besides that the surrogate is not only real-time capable but also more than 2300 times faster than the high-fidelity counterpart. This performance gain underlines the potential of the proposed approach for deployment in time-sensitive applications. Furthermore, due to its lightweight nature, the surrogate model can be easily integrated into other simulation environments without incurring significant computational overhead.

All methods introduced and compared throughout this chapter have demonstrated both the potential and limitations of surrogate modeling for structural dynamical systems. Highly efficient yet accurate surrogate models have proven capable of capturing displacements in crash scenarios and human occupant behavior under highly dynamic and nonlinear conditions. Despite these achievements, the approaches presented thus far do not yet address several important challenges, including varying hardware constraints, multi-scale phenomena, uncertainty quantification, and the enforcement of physical consistency. These topics are systematically explored in the subsequent chapters.

The surrogate model used to generate the presented results along with the corresponding data can be accessed via [KneiflHayFehr22a] under BSD-3-Clause License.

Chapter 6

Multi-hierarchic Surrogate Modeling

I think it's time to leave and find another road
'Cause I'm sure there's a story that is yet to be told.

Bukahara, Border

In the analysis of structural dynamical systems, effects frequently occur at smaller and larger scales, necessitating models capable of capturing diverse spatial resolutions (cf. Chall. 5). In conventional surrogate modeling approaches, micro- and macroscale characteristics are often collectively processed, potentially overshadowing microscale features by their larger counterparts. The modeling of multi-scale systems poses a particular challenge, and as such, it is often approached in a special way. For example, multigrid methods [McCormick87, TrottenbergOosterleeSchuller00] employ a coarse-grained model that is gradually refined in areas of high inaccuracy to achieve the required global accuracy. Such methods have also been unified with convolutional neural networks [HeXu19].

In addition, the majority of data-driven surrogate modeling approaches operate directly on a given high-dimensional discretization. Hence, they are tied to this fixed high resolution. However, the large number of variables used to describe a system often stems from the numerical discretization scheme rather than the underlying physical phenomena, cf. Section 3.1, although often only a small subset of these variables is of interest. Thus, unnecessary expenditure of computational resources are the consequence and information flow over large spatial distances is complicated.

In contrast, the employment of a surrogate capable of providing predictions across various resolutions has the potential to reduce associated computational costs, during evaluation but also in the expensive visualization of complex three-dimensional systems. Furthermore, static resolutions are incapable of reacting to dynamical changes in computational

environments. Hence, changes in available memory, electric energy, or even the desired approximation quality can not be considered. Fortunately, data-driven surrogates offer the flexibility to operate on arbitrarily selected quantities of a model as they are not tied to the spatial convergence of classic numerical models. The accuracy of such data-driven surrogates is constrained only by their expressiveness and the quality of the high-fidelity data. This raises important questions: Why should data-driven surrogate models be confined to the fixed original resolutions and how can multi-scale features be effectively addressed within a framework?

To address both questions, we developed an approach in [KneiflEtAl24] that is designed for the precise capture of micro- and macroscale features (Chall. 5) and the elegant handling of dimensionality (Chall. 1) by automatically selecting an optimal subset of spatial quantities to evaluate the system. Specifically, a Multi-Hierarchical (MH) framework is introduced to systematically develop a series of surrogate models at different resolutions. The macroscale features are resolved on coarse surrogates, while microscale effects are captured on finer surrogates, with transfer learning facilitating the exchange of information across scales. To implement this, the high-resolution mesh of the original numerical model is simplified to generate visually interpretable multi-resolution representations.

Moreover, the surrogates on each level utilize the recent advancements in Graph Convolutional Neural Networks (GCNNs) [ZhouEtAl20]. In detail, the individual surrogate models are composed of graph convolutional autoencoders for the task of dimensionality reduction in combination with Multi Layer Perceptrons (MLPs) to capture the low-dimensional latent dynamics on temporal and parametric inputs. Subsequent surrogates are trained on residuals using increasingly finer resolutions. By doing so, we can speed up the learning process, create multiple surrogates that balance hardware requirements with accuracy, and resolve multi-scale issues that often arise in spatio-temporal dynamics of structural systems.

The following steps comprise the core idea of the proposed methodology: (i) the High-Fidelity (HF) model is reformulated as a graph, (ii) coarse visually interpretable representations are generated via mesh simplification, (iii) a surrogate is created on the coarsest representation, (iv) the model is refined to the next finer level, and (v) another surrogate is trained on the refined level using transfer learning. Steps (iv) and (v) are iterated until either a desired performance threshold is achieved or no further coarse representations are available. This approach adapts and extends concepts from multiresolution autoencoders [LiuEtAl23], tailoring them specifically to handle irregular data effectively. A visual impression of this workflow is depicted in Fig. 6.1. There, a structural similarity of the approach to U-NETs [RonnebergerFischerBrox15] can be observed.

By following this approach, we leverage the fact that the surrogates are not constrained to a fine spatial resolution enabling to exclude substantial parts of the model during the surrogate's creation. Consequently, the approach is inherently mesh-free, meaning it is

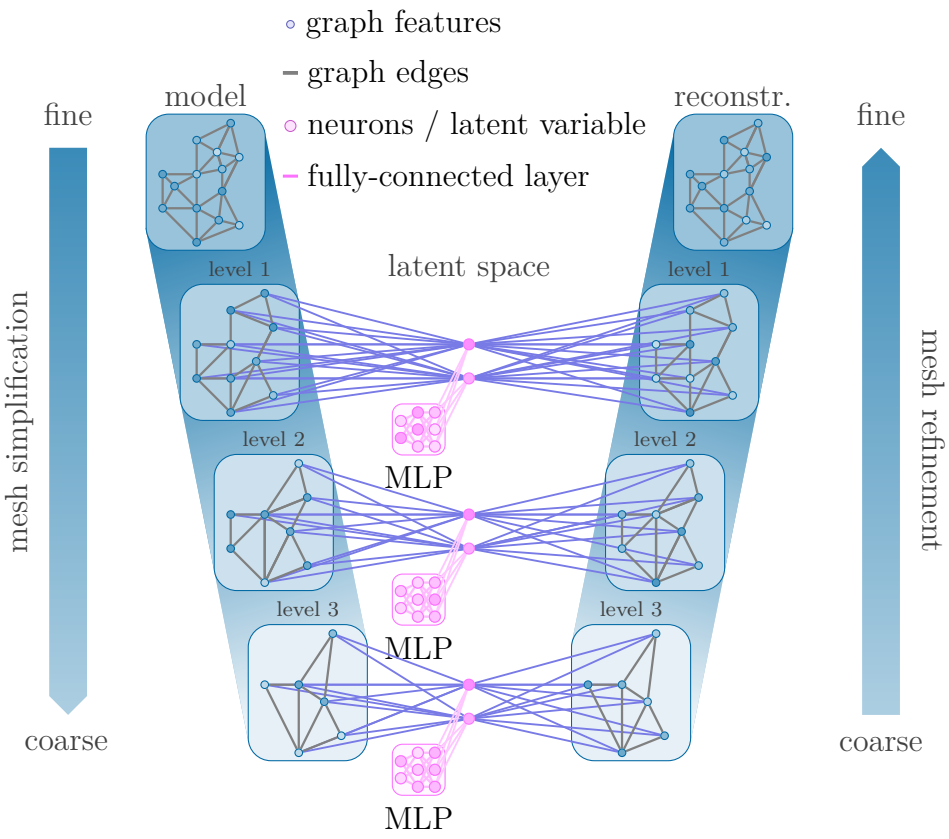


Figure 6.1: The MH surrogate modeling approach executes the learning process on coarse representations of the original system rather than on the full discretization itself. Initially, different levels of discretization are generated for the original model (Section 6.2.2). On the coarsest level, a surrogate model is trained (Section 6.2.3). The coarse representation is then progressively refined and subsequent surrogate models are trained on the residual error Section 6.2.4. This refinement process is repeated.

not limited to the underlying high-resolution discretization. Further discretization-free approximation schemes for parametrized Partial Differential Equations (PDEs) can be found in the literature, including the works of [ChenEtAl22b], [RodriguezEtAl22] and [LiEtAl20a, LiEtAl20b]. Due to the hierarchical structure of our proposed approach, the framework is naturally suited for multi-scale problems, where macro- and microscale dynamics occur at the same time. Furthermore, the sequential construction of surrogates enables the termination of the learning process at any desired point, thereby leveraging the knowledge acquired from previously learned behaviors through the application of transfer learning from coarser to refined surrogates.

The key highlights of the presented framework can be summarized as follows:

1. A multi-hierarchical nonlinear model reduction framework is introduced that constructs adaptive surrogate models tailored to different computational needs. This

approach operates within visually interpretable domains and leverages transfer learning to iteratively refine surrogate models.

2. The proposed surrogate modeling architecture is particularly well-suited for multi-scale problems and highly complex, discretized three-dimensional structural dynamical systems.

3. Efficient yet highly accurate data-driven surrogates are derived for the deformation behavior of the nonlinear Finite Element (FE) kart frame crash simulation.

In the ensuing sections, the fundamentals of GCNNs are elucidated, prior to an explanation of the multi-hierarchical surrogate modelling process itself. This is followed by numerical results on the kart example and a breakdown of the results.

6.1 Graph Convolutional Neural Networks

GCNNs [ZhouEtAl20] represent an evolution of Convolutional Neural Networks (CNNs), which are widely used in deep learning for tasks like image classification and object detection. CNNs rely on convolutional filters (small matrices such as 3×3 or 5×5) that slide across structured input data, extracting features like edges or patterns, while pooling layers reduce the dimension of the data [LiEtAl22]. By applying learnable filters across multiple data points, CNNs leverage weight sharing, significantly reducing the number of trainable parameters compared to fully connected networks and enhancing generalization by detecting local patterns. However, CNNs are inherently designed for regular grid-like data as it occurs in images, making them less effective for irregular data structures.

Irregular data, such as those arising from FE simulations, require alternative approaches, driving advancements in geometric deep learning [BronsteinEtAl21]. Some techniques address this by mapping irregular data onto a structured grid [GaoSunWang21], enabling the use of standard convolution operations. Other strategies directly apply convolutional-like operations on graphs dynamically constructed from point clouds [WangEtAl19], or adapt CNN architectures to handle non-Euclidean spaces [MontiEtAl17].

GCNNs are designed to handle irregular data by adapting the concept of convolutions to geometric structures. These networks extract information about node features and their relationships through spatial connections. Early work on GCNNs is presented in [DefferrardBressonVandergheynst16], with an influential adaptation described in [KipfWelling16]. In the context of Model Order Reduction (MOR), GCNNs have been utilized in several studies. For example, [GruberEtAl22] compares a graph convolutional autoencoder with a traditional fully connected Autoencoder (AE) for generating reduced-order models. Similarly, [PichiMoyaHesthaven24] apply a spatial graph convolutional autoencoder to derive reduced-order models, while [FrancoEtAl23] explores the use of

GCNNs for approximating time-dependent PDEs under geometric variability. Unlike conventional convolutions, graph convolutions do not inherently incorporate dimensionality reduction. To address this, specialized pooling operations tailored for irregular data have been developed [GrattarolaEtAl21]. Comprehensive insights into GCNNs are provided in [WuEtAl21], and a review focused specifically on their application in MOR is available in [PichiMoyaHesthaven24].

In addition to GCNNs, various innovative approaches in geometric deep learning have gained attention, such as graph networks [BattagliaEtAl18]. Graph network-based simulation models have been applied to create physically informed simulations, where the graph network predicts state-time derivatives that are subsequently integrated using Ordinary Differential Equation (ODE) solvers to evolve the system in time [PfaffEtAl20, SanchezGonzalezEtAl20]. Furthermore, symbolic representations of learned models can be uncovered by applying symbolic regression to elements of its message-passing function [CranmerEtAl19, CranmerEtAl20]. Graph networks also play a role in generative tasks, where graphs are constructed sequentially based on learned probability distributions [LiEtAl18]. A noteworthy recent advancement is the emergence of attention-based graph transformer models [MinEtAl22]. These models have been used as neural operators to approximate solutions of PDEs [BryutkinEtAl24], showcasing their potential for capturing complex patterns in geometric data.

Other works also incorporate hierarchical structures into GCNNs as this can offer a powerful way to enhance their performance. For example, Graph U-Nets [GaoJi21] introduce pooling layers to create smaller, simplified graphs by leveraging a trainable projection vector. Similarly, the multi-scale MeshGraphNet proposed in [FortunatoEtAl22] operates across multiple resolutions. This approach uses coarse-grained representations to facilitate efficient information propagation and address the challenge of slow information propagation across finer resolutions. In [JainHaghighatNelaturi24], a dual MeshGraphNet framework is employed to build surrogate models for finite element simulations of latticed structures. The first network models the dynamics on a reduced graph representation, while the second maps these results onto the full-scale displacements, ensuring accuracy and efficiency.

Another example of mesh reduction is found in [HanEtAl22], where information from a fine-grained graph is encoded into a coarser subset of nodes. This compressed representation enables efficient temporal evolution of latent dynamics using an attention-based model. Hierarchical techniques are not limited to graph applications but, for instance are also integrated into variational autoencoders [LeeEtAl23]. Moreover, hierarchical structures are not only utilized in spatial domains but also for temporal dynamics as well time-evolving systems [LiuKutzBrunton22]. These versatile approaches demonstrate the potential of hierarchical designs in diverse domains of geometric deep learning. However, all these approaches utilize hierarchical structures solely to facilitate the learning process for an

approximation of the fine discretized high-fidelity solution. In contrast, our approach employs coarse representations that are physically and visually interpretable, thereby ensuring direct utility. Moreover, we refrain from subjecting costly training of a surrogate on high-resolution data and opt for a reduction of the latent variable to a vector of the intrinsic dimension instead of representing the system on a coarse graph.

6.1.1 Fundamentals of Graph Convolutional Neural Networks

A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is composed out of nodes $\boldsymbol{\nu}$ (also referred to as vertices) stored in a set of nodes $\mathcal{V}$, $|\mathcal{V}| = n_{\text{node}}$ and a set of edges $\mathcal{E} \subseteq \{(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j) \mid \boldsymbol{\nu}_i, \boldsymbol{\nu}_j \in \mathcal{V} \text{ and } \boldsymbol{\nu}_i \neq \boldsymbol{\nu}_j\}$ defining the connections between node pairs. The adjacency matrix $\mathcal{A} = \mathcal{A}(\mathcal{G}) \in \{0,1\}^{n_{\text{node}} \times n_{\text{node}}}$ provides an alternative representation of the graph's connectivity. It is a square matrix, where the entries indicate whether a pair of nodes $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j)$ is connected. Specifically, $[\mathcal{A}]_{ij} = 1$ if the nodes are adjacent, and $[\mathcal{A}]_{ij} = 0$ otherwise.

The state of all nodes in a graph is stored in a graph signal $\boldsymbol{x} \in \mathbb{R}^{n_{\text{node}}}$. For the purposes of simplicity, the ensuing explanations concentrate on scalar graph signals per node. It should be noted, however, that the values associated with each node can also be vectors, e.g. representing the nodes position. In such a case, the signal $\boldsymbol{x} \in \mathbb{R}^{n_{\text{node}} \times n_{\text{c}}}$ comprises an n_{c}-dimensional feature vector for every node. A useful approach to compute the convolution between a filter $\mathfrak{f} \in \mathbb{R}^{n_{\text{node}}}$ and the signal $\boldsymbol{x}$ is leveraging the property that a convolution corresponds to multiplication in the Fourier space. To obtain the Fourier transform $\check{\boldsymbol{x}}$ of the signal, a Fourier basis is derived from the eigenvalue decomposition of the normalized Laplacian of the graph.

The Laplacian is matrix that captures important structural information and is defined as $\mathscr{L}^* \coloneqq \boldsymbol{D} - \mathcal{A}$, where $\boldsymbol{D}$ is the diagonal matrix of node degrees, with entries $[\boldsymbol{D}]_{ii} = \sum_j [\mathcal{A}]_{ij}$. The normalized Laplacian $\mathscr{L} \coloneqq \boldsymbol{I}_{n_{\text{node}}} - \boldsymbol{D}^{-\frac{1}{2}} \mathcal{A} \boldsymbol{D}^{-\frac{1}{2}}$ is a real symmetric positive semi-definite matrix. Consequently, its eigenvalue decomposition $\mathscr{L} = \boldsymbol{\Upsilon} \boldsymbol{\Lambda} \boldsymbol{\Upsilon}^\intercal$ exists. Here, the matrix $\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{\upsilon}_1 & \boldsymbol{\upsilon}_2 & \dots & \boldsymbol{\upsilon}_{n_{\text{node}}} \end{bmatrix}$ contains the eigenvectors of the Laplacian, arranged according to their corresponding eigenvalues stored in the diagonal matrix $\boldsymbol{\Lambda}$. The eigenvectors $\boldsymbol{\upsilon}$ are referred to as the Fourier modes of the graph $\mathcal{G}$.

The Fourier transform of a signal $\boldsymbol{x}$ is then expressed as $\check{\boldsymbol{x}} = \mathscr{F}(\boldsymbol{x}) = \boldsymbol{\Upsilon}^\intercal \boldsymbol{x}$, while the inverse Fourier transform is defined as $\boldsymbol{x} = \mathscr{F}^{-1}(\check{\boldsymbol{x}}) = \boldsymbol{\Upsilon} \check{\boldsymbol{x}}$. Utilizing these transformations, the graph convolution of the signal $\boldsymbol{x}$ with a filter $\mathfrak{f}$ can be derived as

$$\boldsymbol{x} * \mathfrak{f} = \mathscr{F}^{-1}(\mathscr{F}(\boldsymbol{x}) \odot \mathscr{F}(\mathfrak{f})) = \boldsymbol{\Upsilon}(\boldsymbol{\Upsilon}^\intercal \boldsymbol{x} \odot \boldsymbol{\Upsilon}^\intercal \mathfrak{f}), \tag{6.1}$$

where $\odot$ denotes the Hadamard (element-wise) product. This equation can be further simplified to

$$\boldsymbol{x} * \mathfrak{f}_{\boldsymbol{w}} = \boldsymbol{\Upsilon} \mathfrak{f}_{\boldsymbol{w}} \boldsymbol{\Upsilon}^\intercal \boldsymbol{x}, \tag{6.2}$$

by defining the filter as $\mathfrak{f}_{\boldsymbol{w}} = \mathrm{diag}(\boldsymbol{\Upsilon}^\intercal \mathfrak{f})$ and applying $\boldsymbol{a} \odot \boldsymbol{b} = \mathrm{diag}(\boldsymbol{b})\boldsymbol{a}$. Formulation (6.2) is utilized by all spectral-based GCNNs.

This formulation is put into the context of Neural Networks (NNs) by parameterizing the filters $\mathfrak{f}_{\boldsymbol{w}} = \boldsymbol{W} = \mathrm{diag}(\boldsymbol{w})$ using trainable weights $\boldsymbol{w}$. The operation in a convolutional layer is then described by

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{a}^{(l)} \left(\boldsymbol{\Upsilon} \boldsymbol{W}^{(l)} \boldsymbol{\Upsilon}^\intercal \boldsymbol{x}^{(l)} \right), \tag{6.3}$$

where the superscripts $\square^{(l)}$ represent the layer index, $\boldsymbol{a}^{(l)}$ is the activation function applied at the l-th layer, and $\boldsymbol{W}^{(l)}$ is a diagonal matrix containing the trainable weights. The input to the layer, $\boldsymbol{x}^{(l)}$, is the graph signal at that layer, with $\boldsymbol{x}^{(0)}$ being the original input signal of the graph.

Although the formulation (6.3) enables spectral graph convolution, it has notable drawbacks. Specifically, the filters are not spatially localized, and the computation requires expensive operations due to matrix multiplications involving the dense Fourier basis $\boldsymbol{\Upsilon}$. To address these issues, [DefferrardBressonVandergheynst16] introduced ChebNet.

Chebyshev Spectral Convolutional Neural Networks In Chebyshev spectral convolutional neural networks (ChebNet), the filter $\mathfrak{f}_{\boldsymbol{w}}$ is approximated using Chebyshev polynomials T of order n_{cheb}. These polynomials are defined recursively as $T_i(\square) = 2 \, \square \, T_{i-1}(\square) - T_{i-2}(\square)$, with initial values $T_1(\square) = \square$ and $T_0(\square) = \boldsymbol{I}$, where $\square$ serves as placeholder for a variable. This recursive formulation avoids the need for computationally expensive multiplications with the dense Fourier basis by replacing them with n_{cheb} multiplications involving the sparse graph Laplacian. In particular, a filter $\mathfrak{f}_{\boldsymbol{w}}$ is computed as

$$\mathfrak{f}_{\boldsymbol{w}}(\grave{\boldsymbol{\Lambda}}) = \sum_{i=0}^{n_{\mathrm{cheb}}} \boldsymbol{w}_i T_i(\grave{\boldsymbol{\Lambda}}), \tag{6.4}$$

where the weights $\boldsymbol{w}_i$ are learnable coefficients of the polynomial filter. Moreover, $\grave{\mathscr{L}} = \frac{2\mathscr{L}}{\lambda_{\max}} - \boldsymbol{I}_{n_{\mathrm{node}}}$ is a scaled version of the Laplacian and $\grave{\boldsymbol{\Lambda}}$ its eigenvalue matrix. Due to the scaling, $\grave{\boldsymbol{\Lambda}}$ only has eigenvalues within the range $[-1, 1]$. Substituting, (6.4) in (6.2) and utilizing the transformation $\boldsymbol{\Upsilon} \mathfrak{f}_{\boldsymbol{w}}(\grave{\boldsymbol{\Lambda}}) \boldsymbol{\Upsilon}^\intercal = \mathfrak{f}_{\boldsymbol{w}}(\grave{\mathscr{L}})$, it becomes apparent that the respective graph convolution

$$\boldsymbol{x} * \mathfrak{f}_{\boldsymbol{w}} = \boldsymbol{\Upsilon} \mathfrak{f}_{\boldsymbol{w}}(\grave{\boldsymbol{\Lambda}}) \boldsymbol{\Upsilon}^\intercal \boldsymbol{x} = \mathfrak{f}_{\boldsymbol{w}}(\grave{\mathscr{L}}) \boldsymbol{x} = \sum_{i=0}^{n_{\mathrm{cheb}}} \boldsymbol{w}_i T_i(\grave{\mathscr{L}}) \boldsymbol{x} \tag{6.5}$$

gets rid of the expensive multiplication with $\boldsymbol{\Upsilon}$.

The Graph Convolutional Network (GCN), introduced in [KipfWelling16], simplifies ChebNet by employing a first-order approximation. Although effective, GCNs often

encounter issues such as overfitting and oversmoothing. To address these limitations, GCN2 [ChenEtAl20b] incorporates skip connections, enabling better information flow across multiple layers. However, for the following example, ChebNet delivers superior performance and is therefore used.

6.2 Multi-Hierarchic Surrogate Model Architecture

The fundamental premise of the multi-hierarchic approach is to operate on coarse representations of a given model rather than on its original discretization. This representation should still be visually interpretable and analyzable in the physical space. Thus, the objective is to identify a suitable subselection of the original nodes and the corresponding down- and upsampling mechanisms to find coarser equivalents of the original model.

6.2.1 Down- and Upsampling

Recall an n_{node}-dimensional graph signal $\boldsymbol{x}$ that stores information for every node of the original discretization. A downsampled signal

$$\boldsymbol{x}_{\downarrow} \coloneqq \mathscr{D}\boldsymbol{x} \subseteq \mathbb{R}^{n_{\downarrow}} \tag{6.6}$$

is an $n_{\downarrow}$-dimensional vector containing the information of a subselection of the original nodes $\mathcal{V}_{\downarrow} \subset \mathcal{V}$, $|\mathcal{V}_{\downarrow}| = n_{\downarrow}$ with $n_{\downarrow} < n_{\text{node}}$. The binary matrix $\mathscr{D} \in \{0,1\}^{n_{\downarrow} \times n_{\text{node}}} : \mathcal{V} \to \mathcal{V}_{\downarrow}$ selects which nodes are kept. Its entries are $[\mathscr{D}]_{ij} = 1$ when the i-th node in $\mathcal{V}_{\downarrow}$ corresponds to the j-th node in $\mathcal{V}$ and $[\mathscr{D}]_{ij} = 0$ else.

The method for selecting nodes follows an approach from the field of computer vision described in [GarlandHeckbert97], which utilizes iterative vertex pair contraction to choose nodes based on their spatial positions. In this process, for any given pair of nodes $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j)$, a vertex pair contraction $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j) \to \boldsymbol{\nu}_j$ involves redirecting all incident edges to $\boldsymbol{\nu}_j$, removing the second node $\boldsymbol{\nu}_i$, and eliminating degenerate edges. A quadratic error

$$\boldsymbol{\nu}_{i,\text{pos}}^{\intercal}(\mathscr{O}_i)\boldsymbol{\nu}_{i,\text{pos}} \tag{6.7}$$

is used as measure for determining whether the node $\boldsymbol{\nu}_i$ is kept. The matrix $\mathscr{O}_i \in \mathbb{R}^{4\times4}$ characterizes the distance of the node to the set of planes formed by the intersections at which it resides and $\boldsymbol{\nu}_{i,\text{pos}} = [\nu_{i,x}, \nu_{i,y}, \nu_{i,z}, 1]^{\intercal}$ is the node's position in space.

The procedure can be summarized in the following steps:

1. Identify valid vertex pairs, either connected by an edge or within a close distance.
2. For each valid pair $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j)$, determine the best node from $\{\boldsymbol{\nu}_i, \boldsymbol{\nu}_j\}$ by minimizing the cost function $\boldsymbol{\nu}_{i/j,\text{pos}}^{\intercal}(\mathscr{O}_i + \mathscr{O}_j)\boldsymbol{\nu}_{i/j,\text{pos}}$.

3. Iteratively eliminate the node in the pair $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j)$ that has the least cost, updating the costs accordingly at each step.

Besides the downsampling of the graph, we seek a method to reconstruct the original graph from its coarse representation. However, achieving a lossless reconstruction of the original one from its simplified counterpart is generally infeasible. Instead, the goal is to approximate the original graph using an upsampling matrix $\mathscr{U} \in \mathbb{R}^{n_{\text{node}} \times n_{\downarrow}}$, so that an upsampled graph signal

$$\boldsymbol{x} \approx \boldsymbol{x}_{\uparrow} \coloneqq \mathscr{U} \boldsymbol{x}_{\downarrow}, \tag{6.8}$$

approximates the original signal. In this work, we adopt the approach outlined in [RanjanEtAl18], generating the upsampling matrix concurrently with the creation of the downsampling matrix.

During the downsampling process, a retained node directly contributes to the upsampling matrix by setting $[\mathscr{U}]_{ij} = 1$ when it represents the i-th node in $\mathcal{V}$ and the j-th node in $\mathcal{V}_{\downarrow}$. For a discarded node $\boldsymbol{\nu}_k$, its position $\boldsymbol{\nu}_{k,\text{pos}}$ is projected onto the nearest triangle $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j, \boldsymbol{\nu}_l) \in \mathcal{V}_{\downarrow}$ in the down-sampled graph following

$$\tilde{\boldsymbol{\nu}}_{k,\text{pos}} = w_i \boldsymbol{\nu}_{i,\text{pos}} + w_j \boldsymbol{\nu}_{j,\text{pos}} + w_l \boldsymbol{\nu}_{l,\text{pos}}, \tag{6.9}$$

using barycentric coordinates and weights that satisfy $w_i + w_j + w_l = 1$. The upsampling matrix is then updated based on the barycentric weights as

$$[\mathscr{U}]_{ki} = w_i, \quad [\mathscr{U}]_{kj} = w_j, \quad [\mathscr{U}]_{kl} = w_l. \tag{6.10}$$

Visual examples of the resulting coarsened finite element (FE) model of the kart are provided in Fig. 6.2.

6.2.2 Multi-hierarchical Modeling

Once multiple representations of the original model at varying resolution levels are generated using the previously described downsampling approach, the surrogate modeling process can begin. To distinguish entities, formulations, and models across different levels, we use the notation $\square^{(l)}$, where the superscript l indicates the associated level. The original discretization corresponds to level 0. Moreover, for the downsampling operations, the notation $\mathscr{D}_j^i$ denotes the downsampling matrix that takes states from level i to level j with $i > j$, i.e. $\boldsymbol{x}^{(j)} = \mathscr{D}_j^i \boldsymbol{x}^{(i)}$. Conversely, for upsampling operations, the notation $\mathscr{U}_i^j$ denotes the upsampling matrix that approximates states at a finer level from those at a coarser level given by $\boldsymbol{x}^{(i)} \approx \mathscr{U}_i^j \boldsymbol{x}^{(j)}$.

The l-th surrogate $\tilde{\boldsymbol{F}}^{(l)}$ accordingly only needs to approximate the states of a subselection of nodes from the original system, i.e.

$$\tilde{\boldsymbol{x}}^{(l)}(t, \boldsymbol{\mu}) = \tilde{\boldsymbol{F}}^{(l)}(t, \boldsymbol{\mu}) \approx \mathscr{D}_l^0 \boldsymbol{x}^{(0)}(t, \boldsymbol{\mu}) = \boldsymbol{x}^{(l)}(t, \boldsymbol{\mu}). \tag{6.11}$$

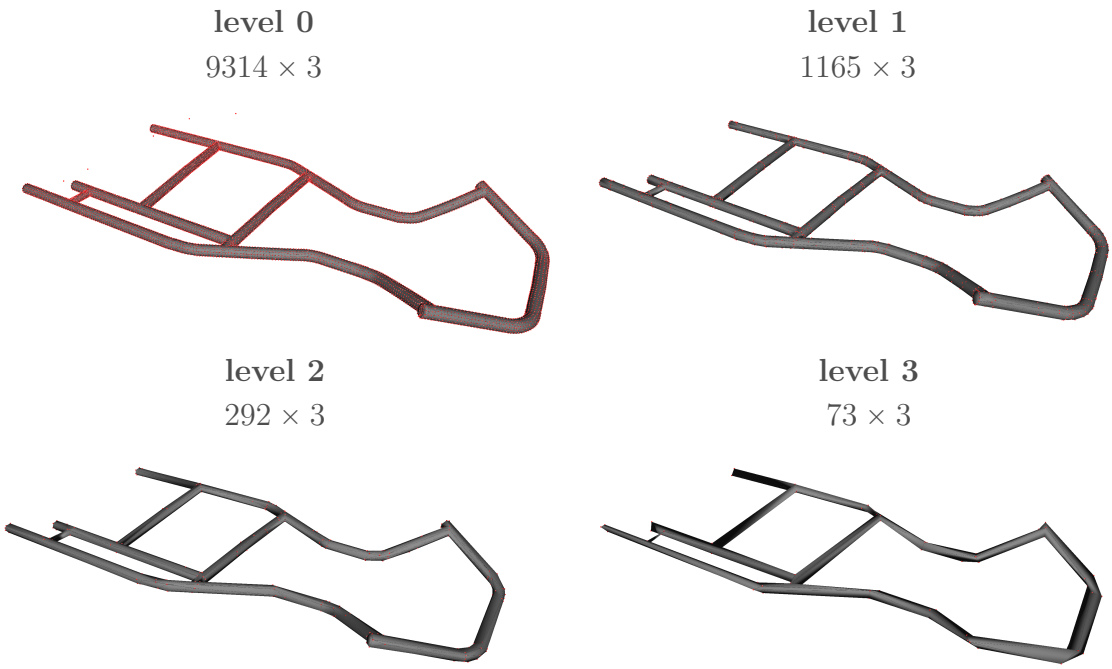


Figure 6.2: Differently resolved representations of the kart frame from the original discretization (level 0) to the coarsest one (level 3). The red dots represent the nodes and the number indicated above each kart corresponds to the number of nodes.

The MH modeling approach starts with creating a surrogate $\tilde{\boldsymbol{F}}^{(n_\ell)}$ on the deepest, i.e. coarsest level n_ℓ. Once the coarse surrogate is created, the surrogate modeling process continues to the next finer level.

6.2.3 Surrogate Model Architecture

The fundamental surrogate modeling architecture employed in this work for the MH approach, as well as for the comparative methods, is built upon the in the previous chapter introduced two key components: (i) an autoencoder—or more generally, a dimensionality reduction technique—used to learn a low-dimensional embedding $\mathcal{Z} \subseteq \mathbb{R}^r$ of the high-dimensional state space $\mathcal{X}$, and (ii) a multilayer perceptron (MLP) tasked with capturing the parameter- and time-dependencies within this low-dimensional latent manifold.

Recall that, the autoencoder comprises an encoder $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{Z}$ with trainable weights $\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}$, which maps the high-dimensional state to a compact latent representation $\boldsymbol{z} = \boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}})$, and a decoder $\boldsymbol{\psi} : \mathcal{Z} \to \mathcal{X}$ with weights $\boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}$ that reconstructs the full state from its latent representation $\breve{\boldsymbol{x}} = \boldsymbol{\psi}(\boldsymbol{z}; \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}})$. The MLP $\boldsymbol{\theta} : \mathcal{P} \times \mathcal{T} \to \mathcal{Z}$ maps the parameters and time to the corresponding latent state $\tilde{\boldsymbol{z}} = \boldsymbol{\theta}(t, \boldsymbol{\mu}; \boldsymbol{W}_{\boldsymbol{\theta}})$, with trainable weights $\boldsymbol{W}_{\boldsymbol{\theta}}$.

The complete autoencoder defines a reconstruction mapping as

$$\breve{\boldsymbol{x}} = \boldsymbol{\psi} \circ \boldsymbol{\phi}(\boldsymbol{x}), \tag{6.12}$$

while the full surrogate model capturing the parametric system dynamics is given by the composition of the MLP and the decoder:

$$\tilde{\boldsymbol{x}} = \boldsymbol{\psi} \circ \boldsymbol{\theta}(t, \boldsymbol{\mu}). \tag{6.13}$$

In contrast, to the separate optimization of reduction and latent approximation in the previous chapter, a joint optimization is used for the MH models. Accordingly, the weights $\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}$, and $\boldsymbol{W}_{\boldsymbol{\theta}}$ are optimized by minimizing the loss function

$$L(\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}, \boldsymbol{W}_{\boldsymbol{\theta}}) \coloneqq \frac{1}{n_s} \sum_{i=1}^{n_s} \lambda_{\text{approx}} \left\| \boldsymbol{x} - \tilde{\boldsymbol{x}} \right\|^2 + \lambda_{\text{rec}} \left\| \boldsymbol{x} - \breve{\boldsymbol{x}} \right\|^2 \tag{6.14a}$$

$$= \underbrace{\lambda_{\text{approx}} \left\| \boldsymbol{x} - \boldsymbol{\psi}(\boldsymbol{\theta}(t, \boldsymbol{\mu}; \boldsymbol{W}_{\boldsymbol{\theta}}), \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}) \right\|^2}_{\text{Decoder and MLP}} \tag{6.14b}$$

$$+ \underbrace{\lambda_{\text{rec}} \left\| \boldsymbol{x} - \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}})) \right\|^2}_{\text{Encoder and Decoder}}. \tag{6.14c}$$

The first term (6.14b) ensures that the surrogate accurately captures the system dynamics for given parameters and time, while the second term (6.14c) guarantees that the autoencoder can reliably reconstruct the full state from the latent representation. These loss components guide the training of the full surrogate model and are balanced by the loss factors $\lambda_{\text{approx}} \geqslant 0$ and $\lambda_{\text{rec}} \geqslant 0$.

Surrogate on the Coarsest Representation To construct the MH surrogate model at the deepest level, i.e. coarsest representation, we follow the methodology previously outlined employing a graph convolutional autoencoder. Specifically, the graph convolutional autoencoder $\boldsymbol{\Psi}_{\text{ae}}^{(n_\ell)}$ is used to learn a low-dimensional embedding $\mathcal{Z} \subseteq \mathbb{R}^r$ for the state of the coarsened graph, given by $\boldsymbol{x}^{(n_\ell)}(t, \boldsymbol{\mu}) = \mathscr{D}_{n_\ell}^0 \boldsymbol{x}(t, \boldsymbol{\mu})$.

The encoder $\boldsymbol{\phi}_\mathrm{e}^{(n_\ell)} : \mathcal{X}^{(n_\ell)} \to \mathcal{Z}$ maps the coarse state space $\mathcal{X}^{(n_\ell)} \subset \mathcal{X}$ into the latent space $\mathcal{Z}$. It consists of multiple graph convolutional layers, each designed to progressively increase the number of features per node. The final layer is a fully connected layer that compresses the output of the last graph convolutional layer into the latent dimensionality. The decoder $\boldsymbol{\psi}_\mathrm{d}^{(n_\ell)} : \mathcal{Z} \to \mathcal{X}^{(n_\ell)}$ follows a mirrored structure. It begins with a fully connected layer that expands the latent representation, followed by graph convolutional layers that successively reduce the number of features per node to reconstruct the coarse state space $\mathcal{X}^{(n_\ell)}$.

The graph convolutional layers are implemented using ChebNet, where the trainable filters are parameterized by weights according to the formulation in (6.4). A schematic representation of the complete surrogate architecture is provided in Fig. 6.3.It is important to note

that the proposed framework is not limited to GCNNs, any data-driven dimensionality reduction technique, whether linear or nonlinear, can be utilized as a substitute. However, GCNNs benefit to a larger extent from an accelerated and simplified learning process due to their comparably high computational costs.

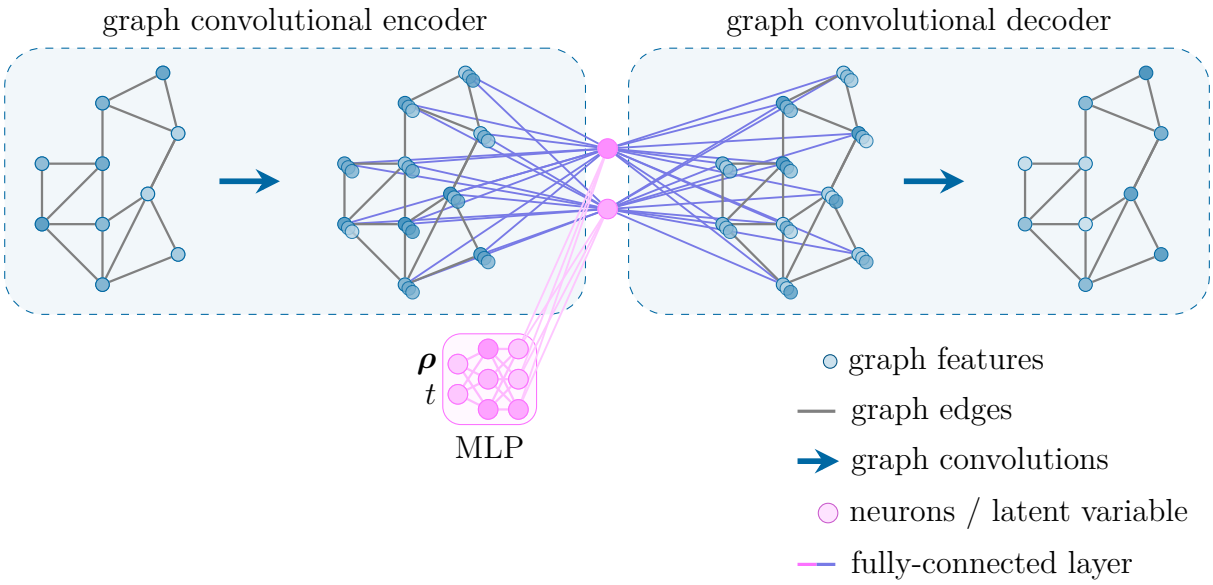


Figure 6.3: The proposed architecture of the surrogate is composed of graph convolutional layers that do not employ pooling but augment the features per node. Additionally, the architecture incorporates fully-connected layers in the middle, which reduce the dimensionality. An additional MLP is incorporated to capture the reduced dynamics.

6.2.4 Transfer Learning

After obtaining a surrogate model on the coarsest level, the surrogate modeling process can be extended to progressively finer levels of resolution. Instead of training each level independently, the finer surrogate builds upon the knowledge already acquired by the coarser surrogate. The transition between coarse and fine graph representations is achieved via down- and upsampling operations to enable the transfer of knowledge between levels of different resolution. In this framework, the already-trained coarse surrogate is fixed, and its output is integrated into the finer surrogate, as depicted in Fig. 6.4. This design allows the finer model to focus exclusively on resolving inaccuracies and capturing behaviors not accounted for by the coarser surrogate. The general architecture of the finer surrogate's encoder, decoder, and MLP adheres to the definitions established for the coarsest surrogate.

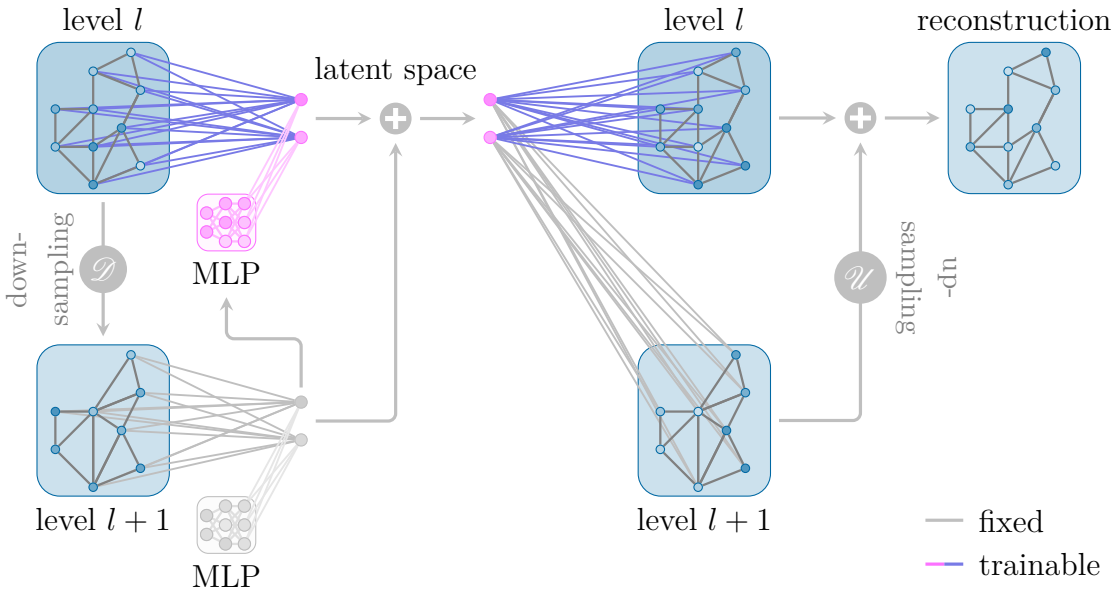


Figure 6.4: Transfer learning step inside the MH surrogate modeling approach. The weights of the coarse network are fixed, so that the finer surrogate only needs to capture not covered aspects.

For the following explanations, let's assume that we are currently training the surrogate $\tilde{\boldsymbol{F}}^{(l)}$ on level l, while $\tilde{\boldsymbol{F}}^{(l+1)}$ represents the already trained coarser surrogate. That means the weights $\boldsymbol{W}^{(l+1)}$ of the latter are fixed, and the optimization focuses solely on updating the weights $\boldsymbol{W}^{(l)}$ of the current surrogate. The first modification compared to the standard modeling scheme for the coarsest surrogate occurs during the encoding of the system state. Instead of relying on a single encoder, the latent state is computed as a combination of outputs from two encoders: the trained and fixed encoder from the coarser level $\boldsymbol{\phi}_{\mathrm{e}}^{(l+1)} : \mathcal{X}^{(l+1)} \to \mathcal{Z}$ and a new trainable encoder $\boldsymbol{\phi}_{\mathrm{e}}^{(l)^*} : \mathcal{X}^{(l)} \to \mathcal{Z}$. The latent variable at the current level is then computed as a combination of their outputs as

$$\begin{aligned} \boldsymbol{z}^{(l)} &\coloneqq \boldsymbol{\phi}_{\mathrm{e}}^{(l)}(\boldsymbol{x}^{(l)}) \\ &\coloneqq \boldsymbol{\phi}_{\mathrm{e}}^{(l+1)}(\mathscr{D}_{n_\ell}^{l}\boldsymbol{x}^{(l)}) + \boldsymbol{\phi}_{\mathrm{e}}^{(l)^*}(\boldsymbol{x}^{(l)}) \\ &= \boldsymbol{\phi}_{\mathrm{e}}^{(l+1)}(\boldsymbol{x}^{(l+1)}) + \boldsymbol{\phi}_{\mathrm{e}}^{(l)^*}(\boldsymbol{x}^{(l)}) \\ &= \boldsymbol{z}^{(l+1)} + \boldsymbol{\phi}_{\mathrm{e}}^{(l)^*}(\boldsymbol{x}^{(l)}). \end{aligned} \tag{6.15}$$

The reconstruction of the state in physical space follows a similar approach. To achieve this, the output of the fixed, trained decoder $\boldsymbol{\psi}_{\mathrm{d}}^{(l+1)} : \mathcal{Z} \to \mathcal{X}^{(l+1)}$ is combined with the output of a newly introduced, trainable decoder $\boldsymbol{\psi}_{\mathrm{d}}^{(l)^*} : \mathcal{Z} \to \mathcal{X}^{(l)}$. These two components are blended into a unified refined decoder $\boldsymbol{\psi}_{\mathrm{d}}^{(l)} : \mathcal{Z} \to \mathcal{X}^{(l)}$, which reconstructs the state at

the current level. The refined decoder computes the reconstructed state as

$$\begin{aligned}\breve{\boldsymbol{x}}^{(l)} &\coloneqq \boldsymbol{\psi}_{\mathrm{d}}^{(l)}(\boldsymbol{z}^{(l)}) \\ &\coloneqq \mathscr{U}_l^{l+1}\boldsymbol{\psi}_{\mathrm{d}}^{(l+1)}(\boldsymbol{z}^{(l)}) + \boldsymbol{\psi}_{\mathrm{d}}^{(l)^*}(\boldsymbol{z}^{(l)}) \\ &= \mathscr{U}_l^{l+1}\breve{\boldsymbol{x}}^{(l+1)} + \boldsymbol{\psi}_{\mathrm{d}}^{(l)^*}(\boldsymbol{z}^{(l)}).\end{aligned} \tag{6.16}$$

This design leverages the previously trained decoder's knowledge while allowing the trainable component to refine and adapt the reconstruction to the higher-resolution state space.

The static and potentially error-prone upsampling matrix used in (6.16) can be replaced with a more flexible and adaptive learnable upsampling approach to further reduce reconstruction errors. In this work, a simple linear fully-connected layer

$$\boldsymbol{x}^{(l)} \approx \boldsymbol{\Theta}_l^{l+1}(\boldsymbol{x}^{(l+1)}; \boldsymbol{W}_{\boldsymbol{\Theta}}^{(l)}, \boldsymbol{b}_{\boldsymbol{\Theta}}^{(l)}) \coloneqq \boldsymbol{W}_{\boldsymbol{\Theta}}^{(l)}\boldsymbol{x}^{(l+1)} + \boldsymbol{b}_{\boldsymbol{\Theta}}^{(l)} \tag{6.17}$$

is employed. The trainable parameters $\boldsymbol{W}_{\boldsymbol{\Theta}}^{(l)}$ and $\boldsymbol{b}_{\boldsymbol{\Theta}}^{(l)}$ correspond to the weights and bias of the layer. This formulation has been shown experimentally to significantly reduce upsampling errors while keeping computational costs minimal. By replacing the original upsampling matrix in (6.16) with the learnable formulation (6.17), the decoder definition used in this work

$$\boldsymbol{\psi}_{\mathrm{d}}^{(l)}(\boldsymbol{z}^{(l)}) = \boldsymbol{\Theta}_l^{l+1}(\boldsymbol{\psi}_{\mathrm{d}}^{(l+1)}(\boldsymbol{z}^{(l)})) + \boldsymbol{\psi}_{\mathrm{d}}^{(l)^*}(\boldsymbol{z}^{(l)}) \tag{6.18}$$

is found.

The refined MLP takes advantage of the output from the previous level as an additional input, enhancing its predictive capabilities. This relationship is expressed as

$$\tilde{\boldsymbol{z}}^{(l)} \coloneqq \boldsymbol{\theta}^{(l)}(\boldsymbol{\mu}, t) = \boldsymbol{\theta}^{(l)*}(\boldsymbol{\mu}, t, \boldsymbol{\theta}^{(l+1)}(\boldsymbol{\mu}, t)) = \boldsymbol{\theta}^{(l)*}(\boldsymbol{\mu}, t, \tilde{\boldsymbol{z}}^{(l+1)}). \tag{6.19}$$

To create the next finer surrogate model $\tilde{\boldsymbol{F}}^{(l-1)}$, the same procedure is repeated but this time with $\tilde{\boldsymbol{F}}^{(l)}$ serving as coarse model.

6.3 Implementation Details

The initial step in implementing the MH approach involves the representation of the model to be approximated as a graph. In the context of FE models, this process is relatively straightforward. The nodes of the FE model can directly serve as nodes of the graph, and the element definitions specify the node connectivity, i.e. the edges $\mathcal{E}$ of the graph and the adjacency matrix $\mathcal{A}$. The physical quantities of these nodes that are to be approximated function as graph signals, that is, they represent the state of the graph.

Table 6.1: Overview of surrogate models showing which data is used and which parameters are optimized.

Model	Reduction Algorithm	Mesh data	Weights
PCANN	Principal Component Analysis	$\boldsymbol{x}$	$\boldsymbol{W}_{\boldsymbol{\theta}}^{\mathrm{PCA}}$
FCAENN	Fully-connected autoencoder	$\boldsymbol{x}$	$\boldsymbol{W}_{\boldsymbol{\theta}}^{\mathrm{AE}}, \boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}^{\mathrm{AE}}, \boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}^{\mathrm{AE}}$
GAENN	Graph convolutional autoencoder	$\boldsymbol{x}$	$\boldsymbol{W}_{\boldsymbol{\theta}}^{\mathrm{GAE}}, \boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}^{\mathrm{GAE}}, \boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}^{\mathrm{GAE}}$
MH1	MH graph conv. autoencoder	$\boldsymbol{x}^{(1)} = \mathscr{D}_1^0 \boldsymbol{x}$	$\boldsymbol{W}_{\boldsymbol{\theta}}^{(1)}, \boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}^{(1)}, \boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}^{(1)}, \boldsymbol{W}_{\boldsymbol{\Theta}}^{(1)}$
MH2	MH graph conv. autoencoder	$\boldsymbol{x}^{(2)} = \mathscr{D}_2^0 \boldsymbol{x}$	$\boldsymbol{W}_{\boldsymbol{\theta}}^{(2)}, \boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}^{(2)}, \boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}^{(2)}, \boldsymbol{W}_{\boldsymbol{\Theta}}^{(2)}$
MH3	Graph convolutional autoencoder	$\boldsymbol{x}^{(3)} = \mathscr{D}_3^0 \boldsymbol{x}$	$\boldsymbol{W}_{\boldsymbol{\theta}}^{(3)}, \boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}^{(3)}, \boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}^{(3)}$

For the following investigations, the kart model from Section 2.2.1 serves as numerical example. The differently resolved representations of it correspond to those shown in Fig. 6.2. The objective of the surrogate modeling process is to find a good proxy for the displacements $\boldsymbol{d} \in \mathbb{R}^{n_{\mathrm{node}} \times n_{\mathrm{c}}}$. Hence, the graph signal for every node consists of $n_{\mathrm{c}} = 3$ channels representing the displacements in the respective spatial directions so that the overall signal is $\boldsymbol{x} \coloneqq \boldsymbol{d}$. The implementation of the MH models follows the previous definition. As the kart depends on three parametric inputs and the time, we reduce it to a latent dimension of $r = 4$.

Multiple other surrogate models are generated following the description of Section 6.2.3 to compare the proposed framework against more conventional approaches. Those surrogates directly operate on the original data instead the downsampled one. In Table 6.1 the relation which surrogate leans on which data is highlighted. Moreover it provides an overview which reduction algorithm is used and which weights are trained, where $\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}$, $\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}$, $\boldsymbol{W}_{\boldsymbol{\theta}}$, and $\boldsymbol{W}_{\boldsymbol{\Theta}}$ represent the weights of the encoder, decoder, MLP, and upsampling network respectively. The superscript in the table indicates to which surrogate model the individual weights belong.

In particular, PCA, a FCAE and a GAE are implemented for the reduction step, while all surrogates utilize an MLP of the same architecture to resolve the task of approximating the latent dynamics. The conventional surrogates are referred to as PCANN, AENN, and GAENN, while the surrogates created with the MH approach are called MH1, MH2, and MH3 (from finest to coarsest surrogate). Further details concerning the architectures and training specifics can be found in Table 6.2. It is noteworthy that graph convolutional networks possess considerably fewer parameters than a comparable multi-layer fully-connected networked architecture.

Table 6.2: Model Architectures. In the case of autoencoder models, the layer structure of the encoder is provided, while the decoder follows a similar architecture but in reverse order.

Model	Hyperparameter Category	Details
PCA	# Parameters	111 768
All NNs	Optimizer	Adam, Learning Rate: $1e-3$
	Training	Epochs: 400, Batchsize: 128
	Activation fcn.	elu
	Output act. fcn.	linear
	Regularization	Select best weights w.r.t. validation data
FCAE	# Parameters	11 246 854
	Layer sizes	$27942 \times 200 \times 80 \times r$
GAE	# Parameters	2 023 067
	Layer sizes	$(9314\times3) \times (9314\times6) \times (9314\times12) \times (9314\times24) \times r$
	Graph convolutions	ChebNet of order $n_{\text{cheb}} = 3$
MH1	# Parameters	253 987
	Layer sizes	$(1165\times3) \times (1165\times6) \times (1165\times12) \times (1165\times24) \times r$
	Graph convolutions	ChebNet of order $n_{\text{cheb}} = 3$
MH2	# Parameters	65 419
	Layer sizes	$(292\times3) \times (292\times6) \times (292\times12) \times (292\times24) \times r$
	Graph convolutions	ChebNet of order $n_{\text{cheb}} = 3$
MH3	# Parameters	18 115
	Layer sizes	$(73\times3) \times (73\times6) \times (73\times12) \times (73\times24) \times r$
	Graph convolutions	ChebNet of order $n_{\text{cheb}} = 3$
MLP	# Parameters	8 900
	Layer sizes	$4 \times 64 \times 64 \times 64 \times r$

6.4 Results

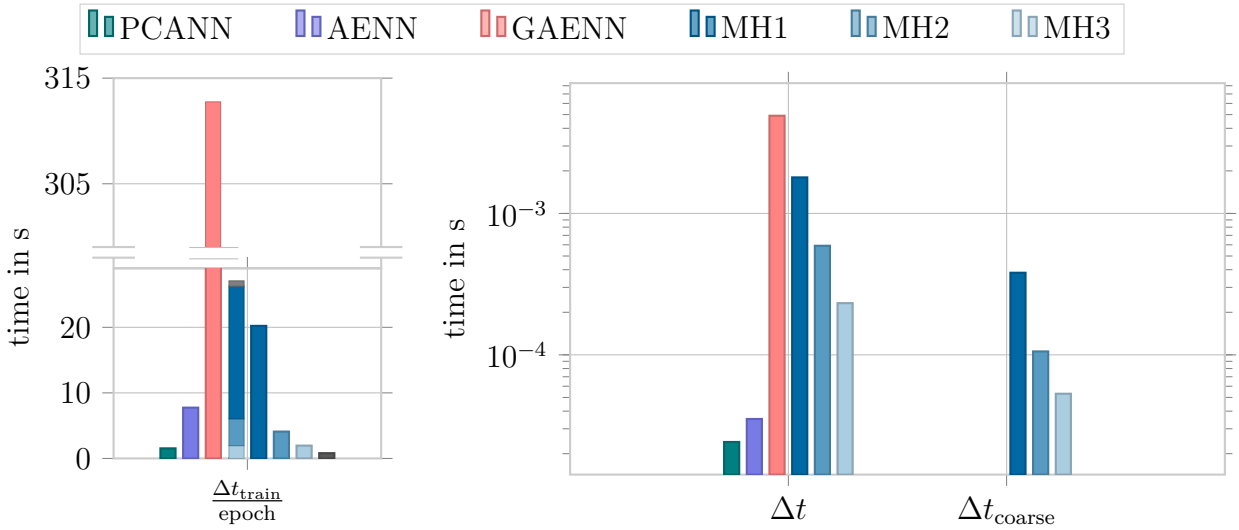
A rigorous evaluation of the developed surrogate models is conducted in the following. This assessment encompasses an analysis of the training phase, an evaluation of approximation accuracy in both the reduced and original representations, and a study of computational efficiency. To ensure a systematic and objective validation of the results, the error measures introduced in Section 5.1.1 are employed.

6.4.1 Training Comparison between Fine and Coarse Models

One severe limitation of the employed graph convolution formulation lies in its computational expense, primarily due to the recursive computation of Chebyshev polynomials. This computational burden significantly increases the training time required for a graph convolutional surrogate model when applied to the full model, as illustrated in Fig. 6.5a. In contrast, when the surrogate is constructed using the MH approach, the tides turn. The training time is substantially reduced, to the extent that the model operating on the coarsest representations achieves faster training times than the conventional FCAE on the full model. Furthermore, even when the training times of all three MH levels used for the kart example are aggregated, the total time remains within a comparable order of magnitude and is more than ten times faster than the GAENN model trained on the full model.

When considering the computation time required for a single prediction of the surrogates, denoted as Δt, a similar trend emerges, as shown in Fig. 6.5b. The GAENN exhibits the highest computational cost. However, our proposed approach effectively mitigates this issue. Among the surrogates, the model employing PCA for dimensionality reduction achieves the fastest predictions, as reconstructing the fine physical space from the reduced representation involves only a single matrix multiplication. In the case of the MH approach, both training and computational times naturally increase with the addition of each level, as the resolution progressively improves. Moreover, the time required to obtain a prediction for the fine original representation is not excessively higher than that for the coarse representations, due to the chosen upsampling procedure described in (6.8). An unnecessary increase in computing time can therefore be avoided by using sparse matrix multiplication.

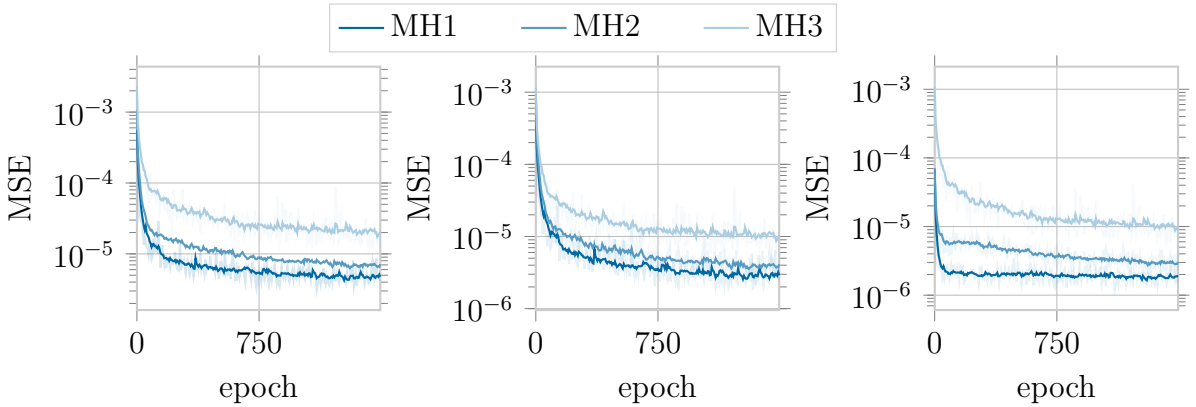
Beyond computation times, the training process of the MH models offers additional insights, particularly with regard to transfer learning. The progression of the loss during training, as illustrated in Fig. 6.6, reveals that the loss values decrease significantly with each successive refinement of the models. This indicates that the information preserved in the coarser models aids the finer models in enhancing their performance, effectively preventing the redundant learning of already known structures. Notably, the reconstruction task benefits substantially from transfer learning, as highlighted in Fig. 6.6c. Moreover, the overall approximation improves consistently with each refinement level, as demonstrated in Fig. 6.6b. These findings underscore the efficacy of the transfer learning approach in leveraging prior knowledge to enhance both reconstruction accuracy and overall model performance.



(a) Training time per epoch using 9696 samples and a batch size of 32.

(b) Averaged computing time per prediction on full model (with upsampling) and on the coarse levels (without upsampling)

Figure 6.5: Computational time required to train the models (a) and to compute a prediction (b) for the kart model on the fine and the coarse mesh using the proposed MH approach and standard approaches. The dark gray bar in (a) is the time required for the down- and upsampling computations.



(a) Loss (6.14a)

(b) Approx. loss (6.14b)

(c) Reconstr. loss (6.14c)

Figure 6.6: The training history for the overall loss (a), the approximation loss (b), and the reconstruction loss (c) of the MH models for the racing kart example is presented. To improve clarity, the loss functions are smoothed, while the actual values are depicted with transparency in the background. The results clearly demonstrate a significant reduction in loss with each refinement level compared to the preceding coarser level. This trend highlights the effectiveness of the MH approach in leveraging progressive refinements to enhance model performance across multiple levels.

6.4.2 Evaluation on Coarse Levels

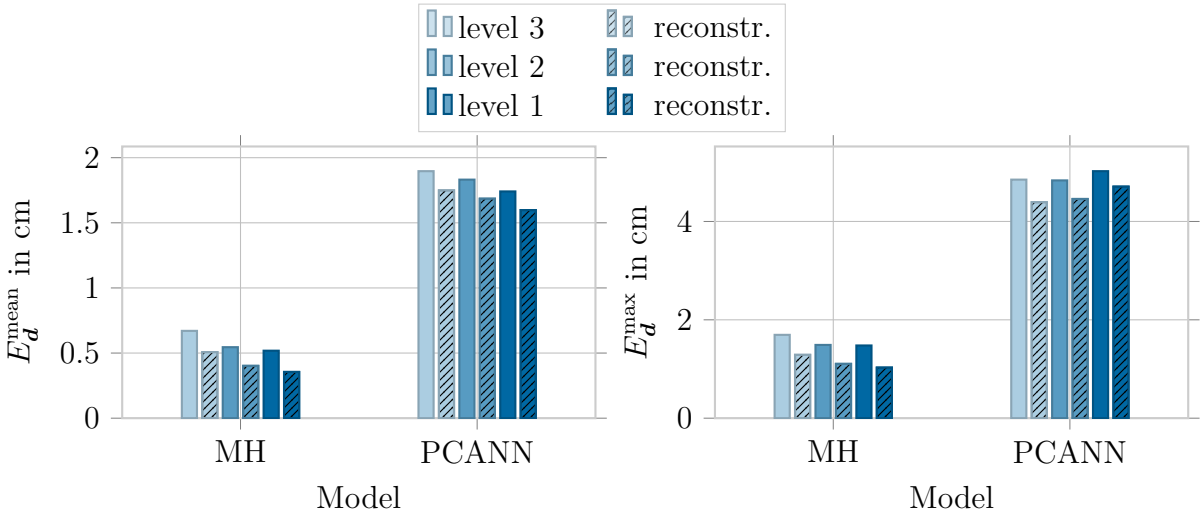
Before the surrogate models are compared against each other within the original discretization of the model, there are compelling reasons to consider the performance within the coarse discretizations. On the one hand, the MH models are explicitly designed to operate within the coarse representations, making their evaluation in this context highly relevant. On the other hand, evaluating performance at the coarse levels eliminates the error introduced by the upsampling process into the original space, thereby providing a clearer assessment of the models' capabilities at these resolutions. Additionally, this intermediate evaluation facilitates a direct comparison of the graph convolutional MH models with a PCANN model fitted at each resolution level. Such a comparison not only highlights the relative advantages of the MH models but also substantiates their utility and justifies their adoption across multiple levels of resolution.

When analyzing the node distance error of the MH models and PCANN surrogates across different discretizations as done in Fig. 6.7, two key observations emerge. First, all MH models consistently outperform the PCA-based surrogate. This result demonstrates that the graph convolutional architecture is highly effective, thereby justifying its adoption over alternative architectures with similar applicability. Second, for the MH models, the error decreases progressively with each additional refinement level. For an error measured at the original fine resolution, this reduction is expected, as the upsampling error diminishes with each level. However, on the coarse discretizations, this trend clearly indicates that transfer learning plays a pivotal role in reducing the error at each level, further enhancing the performance and efficiency of the MH models.

An additional analysis investigates which dynamic effects are learned at each refinement level. Specifically, the contribution of a given level can be assessed by looking at the difference between its approximation and the approximation of the next coarse level, i.e.

$$\mathscr{U}_0^l \tilde{\boldsymbol{x}}^{(l)} - \mathscr{U}_0^{l+1} \tilde{\boldsymbol{x}}^{(l+1)} = \mathscr{U}_0^l \boldsymbol{\psi}_{\mathrm{d}}^{(l)}(\boldsymbol{\theta}^{(l)}(\boldsymbol{\mu}, t)) - \mathscr{U}_0^{l+1} \boldsymbol{\psi}_{\mathrm{d}}^{(l+1)}(\boldsymbol{\theta}^{(l+1)}(\boldsymbol{\mu}, t)). \tag{6.20}$$

However, it should be noted that this does not permit a completely isolated perspective on the individual contributions, as the specific interdependencies of the surrogate models are not resolved in this manner. Nevertheless, it constitutes a straightforward method for determining the attributes that are newly learned at each level. The corresponding Fig. 6.8 illustrates the contributions of each level. The results reveal that the global dynamic behavior is predominantly captured by the coarsest surrogate (level 3). At this level, significant phenomena, such as the strong deflection of the front fork and the rotation of the entire kart, are accurately modeled. In contrast, the finer levels (level 2 and level 1) focus on capturing minor deformations, particularly in areas where the coarser levels lack sufficient degrees of freedom.



(a) Mean error (b) Maximum error

Figure 6.7: Mean (a) and maximum (b) averaged nodal displacement errors on differently resolved meshes, comparing the proposed MH approach with a standard PCANN surrogate. Errors are evaluated both for the plain reconstruction of the kart geometry and for the approximations of the surrogates. For context, note that the maximum nodal displacements in the reference simulations reach up to 150 cm.

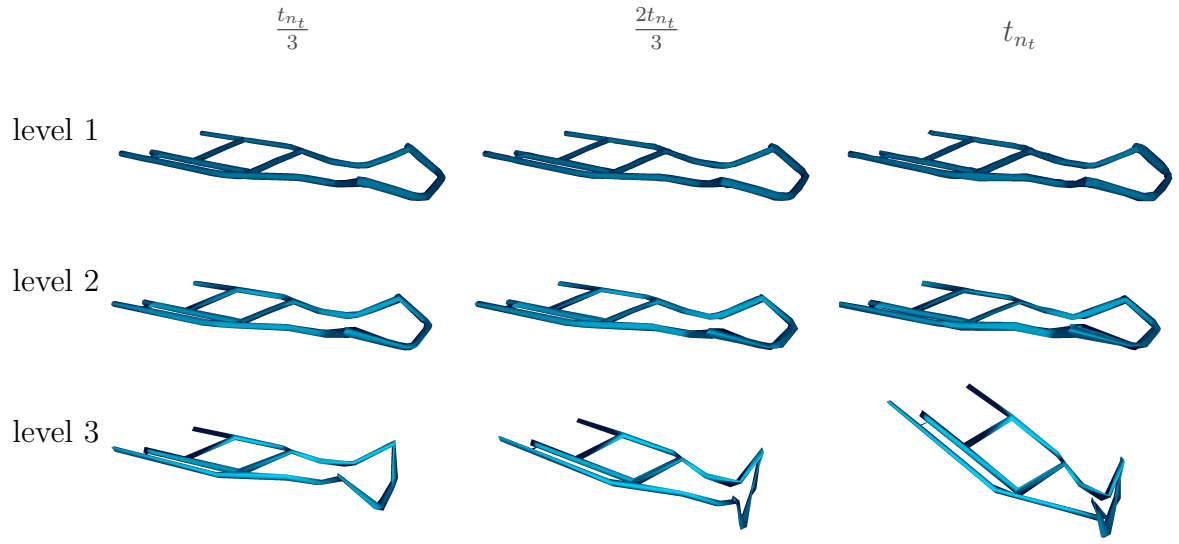


Figure 6.8: Learned behavior across different resolution levels. The global dynamic behavior is predominantly captured at the coarsest level. This learned behavior is subsequently transferred to finer levels, where only minor adjustments and localized refinements are performed to address residual errors.

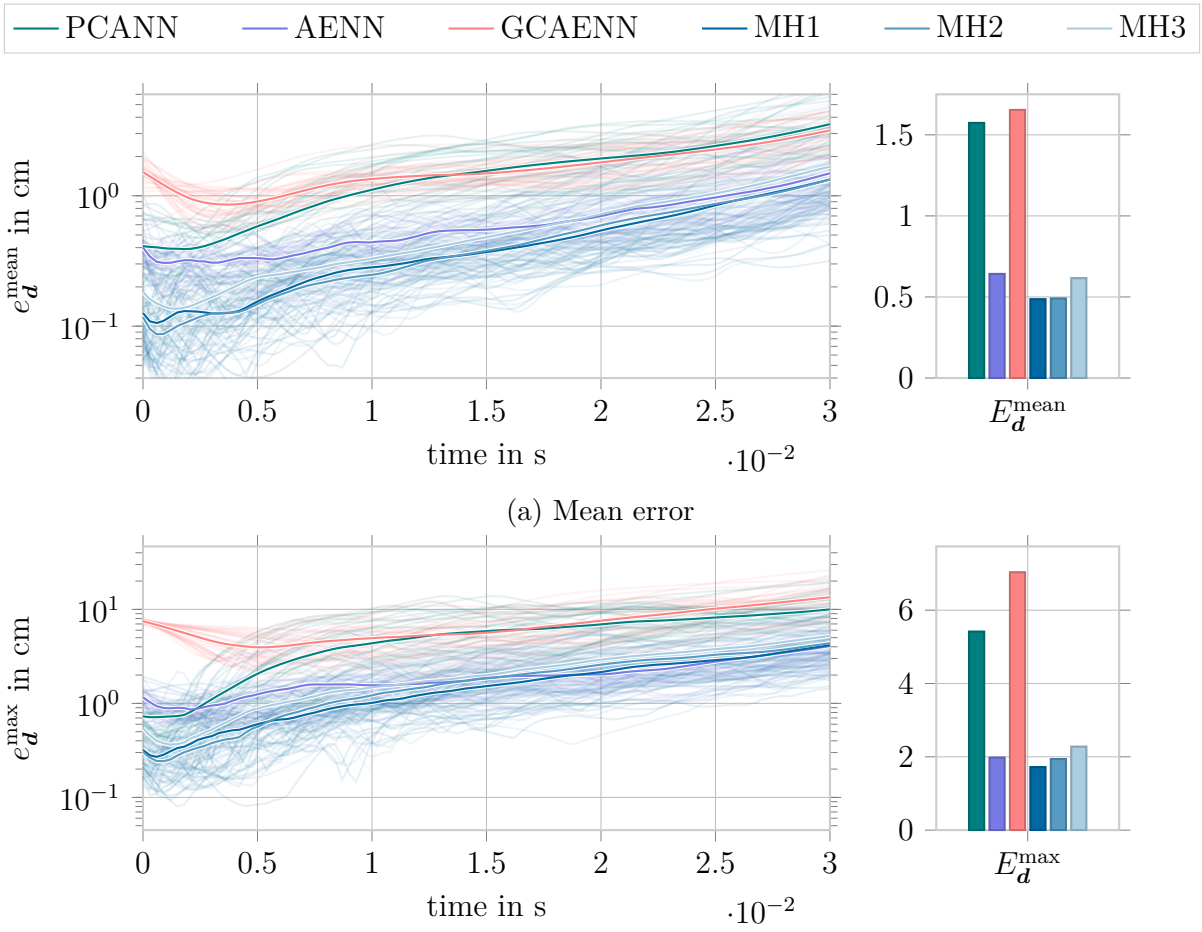
6.4.3 Approximation Quality

In the concluding evaluation of surrogate models, performance measures are consistently measured within the original discretization of the model. This approach enables a direct comparison between the MH models and models trained on the original data. For the upsampling of the coarse approximation to the original discretization, the states are upsampled using the static upsampling matrices $\mathscr{U}_l^0$. This time, all surrogate modeling approaches, namely PCANN, AENN, GAENN, MH1, MH2, and MH3, compete against each other. The respective mean and maximum node distance errors over time are displayed in Fig. 6.9. Notably, the graph convolutional autoencoder-based surrogate without the MH structure fails to capture the system dynamics effectively, resulting in the largest errors among the models. Similarly, the surrogate employing linear dimensionality reduction via PCA faces significant challenges in approximating intervals of high dynamic activity. This limitation arises because the reduced basis, with only $r = 4$ vectors, lacks sufficient expressiveness to capture the complex deformations observed in the simulations.

The advantages of nonlinear dimensionality reduction are demonstrated by the AENN surrogate model, based on a conventional autoencoder. It demonstrates a high degree of efficacy in capturing dynamic phenomena over most segments of the simulation. However, it exhibits a comparatively higher error, especially at the beginning in phases of low dynamics. The coarsest MH model is already able to outperform the AENN model on average w.r.t the mean Euclidean distance error, which decreases further at each subsequent finer level. That said, the performance improvement diminishes as more refinement levels are added. For the maximum error, the observations differ. The coarsest MH model does not surpass the AENN model, and only the finer models achieve superior performance. Interestingly, the maximum error continues to decrease significantly with additional levels, contrary to the trend observed for the mean error. This indicates that while the overall performance gains may plateau at finer resolutions, areas prone to high error still benefit substantially from the increased detail. Key performance metrics summarizing these results are presented in Table 6.3, offering a concise overview of the primary findings.

6.5 Discussion

The findings demonstrate that the proposed MH surrogate modeling framework is effective for constructing efficient yet accurate Reduced Order Models (ROMs) across diverse spatial resolutions for structural dynamical systems like the kart frame example. This approach captures the transient dynamics, including significant plastic deformations of the kart's frame caused by impact, while outperforming conventional methods in terms of accuracy and maintaining competitive computational efficiency. Moreover, the number of parameters is on par with state-of-the-art linear counterparts and surpasses conventional



(a) Mean error

(b) Maximum error

Figure 6.9: Mean (a) and maximum (b) node displacement errors over time for all test simulations achieved by various surrogate models are presented. The individual test trajectories are drawn transparently, while the mean value of all test simulations is shown opaquely.

Table 6.3: Performance measures: The time specifications provided in brackets indicate the total aggregate training time across all levels for the MH models. For the PCANN model, these values correspond to the training time for the MLP. This comparison highlights the relative computational efficiency and resource requirements of each surrogate modeling approach.

	PCANN	**AENN**	**GAENN**	**MH1**	**MH2**	**MH3**
$\frac{\Delta t_{\text{train}}}{epoch}$ in s	- (1.54)	7.75	312.75	20.22 (27.08)	4.10 (6.27)	**1.96**
Δt in ms	**0.02**	0.04	0.91	0.38	0.11	0.05
$E_{\boldsymbol{d}}^{\text{mean}}$ in cm	1.57	0.64	1.65	**0.49**	0.49	0.62
$E_{\boldsymbol{d}}^{\text{max}}$ in cm	5.42	1.98	7.04	**1.72**	1.94	2.28

nonlinear data-driven approaches.

The framework achieves this by operating on multiple representations of the system at varying resolutions rather than relying solely on a single high-resolution discretization. This hierarchical structure naturally facilitates the approximation of multi-scale effects, with global dynamics learned at coarse resolutions and microscale dynamics captured at finer ones. Additionally, low-resolution approximations ease the learning process and improve the accuracy of medium- and fine-resolution models by transferring knowledge across levels, enabling finer models to focus only on residuals. These coarse-level predictions remain visually interpretable and computationally efficient, making them suitable for graphical applications in hardware-constrained scenarios. Importantly, the upsampling of the coarse-level predictions is computationally lightweight, relying on sparse matrix multiplications or adaptive upsampling networks.

The results further indicate that the global dynamic behavior observed in the crash scenario is already captured at the coarsest surrogate level, with finer surrogates focusing on microscale effects. Notably, accuracy improves with each refinement level, even within the coarse domains. This trend is also reflected in the training loss progression, where finer models achieve significantly faster convergence. These observations underscore the role of transfer learning in the surrogate modeling process.

Moreover, the hierarchical learning process decreases the associated computational burdens for any surrogate as it eliminates the need to work with the original fine resolution data and creates multiple models with varying memory and computational demands, all operating in visually and physically interpretable domains. Working on coarse representations has the potential to speed up surrogate modeling in general. The lower associated computational costs lead to faster training and computation times. Hence, computationally expensive hyperparameter tuning can be significantly accelerated even for surrogate modeling schemes that aim to operate on the original discretization later on.

While the proposed approach offers significant advantages, it is accompanied by certain drawbacks and limitations. First, the method requires knowledge of the system's internal geometrical structure, meaning that data alone is insufficient. However, in most cases, this geometry can be readily exported from commercial software with minimal effort. Second, the mesh simplification process employed in the method is only based on spatial criteria, which may result in the loss of other quantities of interest unless explicitly accounted for during mesh simplification. Additionally, the mesh simplification process itself adds computational overhead to the offline phase, although this additional effort is negligible compared to the overall training time required for the networks. Third, the use of the MH architecture and graph convolution operations introduces numerous design choices and hyperparameters, increasing the complexity of the surrogate modeling process compared to simpler methods like PCA combined with neural networks. Despite this added complexity, the MH models consistently outperform conventional methods, even without extensive

hyperparameter tuning. Finally, similar to other nonlinear dimensionality reduction techniques, the benefits of this approach are most pronounced when reducing the system to its intrinsic dimensionality. For cases involving larger latent spaces, traditional linear methods can still provide competitive results, narrowing the performance gap.

An important aspect to discuss is the selection of graph convolutions for dimensionality reduction within the MH framework. While the MH approach is inherently flexible and compatible with other data-driven reduction methods, the use of GCNNs aligns naturally with the graph structure occurring during the mesh simplification process. As demonstrated in the results, the graph convolution-based surrogate operating on the coarse mesh significantly outperforms linear reduction techniques, even in the absence of transfer learning at this level. This highlights the effectiveness of graph convolutions in capturing complex system dynamics. Furthermore, the utilized graph convolutions benefit significantly from the hierarchical structure as the respective operations are computationally expensive leading to particularly impactful computational time savings. Moreover, the parameter-sharing mechanism in graph convolutions reduces the number of trainable parameters required, while maintaining higher expressiveness in capturing data features within the framework. Interestingly, a graph convolution-based surrogate applied directly to the original fine mesh failed to adequately approximate the system. This limitation is likely attributable to several factors, including oversmoothing issues [ChenEtAl20a, RuschBronsteinMishra23], challenges in transporting information across distant nodes in a fine mesh [AlonYahav20], and spectral bias [RahamanEtAl19]. These challenges underscore the critical role of the MH framework, which not only facilitates a more effective learning process but, in this case, makes successful learning possible in the first place.

The proposed framework relies on the availability and quality of high-fidelity data, making its performance sensitive to the underlying data's accuracy and coverage. Consequently, extrapolation beyond the training domain presents significant challenges. To address this limitation, integrating low-fidelity models into the surrogate modeling process offers a promising direction for future research. Currently, all hierarchical models in the MH approach are derived exclusively from high-fidelity data. However, the incorporation of low-fidelity FE models could extend the surrogate's applicability to parameter domains outside the original training set. These low-fidelity models, e.g. derived from coarser meshes, could be used to supplement the surrogate models by leveraging multi-fidelity techniques. For instance, low-fidelity simulations could provide inexpensive predictions that enhance high-fidelity results by learning and correcting residuals, as demonstrated in recent multi-fidelity frameworks [KastGuoHesthaven20, ContiEtAl23b, DemoTezzeleRozza23].

Additionally, the proposed architecture offers significant flexibility for customization and extension. Refinement could be selectively applied to spatial regions where errors are higher, enabling localized improvements without additional global computational costs.

Alternatively, surrogate models for all resolution levels could be trained simultaneously within a unified latent space, employing distinct encoder and decoder networks for each level. This adaptability underscores the potential of the MH framework to efficiently address a broad spectrum of applications.

Part III

Latent Discovery of Reduced Order Models

Chapter 7

Structure-preserving Latent Discovery

Ich tu da nichts dagegen
Ich tu da was dafür
Nämlich einfach leben
Und das empfehle ich auch dir

Kim Hoss, Underboob Sweat

In Part II of this thesis, we explored Latent Approximation of Dynamics (LADy), i.e. black-box surrogate modeling techniques to capture the behavior of parametrized structural dynamical systems directly from high-dimensional, High-Fidelity (HF) data within a low-dimensional latent space. In the following part, we turn our attention to alternative strategies that aim not only to replicate system behavior but to explicitly uncover the governing equations of the underlying HF system. These methods, framed under the umbrella term Latent Discovery of Dynamics (LDD), enable the construction of descriptive and predictive Reduced Order Models (ROMs) grounded in interpretable dynamics extracted directly from data.

To begin with, this chapter introduces a structure-preserving method to identify ROMs. For this, the Port-Hamiltonian (PH) formulation is utilized that can guarantee specific system-theoretical properties in the identified model and is particular useful for multi-physics problems (cf. Chall. 5) as multiple domains can be connected via so called ports. In contrast, the subsequent chapter copes with generative models to deal with the challenge of noise in data (cf. Chall. 6) and the reliability of the identified models via Uncertainty Quantification (UQ).

An explicit formulation of system equations offers distinct advantages. By revealing the governing principles, these models facilitate direct physical interpretation and enable

rigorous analysis using classical system theory. Consequently, such approaches specifically address the challenge of ensuring physical consistency in data-driven discovery methods (cf. Chall. 3). Furthermore, by incorporating physically consistent representations, these methods enhance extrapolation capabilities, leading to ROMs that generalize effectively beyond the observed data.

Moreover, explicitly identifying the governing system of equations enables additional post-processing steps. One such application is the use of identified models in continuation methods, as demonstrated in [ContiEtAl23a]. These numerical techniques, which are also referred to as path-following methods, facilitate the tracking of solution branches under parameter variations, allowing for the identification of bifurcations and instability points without requiring the transient phase to be simulated for each parameter change. By leveraging the structural similarity between neighboring parameter configurations, continuation methods significantly reduce the computational cost associated with parameter sweeps. This efficiency is particularly beneficial for identifying critical points, which are of paramount importance in engineering applications due to the distinctive structural behavior observed in their vicinity.

In the context of continuation methods, the computation of critical points can be accelerated even to a further extent using surrogate models as we demonstrated in [StraußEtAl24]. Instead of employing traditional path-following techniques, the method proposed in this publication utilizes surrogate models to provide an initial estimate for the exact computation of critical points. By replacing the iterative tracking process with a learned predictive model the computational overhead is reduced while maintaining accuracy in determining structural instabilities and bifurcations.

In general, system identification faces significant challenges when applied to high-dimensional systems, just like other dynamical approximation methods. This is particularly challenging when determining meaningful state variables (cf. Challs. 1 and 2). The vast number of possible interactions within high-dimensional state spaces complicates the identification process, often rendering direct modeling infeasible. To address this challenge, many approaches perform the identification of dynamics in a low-dimensional latent space, where efficient ROMs can be constructed. However, this necessitates the simultaneous discovery of both the reduced state variables and the latent dynamical model governing the observed system behavior. To formalize this procedure and align it with the overarching objective stated at the beginning of this thesis, we reformulate the problem setup (5.1) in the context of LDD.

Let us assume the HF data is produced from a high-dimensional system of Ordinary Differential Equations (ODEs) as described in (2.3), where the system dynamics is governed by an unknown function $\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{\mu}) : \mathcal{T} \times \mathcal{X} \times \mathcal{P} \to \mathcal{X}$ that depends on the time $t \in \mathcal{T}$, the system states $\boldsymbol{x} \in \mathcal{X}$, and some parameters or forcing terms $\boldsymbol{\mu} \in \mathcal{P}$. Then we seek an efficient ROM by identifying a dimensionality reduction mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{Z}$

and a reconstruction mapping $\boldsymbol{\psi} : \mathcal{Z} \to \mathcal{X}$ along with a dynamical description of the latent system $\tilde{\boldsymbol{f}} : \mathcal{T} \times \mathcal{Z} \times \mathcal{P} \to \mathcal{Z}$ that expresses $\boldsymbol{f}$ in terms of the new set of latent coordinates. The dynamics can then be evolved from a latent representation of the initial state $\boldsymbol{z}_0 = \boldsymbol{\phi}(\boldsymbol{x}_0)$. This can be summarized in the reformulated optimization goal

$$\begin{aligned} \min_{\tilde{\boldsymbol{f}},\boldsymbol{\phi},\boldsymbol{\psi}} \quad & \frac{1}{n_t n_{\text{sim}}} \sum_{t\in\mathbb{T}} \sum_{\boldsymbol{\mu}\in\mathbb{P}} L(\boldsymbol{x}(t,\boldsymbol{\mu}), \tilde{\boldsymbol{x}}(t,\boldsymbol{\mu})) \\ \text{s.t.} \quad & \dot{\boldsymbol{x}} = \boldsymbol{f}(t,\boldsymbol{x},\boldsymbol{\mu}), \quad \boldsymbol{x}_0 = \boldsymbol{x}(t_0,\boldsymbol{\mu}) \\ & \dot{\boldsymbol{z}} = \tilde{\boldsymbol{f}}(t,\boldsymbol{z},\boldsymbol{\mu}), \quad \boldsymbol{z}_0 = \boldsymbol{\phi}(\boldsymbol{x}_0) \\ & \tilde{\boldsymbol{x}} = \boldsymbol{\psi}(\boldsymbol{z}), \end{aligned} \tag{7.1}$$

where the reduced-order dynamical system, owing to its lower dimensionality, is expected to significantly reduce the computational burden associated with solving the corresponding system of ODEs. Recall that $L : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is some objective function that measures the approximation quality of the identified surrogate model.

To tackle this optimization goal, recent advancements in data-driven modeling have facilitated the direct extraction of governing equations within a latent space. In particular, autoencoder-based methods have been integrated with Sparse Identification of Nonlinear Dynamics (SINDy) for system identification [ChampionEtAl19], with subsequent extensions to accommodate partial measurements [BakarjiEtAl23] and multi-fidelity modeling [ContiEtAl23b]. Other methodologies employ similar frameworks but do not promote sparsity in the discovered dynamics. For instance, [FriesHeChoi22] presents an alternative framework for the identification of parametric latent space dynamics that omits sparsity regularization, while [BonnevilleEtAl23] refines this methodology by integrating Gaussian Processes (GPs) to interpolate latent-space ODEs.

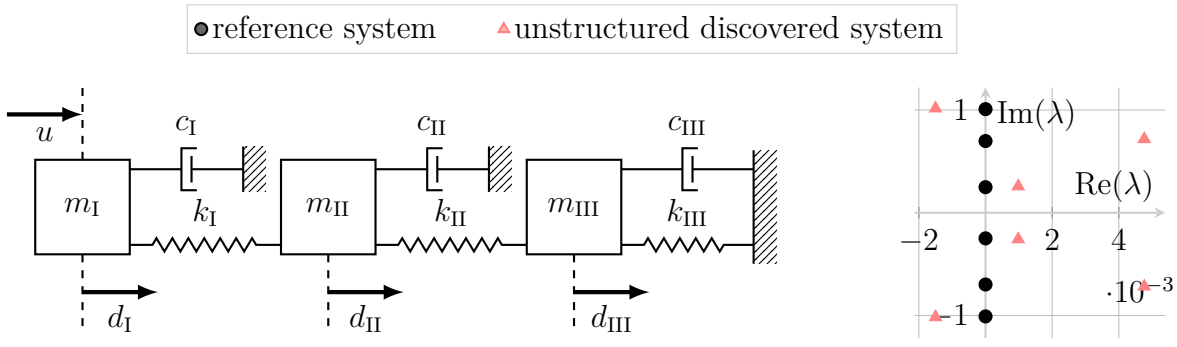
Moreover, other approaches seek to identify effective coordinate systems for the description of the dynamics. For instance, certain methods leverage Koopman operator theory to construct representations in which originally nonlinear dynamics appear approximately linear [LuschKutzBrunton18, BruntonKutz22]. Other approaches aim to infer hidden state variables directly from observational data, such as video recordings [ChenEtAl22a], thereby constructing implicit state-space representations without requiring explicit knowledge of the underlying governing equations. For a comprehensive review of parsimonious model discovery techniques, we refer to [KutzBrunton22, BruntonKutz23]. However, many such approaches overlook the opportunity to incorporate the intrinsic structural properties of the system they seek to identify.

Adding Structure to Systems Physical systems typically belong to a specific class characterized by fundamental properties such as conservation laws, symmetries, or intrinsic structural constraints. Ideally, a discovered model should adhere to the same class, thereby preserving the original structure and maintaining the fundamental properties of the original

system. To achieve this, prior knowledge is often embedded directly into the learning process through specialized model architectures, a concept known as inductive bias. This approach constrains the discovered model to a predefined class of physically consistent representations, ensuring that critical properties are inherently maintained. Enforcing such constraints ensures that learned representations align with established physical principles.

Example 2. *To illustrate the importance of incorporating prior knowledge and structural constraints in the learning process, we consider a simple mechanical example: a Mass-Spring-Damper (MSD) chain, as depicted in Fig. 7.1a. For this demonstration, the damping values are set to zero, ensuring that the system remains marginally stable. While further details on this numerical example are provided in the numerical experiments section (Section 7.3), they are not critical for the present discussion. Using simulation data from this system, we identify a Linear Time-Invariant (LTI) model of the form* $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u$ *through an approach similar to the one introduced in this chapter, but without enforcing any structural constraints on the system matrix* $\boldsymbol{A}$*. The implementation details are omitted here, as the primary objective is to emphasize the qualitative effects of disregarding structural properties.*

As illustrated in Fig. 7.1b, the identified system exhibits multiple eigenvalues with positive real parts, implying instability—contrary to the marginal stability of the original system. As a result, simulations of this identified system may diverge, leading to qualitatively incorrect and physically implausible outcomes. While this example is deliberately designed to highlight the risks of neglecting inherent system structure, it effectively demonstrates the potential consequences of failing to preserve key physical properties. Conversely, it also underscores the advantages of incorporating structural constraints, ensuring both stability and physical consistency in data-driven system identification.



(a) A one-dimensional mass-spring-damper chain with 3 masses.

(b) Eigenvalues $\boldsymbol{\lambda}$.

Figure 7.1: A mass-spring-damper chain (a) that serves as a simple mechanical example for the model discovery purposes. The eigenvalues of the reference and the identified unstructured (LTI) system are shown in (b) and provide information about the identified system behavior.

In this chapter, we focus on the data-driven discovery of PH systems, which offer an energy-based modeling framework particularly well-suited for multi-physical systems while ensuring key system-theoretical properties such as passivity, stability, and boundedness of solutions under mild assumptions. The framework, that we introduced in [RettbergEtAl24], enables the identification of physically consistent, low-dimensional models while preserving key system-theoretical properties inherent to the original system.

For this, high-dimensional state data is transformed nonlinearly into the low-dimensional latent space using an Autoencoder (AE). Within this latent space, a linear PH system is identified using our newly introduced Port-Hamiltonian Identification Network (PHIN). By construction, the discovered system inherently satisfies the fundamental PH properties, ensuring passivity and stability. The coordinate transformation learned by the autoencoder and the PH system dynamics learned by PHIN are jointly optimized to ensure that the latent dynamics accurately capture the observed system behavior while adhering to the PH formulation. The whole procedure is referred to as Autoencoder-based Port-Hamiltonian Identification Network (APHIN). Despite the linearity of the identified latent model, it remains capable of representing nonlinear systems, as the autoencoder can handle the system's nonlinearity within the coordinate transformation. A schematic representation of the overall workflow is provided in Fig. 7.2.

The key contributions of the presented framework include:

1. A structure-preserving, data-driven model discovery framework is developed for PH systems, that ensures that the discovered models retain fundamental physical properties.

2. It is shown that, despite the nonlinear coordinate transformation introduced by the autoencoder, critical system-theoretical properties are preserved and correctly transferred from the low-dimensional latent space back to the original high-dimensional system.

3. The proposed approach is validated on multiple benchmark systems, including a low-dimensional parametric mechanical system, a nonlinear pendulum, and a high-dimensional thermomechanical system.

The following sections provide an introduction to PH systems, outlining their fundamental properties and advantages for structured modeling. This is followed by a detailed explanation of the structure-preserving model discovery method, PHIN and its integration within an autoencoder framework. Finally, the effectiveness of the proposed approach is illustrated through numerical examples.

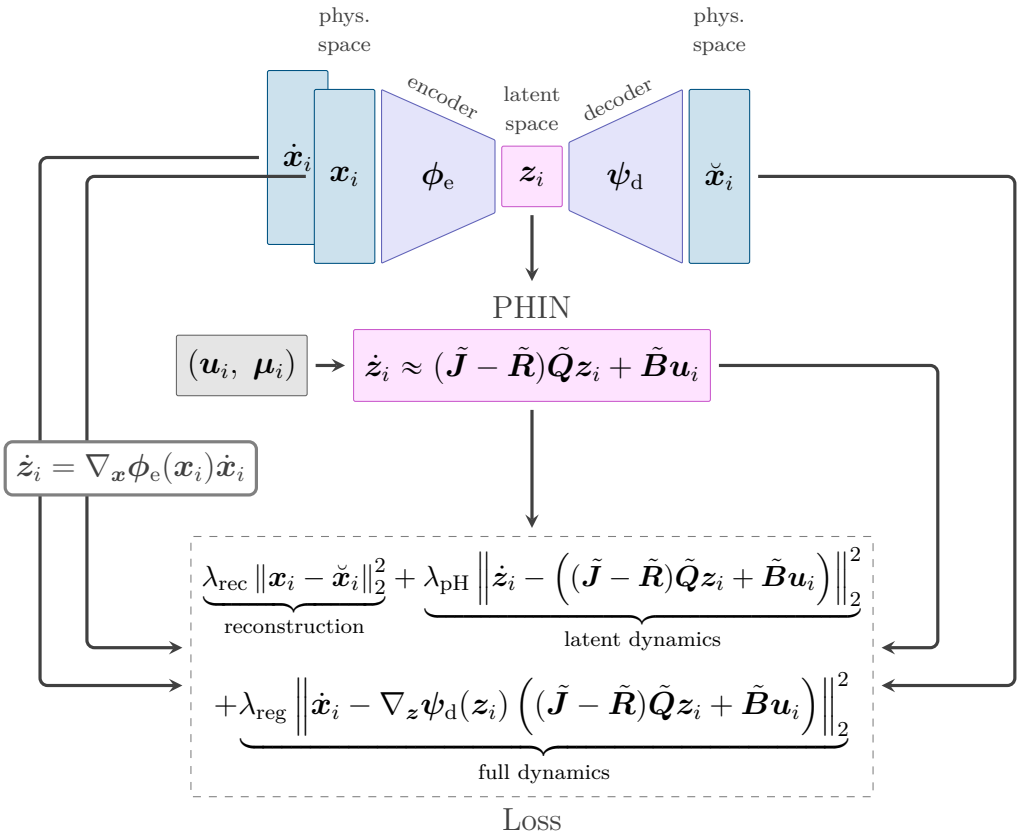


Figure 7.2: Schematic illustration of the APHIN approach: An autoencoder identifies a low-dimensional manifold on which a PH system is identified.

7.1 Port-Hamiltonian Systems

PH systems provide an energy-based modeling framework in which energy functions can be used to prove desirable system-theoretical properties and to interpret system dynamics. Compared to classical energy-based approaches, such as Lagrangian and Hamiltonian mechanics, PH systems offer distinct advantages by naturally incorporating both dissipative effects and external interactions via ports. To account for these extensions, PH systems replace the usual energy conservation through a dissipation inequality, ensuring that the system's energy remains bounded by the energy transferred through the ports. This formulation is particularly advantageous for network modeling, as it enables the physically consistent interconnection of different physical domains [SchaftJeltsema14, MehrmannUnger23, DuindamEtAl09].

A fundamental advantage of this framework is its closure under interconnection: The interconnection of multiple PH systems inherently results in another PH system. This makes them particularly favorable for modular multi-physical modeling, see e.g. [RettbergEtAl23]. Moreover, the explicit definition of ports facilitates hierarchical modeling, and pro-

vides a structured interface for integrating controllers. Several notable control methodologies leverage the properties of PH systems, including control through interconnection [Schaft20], interconnection and dissipation assignment passivity-based control [OrtegaGarcíaCanseco04], and optimization-based approaches to measure the distance to instabilities [GillisMehrmannSharma18, GillisSharma17].

The general framework of PH systems allows for various representations. Among others, there are linear time-variant, descriptor, nonlinear and infinite-dimensional PH systems [SchaftJeltsema14, MehrmannUnger23, DuindamEtAl09]. In the present work, we identify (parametric) LTI PH systems, which are defined by an initial value problem

$$\begin{aligned} \dot{\boldsymbol{x}}(t,\boldsymbol{\mu}) &= \big(\boldsymbol{J}(\boldsymbol{\mu}) - \boldsymbol{R}(\boldsymbol{\mu})\big)\boldsymbol{Q}(\boldsymbol{\mu})\boldsymbol{x}(t,\boldsymbol{\mu}) + \boldsymbol{B}(\boldsymbol{\mu})\boldsymbol{u}(t) \\ \boldsymbol{x}(0,\boldsymbol{\mu}) &= \boldsymbol{x}_0(\boldsymbol{\mu}) \\ \mathbf{y}(t,\boldsymbol{\mu}) &= \boldsymbol{B}(\boldsymbol{\mu})^\intercal \boldsymbol{Q}(\boldsymbol{\mu})\boldsymbol{x}(t,\boldsymbol{\mu}), \end{aligned} \tag{7.2}$$

with inputs $\boldsymbol{u}(t) \in \mathbb{R}^{n_{\boldsymbol{u}}}$ and outputs $\mathbf{y}(t,\boldsymbol{\mu}) \in \mathbb{R}^{n_{\mathbf{y}}}$. Moreover, the individual matrices have the following interpretation and properties:

- **Energy Matrix**: A symmetric and positive definite matrix

$$\boldsymbol{Q}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times n}, \quad \boldsymbol{Q}(\boldsymbol{\mu}) = \boldsymbol{Q}(\boldsymbol{\mu})^\intercal \succ 0, \tag{7.3}$$

  which defines a scalar-valued function $\mathcal{H} : \mathcal{X} \times \mathcal{P} \to \mathbb{R}$, $(\boldsymbol{x},\boldsymbol{\mu}) \mapsto \frac{1}{2}\boldsymbol{x}^\intercal \boldsymbol{Q}(\boldsymbol{\mu})\boldsymbol{x}$ referred to as the *Hamiltonian* or internal energy.

- **Structure Matrix**: A skew-symmetric matrix

$$\boldsymbol{J}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times n}, \quad \boldsymbol{J}(\boldsymbol{\mu}) = -\boldsymbol{J}(\boldsymbol{\mu})^\intercal \tag{7.4}$$

  representing internal energy flows.

- **Dissipation Matrix**: A symmetric and positive semi-definite matrix

$$\boldsymbol{R}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times n}, \quad \boldsymbol{R}(\boldsymbol{\mu}) = \boldsymbol{R}(\boldsymbol{\mu})^\intercal \succeq 0 \tag{7.5}$$

  accounting for energy losses

- **Input (Port) Matrix**: A matrix $\boldsymbol{B}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times n_{\boldsymbol{u}}}$.

The rate of change of the system's stored energy can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{H}(\boldsymbol{x}(t)) = \mathbf{y}(t)^\intercal \boldsymbol{u}(t) - \underbrace{\boldsymbol{x}(t)^\intercal \boldsymbol{Q}\boldsymbol{R}\boldsymbol{Q}\boldsymbol{x}(t)}_{\geqslant 0} \leqslant \mathbf{y}(t)^\intercal \boldsymbol{u}(t) \tag{7.6}$$

by applying the chain rule along with the relation $\nabla\mathcal{H}(\tilde{\boldsymbol{x}}(t)) = \boldsymbol{Q}\tilde{\boldsymbol{x}}(t)$ and omitting the explicit dependency on the parameter vector $\boldsymbol{\mu}$ for brevity. This expression decomposes

the energy change into two distinct components, a dissipative term characterized by $\boldsymbol{R}$ and the power exchange through the port, given by $\mathbf{y}(t)^\intercal \boldsymbol{u}(t)$. Since $\boldsymbol{R}$ is positive semi-definite, the rate of energy change is bounded from above by the power introduced through the port which leads to the *dissipation inequality*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{H}(\boldsymbol{x}(t)) \leqslant \mathbf{y}(t)^\intercal \boldsymbol{u}(t) \tag{7.7}$$

that can also be expressed in integral form

$$\mathcal{H}(\boldsymbol{x}(t)) \leqslant \mathcal{H}(\boldsymbol{x}(0)) + \int_0^t \mathbf{y}(s)^\intercal \boldsymbol{u}(s)\,\mathrm{d}s. \tag{7.8}$$

This demonstrates that the total energy in the system, $\mathcal{H}(\boldsymbol{x}(t))$, is bounded by the initial stored energy, $\mathcal{H}(\boldsymbol{x}(0))$, plus the energy introduced or extracted via the system ports. In addition to the linear formulation, the nonlinear PH system provides a more general formulation of the PH framework as for example described in [RettbergEtAl24].

7.1.1 System-theoretical Properties of Port-Hamiltonian Systems

Several desirable system-theoretical properties can be derived from the dissipation inequality (7.7). In particular, it provides the basis for establishing *passivity* of the initial value problem (7.10). For a PH system with a Hamiltonian that is bounded from below, there exists a constant $\mathcal{H}_{\min} \in \mathbb{R}$ such that $\mathcal{H}(\boldsymbol{x}) \geqslant \mathcal{H}_{\min},\ \forall \boldsymbol{x} \in \mathcal{X}$. By defining a storage function as $S(\boldsymbol{x}) = \mathcal{H}(\boldsymbol{x}) - \mathcal{H}_{\min}$, it follows that $S(\boldsymbol{x}) \geqslant 0$ for all $\boldsymbol{x} \in \mathcal{X}$. Moreover, this function satisfies the dissipation inequality $\frac{\mathrm{d}}{\mathrm{d}t}S(\boldsymbol{x}(t)) \leqslant \boldsymbol{u}(t)^\intercal \mathbf{y}(t)$, where the term $\boldsymbol{u}(t)^\intercal \mathbf{y}(t)$ is typically interpreted as the supplied power. This function fulfills the necessary conditions for defining passivity (see, e.g. [SchaftJeltsema14, Sec. 7]), demonstrating that PH systems naturally exhibit energy-preserving and dissipative properties.

The storage function S of a passive system can additionally serve as a Lyapunov function to establish Lyapunov stability of its equilibrium points. For a passive system with zero inputs, i.e. $\boldsymbol{u}(t) = 0$ for all $t \in \mathcal{T}$, an equilibrium point that is also a strict local minimum of the storage function S is inherently Lyapunov stable. This follows from the fact that the dissipation inequality ensures $\frac{\mathrm{d}}{\mathrm{d}t}S(\boldsymbol{x}(t)) \leqslant 0$ along trajectories $\boldsymbol{x}(t)$ within a bounded neighborhood $\mathcal{X}_{\mathrm{N}} \subset \mathcal{X}$ containing the equilibrium point. This condition guarantees that small perturbations from the equilibrium do not lead to divergence, thereby satisfying the criteria for Lyapunov stability (see, e.g. [MeyerOffin17, Sec. 12]). Under additional assumptions, it can be demonstrated that the solution of a PH system remains within a bounded neighborhood, see e.g. [RettbergEtAl24]. Intuitively, the boundary of the neighborhood $\mathcal{X}_{\mathrm{N}}$ acts as an energy barrier. Since the input into the system is zero, the total energy cannot exceed its initial value, preventing the system state from escaping this bounded region.

7.2 Identification of Port-Hamiltonian Dynamics

Most existing system identification methods in the PH research domain either require direct access to the original governing equation matrices or operate within the frequency domain. Additionally, purely data-driven approaches are typically constrained to low-dimensional systems and do not integrate their identification framework into a Model Order Reduction (MOR) setting. In contrast, our approach identifies PH systems in a fully data-driven, non-intrusive manner using time-domain data. By leveraging standard deep learning frameworks, we facilitate efficient gradient-based optimization, benefiting from the rapid advancements in this field in recent years. This makes the methodology both scalable and easily adaptable to a wide range of applications.

Addendum 3 (System Identification of and Structure-preserving MOR for PH Systems)**.** *The discovery of PH systems from data is an active research field, with multiple recent contributions addressing different aspects of the problem. Various approaches have been proposed, including frequency-domain identification [Schwerdtner21] and time-domain methods leveraging adaptations of dynamic mode decomposition [MorandinNicodemusUnger23]. Additionally, linear PH realizations have been inferred from input-output data in [CherifiGoyalBenner22] and [OrtegaYin23] for single-input, single-output systems. In [GoyalPontesDuffBenner23], the authors infer stable quadratic models using operator inference by employing a PH-like decomposition of the system matrix.*

Machine learning techniques have also been applied to the identification of PH systems. The authors of [DesaiEtAl21] introduced PH neural networks, which parameterize the flow of low-dimensional PH systems using neural networks without explicitly learning system operators. Similarly, [NearyTopcu23] utilizes neural networks to construct low-dimensional PH systems while leveraging the network modeling properties of PH systems to decompose the problem into simpler subsystems. In [SalnikovFalaizeLozienko23], machine learning is employed to infer the connectivity structure of a system, enabling the transformation of a known unstructured ODE into a PH representation. A broader review of machine learning techniques for PH systems, with a focus on control applications, is provided in [Cherifi20]. Additionally, a pseudo-Hamiltonian neural network, which does not impose structural constraints on the forcing term and thus does not enforce passivity properties is introduced in [EidnesEtAl23]. More recently, [YıldızEtAl24] proposes an approach for identifying quadratic Hamiltonian systems using an autoencoder to either reduce high-dimensional data or lift nonlinear data while weakly enforcing a symplectic structure in the latent space.

Structure-preserving MOR techniques retain the desirable system-theoretical properties of PH systems even in reduced representations. There exist many structure-preserving variants of conventional projection-based reduction methods for PH systems. Notable examples include structure-preserving versions of moment match-

ing [PolyugavanderSchaft10, IonescuAstolfi13, EggerEtAl18], and balanced truncation [GuiverOpmeer13, BreitenMorandinSchulze22, BorjaScherpenFujimoto23], tangential interpolation [BeattieGugercinMehrmann22, GugercinEtAl12], as well as spectral factorization [BreitenUnger22]. Optimization-based structure-preserving approaches have been developed in [MoserLohmann20, SchwerdtnerVoigt23]. Moreover, [RettbergEtAl23] present a sensitivity analysis of structure-preserving projection for a dynamical system in form of a classical guitar.

For nonlinear PH systems, the research area of MOR remains active and deals with several unresolved challenges. Early results include a Principal Component Analysis (PCA)-based approach [BeattieGugercin11] and an $\mathcal{H}_2$ quasi-optimal subspace selection method [ChaturantabutBeattieGugercin16]. Generalizations of balanced truncation for nonlinear PH systems are presented in [KawanoScherpen18, SarkarScherpen23]. Furthermore, MOR has been successfully applied to nonlinear flow problems [LiljegrenSailer20, LiljegrenSailerMarheineke22] and to transport-dominated phenomena [Schulze23a]. More recent work explores structure-preserving MOR using a nonlinear separable approximation ansatz [Schulze23b]. Lastly, [LepriBacciuSantina23] describe an autoencoder-based reduction approach requiring access to Full Order Model (FOM) operators.

In this work, we introduce the Port-Hamiltonian Identification Network (PHIN), a novel framework for discovering linear PH systems from data in both nonparametric and parametric settings. However, restricting the model to a strictly linear PH formulation may be overly constraining for highly nonlinear systems, potentially leading to significant discrepancies between the identified model and the ground-truth dynamics. Additionally, when dealing with data generated by highly nonlinear systems, a linear PH description may not be well-suited to describe it. Hence, we address high-dimensional and nonlinear systems, by integrating PHIN into an autoencoder architecture, resulting in the Autoencoder-based Port-Hamiltonian Identification Network (APHIN).

Unlike in previous chapters, where the autoencoder primarily served as a dimensionality reduction tool with limited emphasis on coordinate selection, the present setting shifts the role of the autoencoder from pure dimensionality reduction to an essential component in selecting an appropriate set of state variables via nonlinear transformations (cf. Chall. 2). Approximating nonlinear systems with linear ones through a nonlinear coordinate transformation is inspired by Koopman theory, which states that any nonlinear system can be represented as a linear system in a sufficiently high-dimensional state space. While this equivalence has not yet been rigorously proven for structured systems, it is a reasonable assumption, as suggested in [YıldızEtAl24]. Additionally, wrapping an autoencoder-based framework around the linear system identification enables a transformation of the original data into a state description that aligns with the PH formulation. Despite this shift in focus, the general structure and functionality of the autoencoder remain aligned with the explanations provided in Section 3.2.8. Although the model discovery framework

is presented in the following to occur in the latent space, the methodology can also be applied directly to non-transformed data. In such cases, we define the latent states as being equivalent to the states, i.e. $\boldsymbol{z} \coloneqq \boldsymbol{x}$.

7.2.1 PHIN: Port-Hamiltonian Identification Network

To introduce our approach, the plain identification of a general linear PH system with inputs is considered first without leveraging an autoencoder. This initial step serves to establish the foundational concepts underlying our methodology, prior to its extension to scenarios coping with high dimensionality. For this, consider a set of observed parameter-dependent state observations $\{\boldsymbol{z}(t_i;\boldsymbol{\mu}_i)\}_{i=1}^{n_s} \subset \mathcal{Z}$, their time-derivatives $\{\dot{\boldsymbol{z}}(t_i;\boldsymbol{\mu}_i)\}_{i=1}^{n_s} \subset \mathcal{Z}$, with respective inputs $\{\boldsymbol{u}(t_i)\}_{i=1}^{n_s}$ and parameter vectors $\{\boldsymbol{\mu}_i\}_{i=1}^{n_s} \subset \mathcal{P}$ collected at discrete time steps $\{t_i\}_{i=1}^{n_s} \subset \mathcal{T}$. If the time derivatives $\dot{\boldsymbol{z}}(t_i)$ are not directly available from the data, they can be approximately obtained using numerical differentiation techniques. During this chapter the parameter vectors $\boldsymbol{\mu}_i$ are considered constant throughout one trajectory and represent time-invariant quantities such as material properties. We account for these dependencies by allowing the PH matrices $\tilde{\boldsymbol{J}}, \tilde{\boldsymbol{R}}, \tilde{\boldsymbol{Q}}$, and $\tilde{\boldsymbol{B}}$ to depend on the parameter vector $\boldsymbol{\mu}$.

Rather than identifying a general right-hand side $\tilde{\boldsymbol{f}}(t, \boldsymbol{z}, \boldsymbol{\mu})$, we aim to identify a LTI PH input-output system. Hence, we eliminate the explicit time dependency while introducing an additional input term $\boldsymbol{u}$. While the input could conceptually be considered part of the parameter set $\boldsymbol{\mu}$, we explicitly separate it to conform to conventional system-theoretic notation for input-affine systems. In summary, we reformulate the right-hand side as

$$\dot{\boldsymbol{z}}(t,\boldsymbol{\mu}) \approx \tilde{\boldsymbol{f}}(\boldsymbol{z}(t,\boldsymbol{\mu}),\boldsymbol{\mu},\boldsymbol{u}(t)) \coloneqq (\tilde{\boldsymbol{J}}(\boldsymbol{\mu}) - \tilde{\boldsymbol{R}}(\boldsymbol{\mu}))\tilde{\boldsymbol{Q}}(\boldsymbol{\mu})\boldsymbol{z}(t,\boldsymbol{\mu}) + \tilde{\boldsymbol{B}}(\boldsymbol{\mu})\boldsymbol{u}(t) \tag{7.9}$$

so that it adheres to the aforementioned structural PH properties. This defines an initial value problem with inputs and outputs as

$$\begin{aligned} \dot{\boldsymbol{z}}(t;\boldsymbol{\mu}) &\approx \tilde{\boldsymbol{f}}(\boldsymbol{z}(t;\boldsymbol{\mu}),\boldsymbol{\mu},\boldsymbol{u}(t)) \\ \boldsymbol{z}(0,\boldsymbol{\mu}) &= \boldsymbol{z}_0(\boldsymbol{\mu}), \\ \tilde{\mathbf{y}}(t,\boldsymbol{\mu}) &= \tilde{\boldsymbol{B}}^{\intercal}(\boldsymbol{\mu})\nabla\tilde{\mathcal{H}}(\boldsymbol{z}(t,\boldsymbol{\mu})). \end{aligned} \tag{7.10}$$

where, the dynamics are approximated as a PH system of the form (7.2) written in the latent coordinates.

To achieve this, the corresponding PH matrices $\tilde{\boldsymbol{J}}(\boldsymbol{\mu})$, $\tilde{\boldsymbol{R}}(\boldsymbol{\mu})$, $\tilde{\boldsymbol{Q}}(\boldsymbol{\mu})$, and $\tilde{\boldsymbol{B}}(\boldsymbol{\mu})$ must be parameterized using trainable weights, allowing for their optimization to best fit the given data. Moreover, we pursue an approach in which the parameterization adheres the structural properties of the PH framework, ensuring that the properties defined for the PH matrices in (7.4), (7.5), and (7.3) are strictly enforced throughout the learning process.

A common parametrization that guarantees the respective properties is

$$\begin{aligned}
\tilde{\boldsymbol{J}}(\boldsymbol{\mu}) &\coloneqq \boldsymbol{W}_{\boldsymbol{J}}(\boldsymbol{\mu}) - \boldsymbol{W}_{\boldsymbol{J}}(\boldsymbol{\mu})^{\intercal} && \implies && \tilde{\boldsymbol{J}}(\boldsymbol{\mu}) = -\tilde{\boldsymbol{J}}(\boldsymbol{\mu})^{\intercal}, \\
\tilde{\boldsymbol{R}}(\boldsymbol{\mu}) &\coloneqq \boldsymbol{W}_{\boldsymbol{R}}(\boldsymbol{\mu})\boldsymbol{W}_{\boldsymbol{R}}(\boldsymbol{\mu})^{\intercal} && \implies && \tilde{\boldsymbol{R}}(\boldsymbol{\mu}) = \tilde{\boldsymbol{R}}(\boldsymbol{\mu})^{\intercal} \geq 0, \\
\tilde{\boldsymbol{Q}}(\boldsymbol{\mu}) &\coloneqq \boldsymbol{W}_{\boldsymbol{Q}}(\boldsymbol{\mu})\boldsymbol{W}_{\boldsymbol{Q}}(\boldsymbol{\mu})^{\intercal} + \epsilon_{\text{pos,def}}\boldsymbol{I} && \implies && \tilde{\boldsymbol{Q}}(\boldsymbol{\mu}) = \tilde{\boldsymbol{Q}}^{\intercal}(\boldsymbol{\mu}) \succ 0 \\
\tilde{\boldsymbol{B}}(\boldsymbol{\mu}) &\coloneqq \boldsymbol{W}_{\boldsymbol{B}}(\boldsymbol{\mu}), &&
\end{aligned} \tag{7.11}$$

where $\boldsymbol{W}_{\boldsymbol{J}}(\boldsymbol{\mu}) \in \mathbb{R}^{r\times r}$ is a strict lower triangular matrix, $\boldsymbol{W}_{\boldsymbol{R}}(\boldsymbol{\mu}) \in \mathbb{R}^{r\times r}$ and $\boldsymbol{W}_{\boldsymbol{Q}}(\boldsymbol{\mu}) \in \mathbb{R}^{r\times r}$ are lower triangular matrices and $\boldsymbol{W}_{\boldsymbol{B}}(\boldsymbol{\mu}) \in \mathbb{R}^{r\times n_{\boldsymbol{u}}}$ is a dense matrix. By construction, the identified matrix $\tilde{\boldsymbol{J}}(\boldsymbol{\mu})$ is skew-symmetric, while the matrices $\tilde{\boldsymbol{R}}(\boldsymbol{\mu})$ and $\tilde{\boldsymbol{Q}}(\boldsymbol{\mu})$ remain symmetric and positive (semi-)definite due to the properties of the Cholesky factorization. To enforce the positive definiteness of $\tilde{\boldsymbol{Q}}(\boldsymbol{\mu})$, a regularization term $\epsilon_{\text{pos,def}}\boldsymbol{I}$ is introduced, where $\epsilon_{\text{pos,def}} = 1 \cdot 10^{-6}$ is chosen in case of single-precision computations.

Each of the matrices $\boldsymbol{W}_{\boldsymbol{J}}(\boldsymbol{\mu})$, $\boldsymbol{W}_{\boldsymbol{R}}(\boldsymbol{\mu})$, $\boldsymbol{W}_{\boldsymbol{Q}}(\boldsymbol{\mu})$, and $\boldsymbol{W}_{\boldsymbol{B}}(\boldsymbol{\mu})$ is assembled from a corresponding set of trainable independent matrix entries $\boldsymbol{w}_{\boldsymbol{J}}(\boldsymbol{\mu}) \in \mathbb{R}^{r_{\text{skew}}}$, $\boldsymbol{w}_{\boldsymbol{R}}(\boldsymbol{\mu}) \in \mathbb{R}^{r_{\text{sym}}}$, $\boldsymbol{w}_{\boldsymbol{Q}}(\boldsymbol{\mu}) \in \mathbb{R}^{r_{\text{sym}}}$, $\boldsymbol{w}_{\boldsymbol{B}}(\boldsymbol{\mu}) \in \mathbb{R}^{r\cdot n_{\boldsymbol{u}}}$. In this context, r_{skew} and r_{sym} denote the number of independent entries in a skew-symmetric and symmetric matrix of size $r \times r$, respectively, and are given by $r_{\text{skew}} = \frac{r(r-1)}{2}$, $r_{\text{sym}} = \frac{r(r+1)}{2}$. To construct the corresponding matrices from these independent matrix entries, we introduce three specialized operators that assemble dense $j \times l$ matrices, lower triangular matrices, and strict lower triangular matrices, ensuring the correct structural properties are maintained throughout the learning process. Those operators are given as

$$\mathbb{M}_{j\times l} : \mathbb{R}^{j\cdot l} \to \mathbb{R}^{j\times l},\ \boldsymbol{w} \mapsto \begin{bmatrix} [\boldsymbol{w}]_1 & \cdots & [\boldsymbol{w}]_l \\ \vdots & & \vdots \\ [\boldsymbol{w}]_{(j-1)\cdot l+1} & \cdots & [\boldsymbol{w}]_{j\cdot l} \end{bmatrix}, \tag{7.12}$$

$$\mathbb{M}_{\text{tril}} : \mathbb{R}^{r_{\text{sym}}} \to \mathbb{R}^{r\times r},\ \boldsymbol{w} \mapsto \begin{bmatrix} [\boldsymbol{w}]_1 & & & \boldsymbol{0} \\ [\boldsymbol{w}]_2 & [\boldsymbol{w}]_3 & & \\ \vdots & \vdots & \ddots & \\ \cdots & \cdots & \cdots & [\boldsymbol{w}]_{r_{\text{sym}}}, \end{bmatrix} \tag{7.13}$$

and

$$\mathbb{M}_{\text{stril}} : \mathbb{R}^{r_{\text{skew}}} \to \mathbb{R}^{r\times r},\ \boldsymbol{w} \mapsto \begin{bmatrix} 0 & & & \boldsymbol{0} \\ [\boldsymbol{w}]_1 & 0 & & \\ \vdots & \ddots & \ddots & \\ \cdots & \cdots & [\boldsymbol{w}]_{r_{\text{skew}}} & 0 \end{bmatrix}, \tag{7.14}$$

respectively. Accordingly, the weight matrices are assembled as

$$\begin{aligned}\boldsymbol{W_J} &\coloneqq \mathbb{M}_{\text{stril}}(\boldsymbol{w_J}(\boldsymbol{\mu})),\\ \boldsymbol{W_R} &\coloneqq \mathbb{M}_{\text{tril}}(\boldsymbol{w_R}(\boldsymbol{\mu})),\\ \boldsymbol{W_Q} &\coloneqq \mathbb{M}_{\text{tril}}(\boldsymbol{w_Q}(\boldsymbol{\mu})), \text{ and}\\ \boldsymbol{W_B} &\coloneqq \mathbb{M}_{r\times n_{\boldsymbol{u}}}(\boldsymbol{w_B}(\boldsymbol{\mu})).\end{aligned} \tag{7.15}$$

We choose to parameterize the independent entries of the PH matrices with an artificial neural network, which is why the proposed method is called PHIN. Specifically, we utilize a deep fully connected neural network $\boldsymbol{\Psi}_{\text{PH}}(\boldsymbol{\mu};\boldsymbol{W_\Psi}) : \boldsymbol{\mu} \mapsto \boldsymbol{w}_{\text{PH}}$, where $\boldsymbol{W_\Psi}$ represents the trainable weights of the network. This neural network maps a given parameter vector $\boldsymbol{\mu}$ to the independent matrix entries $\boldsymbol{w}_{\text{PH}} \coloneqq \left[\boldsymbol{w_J}^\intercal,\ \boldsymbol{w_R}^\intercal,\ \boldsymbol{w_Q}^\intercal,\ \boldsymbol{w_B}^\intercal\right]^\intercal$, which are subsequently used to assemble the corresponding PH matrices.

In summary, to obtain a PH matrix from a given set of parameters $\boldsymbol{\mu}$, the parameters are first processed through a fully connected neural network, which predicts the independent entries of the matrix. These entries are then assembled using one of the three introduced operators so that the final matrices can be computed according to (7.11). This computational pipeline can be expressed as

$$\begin{aligned}&\tilde{\boldsymbol{J}}(\boldsymbol{\mu}) = \boldsymbol{W_J}(\boldsymbol{\mu}) - \boldsymbol{W_J}(\boldsymbol{\mu})^\intercal = \mathbb{M}_{\text{stril}}(\boldsymbol{w_J}(\boldsymbol{\mu})) - \mathbb{M}_{\text{stril}}(\boldsymbol{w_J}(\boldsymbol{\mu}))^\intercal\\ &\tilde{\boldsymbol{R}}(\boldsymbol{\mu}) = \boldsymbol{W_R}(\boldsymbol{\mu})\boldsymbol{W_R}(\boldsymbol{\mu})^\intercal = \mathbb{M}_{\text{tril}}(\boldsymbol{w_R}(\boldsymbol{\mu}))\mathbb{M}_{\text{tril}}(\boldsymbol{w_R}(\boldsymbol{\mu}))^\intercal\\ &\tilde{\boldsymbol{Q}}(\boldsymbol{\mu}) = \boldsymbol{W_Q}(\boldsymbol{\mu})\boldsymbol{W_Q}(\boldsymbol{\mu})^\intercal + \epsilon_{\text{pos,def}}\boldsymbol{I} = \mathbb{M}_{\text{tril}}(\boldsymbol{w_Q}(\boldsymbol{\mu}))\mathbb{M}_{\text{tril}}(\boldsymbol{w_Q}(\boldsymbol{\mu}))^\intercal + \epsilon_{\text{pos,def}}\boldsymbol{I}\\ &\tilde{\boldsymbol{B}}(\boldsymbol{\mu}) = \boldsymbol{W_B}(\boldsymbol{\mu}) = \mathbb{M}_{r\times n_{\boldsymbol{u}}}(\boldsymbol{w_B}(\boldsymbol{\mu}))\\ &\text{with } [\boldsymbol{w_J}(\boldsymbol{\mu})^\intercal,\ \boldsymbol{w_R}(\boldsymbol{\mu})^\intercal,\ \boldsymbol{w_Q}(\boldsymbol{\mu})^\intercal,\ \boldsymbol{w_B}(\boldsymbol{\mu})^\intercal]^\intercal = \boldsymbol{\Psi}_{\text{PH}}(\boldsymbol{\mu};\boldsymbol{W_\Psi}).\end{aligned} \tag{7.16}$$

A visualization of this procedure is given in Fig. 7.3. The computed matrices implicitly depend on the trainable neural network weights $\boldsymbol{W_\Psi}$, ensuring their adaptability for given data while preserving the structural constraints of the PH framework.

The neural network weights $\boldsymbol{W_\Psi}$ are optimized during training by minimizing the PH loss function

$$\begin{aligned}&L_{\text{PH}}(\boldsymbol{W_\Psi})\\ &\coloneqq \frac{1}{n_s}\sum_{i=1}^{n_s}\left\|\dot{\boldsymbol{z}}_i - \left(\left(\tilde{\boldsymbol{J}}(\boldsymbol{\mu}_i;\boldsymbol{W_\Psi}) - \tilde{\boldsymbol{R}}(\boldsymbol{\mu}_i;\boldsymbol{W_\Psi})\right)\tilde{\boldsymbol{Q}}(\boldsymbol{\mu}_i;\boldsymbol{W_\Psi})\boldsymbol{z}_i + \tilde{\boldsymbol{B}}(\boldsymbol{\mu}_i;\boldsymbol{W_\Psi})\boldsymbol{u}_i\right)\right\|_2^2\end{aligned} \tag{7.17}$$

so that the identified matrices $\tilde{\boldsymbol{J}}(\boldsymbol{\mu}_i;\boldsymbol{W_\Psi})$, $\tilde{\boldsymbol{R}}(\boldsymbol{\mu}_i;\boldsymbol{W_\Psi})$, $\tilde{\boldsymbol{Q}}(\boldsymbol{\mu}_i;\boldsymbol{W_\Psi})$, and $\tilde{\boldsymbol{B}}(\boldsymbol{\mu}_i;\boldsymbol{W_\Psi})$ accurately capture the observed system dynamics. Here, the implicit dependency of the identified PH matrices on the neural network weights $\boldsymbol{W_\Psi}$ is highlighted once. Additionally, the following notational abbreviations $\dot{\boldsymbol{z}}_i \coloneqq \dot{\boldsymbol{z}}(t_i;\boldsymbol{\mu}_i)$, $\boldsymbol{z}_i \coloneqq \boldsymbol{z}(t_i;\boldsymbol{\mu}_i)$, and $\boldsymbol{u}_i \coloneqq \boldsymbol{u}(t_i)$ are introduced for better readability.

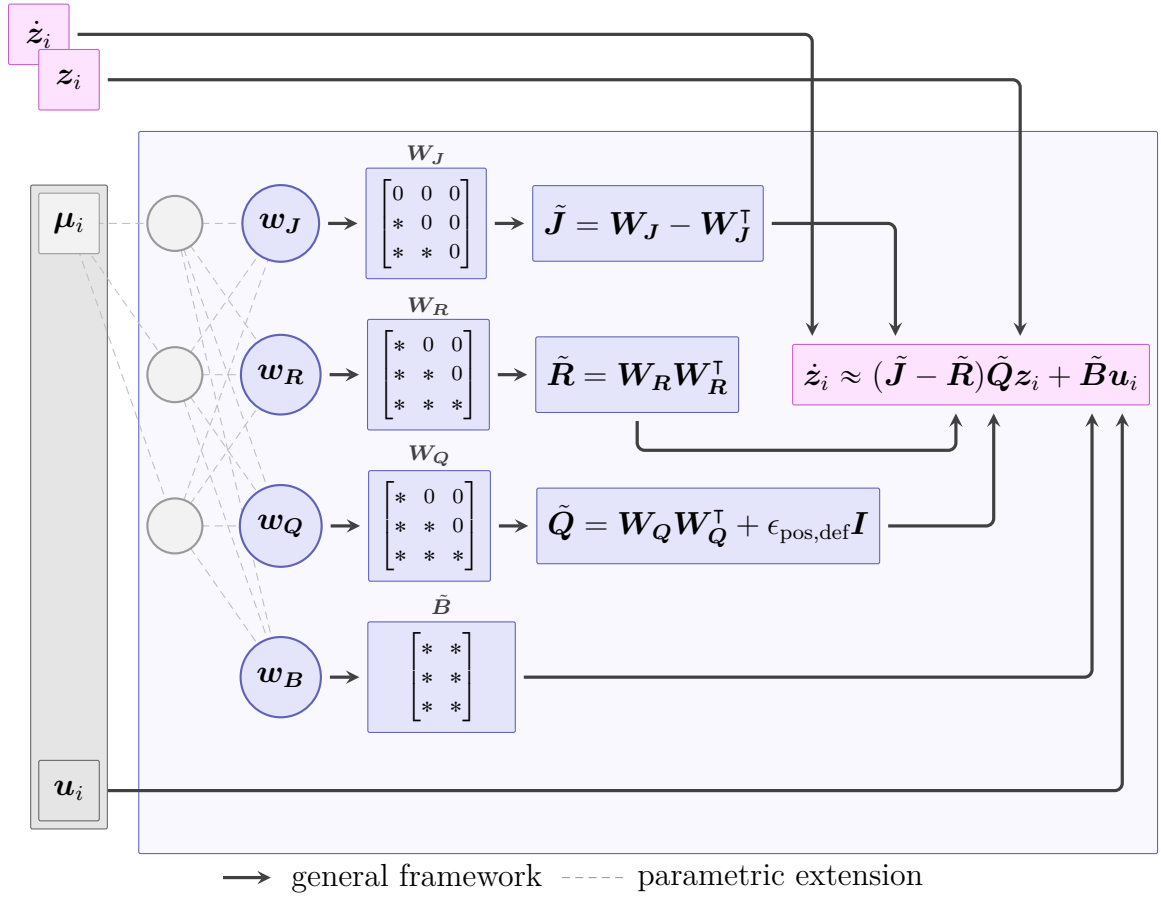


Figure 7.3: Illustration of the computational pipeline of the PH identification network (PHIN). The dependency on the time step t_i and parameter $\boldsymbol{\mu}_i$ is denoted as a subscript $\square_i$ for the states and omitted for the PH matrices for the sake of brevity.

This setup allows for the discovery of the unknown system matrices by optimizing the loss function using gradient-based optimization techniques. In a non-parametric setting, the neural network $\boldsymbol{\Psi}_{\text{PH}}$ is omitted, and the vector of independent matrix entries $\boldsymbol{w}_{\text{PH}}$ is optimized directly. For the sake of brevity, we omit the explicit dependency on the parameter vector $\boldsymbol{\mu}$ and the neural network weights $\boldsymbol{W}_{\boldsymbol{\Psi}}$ in the following discussion. This does not alter the general procedure, and these dependencies can be reintroduced as needed depending on the specific application.

7.2.2 APHIN: Autoencoder-based Port-Hamiltonian Identification Network

For high-dimensional or nonlinear systems, the next step is to seamlessly integrate PHIN into an autoencoder-based framework. Recall that an autoencoder consists of an encoder $\boldsymbol{\phi}_{\text{e}}(\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_{\text{e}}}) : \mathcal{X} \to \mathcal{Z}$, which maps the physical state variables to a reduced latent representation $\boldsymbol{z} = \boldsymbol{\phi}_{\text{e}}(\boldsymbol{x})$, and a decoder $\boldsymbol{\psi}_{\text{d}}(\boldsymbol{z}; \boldsymbol{W}_{\boldsymbol{\psi}_{\text{d}}}) : \mathcal{Z} \to \mathcal{X}$, which reconstructs the

original state from the latent space, satisfying the approximation $\boldsymbol{x} \approx \boldsymbol{\psi}_{\mathrm{d}}(\boldsymbol{z})$. This setup enables to embed PHIN within the latent space, ensuring that the trajectory of the encoded states, $\boldsymbol{z}(t) \in \mathcal{Z}$, is approximated by a trajectory $\tilde{\boldsymbol{z}}(t) \in \mathcal{Z}$ that evolves according to the identified PH dynamics. This leads to a reconstructed approximation of the original system states, given by $\tilde{\boldsymbol{x}}(t) \coloneqq \boldsymbol{\psi}_{\mathrm{d}}(\tilde{\boldsymbol{z}}(t))$, which serves as an approximation of the true state trajectory $\boldsymbol{x}(t)$. This procedure is illustrated in Fig. 7.2.

As previously mentioned, the AE itself has no notion of PH dynamics. Nevertheless, it has the task to transform the system states into a suitable set of latent variables where a PH representation is valid, while simultaneously reducing the dimensionality. To achieve this, a joint optimization of the autoencoder and PHIN is performed. This involves minimizing both the reconstruction loss (3.14) and the PH loss (7.17). By integrating these objectives into a unified learning process, we introduce a learning bias (cf. [KarniadakisEtAl21]) that encourages the AE to favor nonlinear transformations in which the latent dynamics appear both linear and consistent with PH coordinates.

Following the example of [ChampionEtAl19], we introduce an additional consistency loss

$$\begin{aligned} &L_{\mathrm{con}}(\boldsymbol{W}_{\mathrm{AE}}, \boldsymbol{W}_{\boldsymbol{\Psi}}) \\ &\qquad \coloneqq \frac{1}{n_s} \sum_{i=1}^{n_s} \left\| \dot{\boldsymbol{x}}_i - \nabla_{\boldsymbol{z}} \boldsymbol{\psi}_{\mathrm{d}}(\boldsymbol{z}_i; \boldsymbol{W}_{\mathrm{AE}}) \left((\tilde{\boldsymbol{J}} - \tilde{\boldsymbol{R}}) \tilde{\boldsymbol{Q}} \boldsymbol{z}_i + \tilde{\boldsymbol{B}} \boldsymbol{u}_i \right) \right\|_2^2, \end{aligned} \tag{7.18}$$

which aligns the reconstructed state time derivatives with the reference time derivatives $\dot{\boldsymbol{x}}$. Combining the three losses together result the overall objective to minimize

$$\begin{aligned} L(\boldsymbol{W}_{\mathrm{AE}}, \boldsymbol{W}_{\boldsymbol{\Psi}}) &\coloneqq \lambda_{\mathrm{rec}} L_{\mathrm{rec}}(\boldsymbol{W}_{\mathrm{AE}}) + \lambda_{\mathrm{PH}} L_{\mathrm{PH}}(\boldsymbol{W}_{\boldsymbol{\Psi}}) + \lambda_{\mathrm{con}} L_{\mathrm{con}}(\boldsymbol{W}_{\mathrm{AE}}, \boldsymbol{W}_{\boldsymbol{\Psi}}) \\ &= \frac{1}{n_s} \sum_{i=1}^{n_s} \Big(\underbrace{\lambda_{\mathrm{rec}} \left\| \boldsymbol{x}_i - \boldsymbol{\Psi}_{\mathrm{ae}}(\boldsymbol{x}_i; \boldsymbol{W}_{\mathrm{AE}}) \right\|_2^2}_{\text{reconstruction}} \\ &\qquad + \underbrace{\lambda_{\mathrm{PH}} \left\| \dot{\boldsymbol{z}}_i - \left((\tilde{\boldsymbol{J}} - \tilde{\boldsymbol{R}}) \tilde{\boldsymbol{Q}} \boldsymbol{z}_i + \tilde{\boldsymbol{B}} \boldsymbol{u}_i \right) \right\|_2^2}_{\text{latent dynamics}} \\ &\qquad + \underbrace{\lambda_{\mathrm{con}} \left\| \dot{\boldsymbol{x}}_i - \nabla_{\boldsymbol{z}} \boldsymbol{\psi}_{\mathrm{d}}(\boldsymbol{z}_i; \boldsymbol{W}_{\mathrm{AE}}) \left((\tilde{\boldsymbol{J}} - \tilde{\boldsymbol{R}}) \tilde{\boldsymbol{Q}} \boldsymbol{z}_i + \tilde{\boldsymbol{B}} \boldsymbol{u}_i \right) \right\|_2^2}_{\text{full dynamics}} \Big), \end{aligned} \tag{7.19}$$

where the loss factors $\lambda_{\mathrm{rec}},\ \lambda_{\mathrm{PH}},\ \lambda_{\mathrm{con}} \in \mathbb{R}^+$ are hyperparameters to adjust the weighting of the individual loss terms. The purpose of the three primary components of the loss function (7.19) are summarized as follows:

i) Reconstruction loss: Ensures that the full-state variables $\boldsymbol{x}$ can be accurately reconstructed from the latent variables $\boldsymbol{z}$ via the decoder $\boldsymbol{\psi}_{\mathrm{d}}$. This term minimizes the discrepancy between the original data and its reconstruction.

ii) Latent dynamics loss: Enforces that the identified PH system in the latent space accurately captures the observed dynamics and that the encoder $\boldsymbol{\phi}_\mathrm{e}$ transforms the system state into a suitable latent state for the PH description. Specifically, this term minimizes the deviation between the latent time derivatives $\dot{\boldsymbol{z}}$ and their predicted values from the identified PH model.

iii) Full dynamics loss: Acts as an additional regularization term to ensure consistency between the reference time derivatives of the full-state data, $\dot{\boldsymbol{x}}_i$, and the reconstruction of the approximated latent time derivatives $\nabla_{\boldsymbol{z}}\boldsymbol{\psi}_\mathrm{d}(\boldsymbol{z}_i;\boldsymbol{W}_\mathrm{AE})\left((\tilde{\boldsymbol{J}}-\tilde{\boldsymbol{R}})\tilde{\boldsymbol{Q}}\boldsymbol{z}_i+\tilde{\boldsymbol{B}}\boldsymbol{u}_i\right)$. Hence, this term ensures that the inferred latent dynamics align with the observed full-state dynamics.

In addition, the overall loss function can be augmented with L1 regularization $L_{\mathrm{L1}}(\boldsymbol{W}_{\boldsymbol{\Psi}}) = \lambda_{\mathrm{L1}} \left\|\boldsymbol{W}_{\boldsymbol{\Psi}}\right\|_1$ on $\boldsymbol{W}_{\boldsymbol{\Psi}}$ to promote sparsity in the learned matrices.

As a general guideline for tuning the loss coefficients, the reconstruction loss and latent dynamics loss, weighted by λ_rec and λ_PH, respectively, are typically the most dominant terms and should be set orders of magnitude larger than the full dynamics coefficient λ_con. The full dynamics loss becomes particularly relevant in scenarios where the learned latent variables exhibit very small magnitudes. In such cases, increasing the coefficient λ_con helps to prevent the underestimation of the latent dynamics loss so that consistency across different scales is maintained. The selection of optimal hyperparameters is inherently problem-dependent and requires careful fine-tuning for the specific dataset. Iterative tuning and cross-validation have been shown to significantly enhance training stability and convergence.

For high-dimensional state spaces where $n \gg 1$, the computation of the full dynamics loss (7.18) can be computationally expensive and memory-intensive. To mitigate this issue, a linear reduction $\boldsymbol{V}^\intercal\boldsymbol{x} \in \mathbb{R}^{n_\mathrm{PCA}}$ can be applied as a preprocessing step. This reduces the state dimension before training, allowing the encoder $\boldsymbol{\phi}_\mathrm{e}$ and decoder $\boldsymbol{\psi}_\mathrm{d}$ to operate in a space of intermediate dimension $n_\mathrm{PCA} < n$ (see, e.g., [FrescaManzoni22]). A common approach for determining the projection matrix $\boldsymbol{V} \in \mathbb{R}^{n\times n_\mathrm{PCA}}$ is PCA, as described in Section 3.2.3. However, this approach is not well-suited for problems with slowly decaying Kolmogorov n-widths [BuchfinkGlasHaasdonk23]. Consequently, the choice of the intermediate dimension n_PCA presents a trade-off between accuracy and computational efficiency.

7.2.3 Evaluation of the Discovered PH Model

After the training is completed, the discovered PH system in the latent space is expressed as LTI PH system of the form

$$\begin{aligned} \dot{\tilde{\boldsymbol{z}}}(t) &= (\tilde{\boldsymbol{J}} - \tilde{\boldsymbol{R}})\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{z}}(t) + \tilde{\boldsymbol{B}}\boldsymbol{u}(t) \,, \\ \tilde{\mathbf{y}}(t) &= \tilde{\boldsymbol{B}}^{\intercal}\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{z}}(t) \,, \\ \tilde{\boldsymbol{z}}(0) &= \tilde{\boldsymbol{z}}_0 \coloneqq \boldsymbol{z}_0 = \boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x}_0), \end{aligned} \tag{7.20}$$

where $\tilde{\boldsymbol{J}} = -\tilde{\boldsymbol{J}}^{\intercal} \in \mathbb{R}^{r\times r}$, $\tilde{\boldsymbol{R}} = \tilde{\boldsymbol{R}}^{\intercal} \geq 0 \in \mathbb{R}^{r\times r}$, $\tilde{\boldsymbol{Q}} = \tilde{\boldsymbol{Q}}^{\intercal} > 0 \in \mathbb{R}^{r\times r}$ satisfy the PH properties and $\tilde{\boldsymbol{z}}_0$ represents the encoded initial value. Structure-preserving time integration schemes, such as the implicit midpoint rule [KotyczkaLefèvre19], can be used to compute the time evolution of this system for new parameter instances and initial conditions. Such integration schemes preserve the PH properties and its structure even in discrete time. In contrast to calculations in the high-dimensional state space, the transformation into the latent state enables fast computations of the dynamics. The computed time evolution in the latent space can be transformed back into the physical space via the decoder, yielding $\tilde{\boldsymbol{x}}(t) = \boldsymbol{\psi}_{\mathrm{d}}(\tilde{\boldsymbol{z}}(t))$.

Hence, the evaluation of our model is composed out of three key steps. First, the initial condition is mapped into the latent space using the encoder. Second, the system's time evolution is computed using a structure-preserving time integration scheme, incorporating the given initial condition, inputs, and parameters. Third, the latent trajectory is mapped back into the physical space using the decoder. This workflow is visually represented in Fig. 7.4.

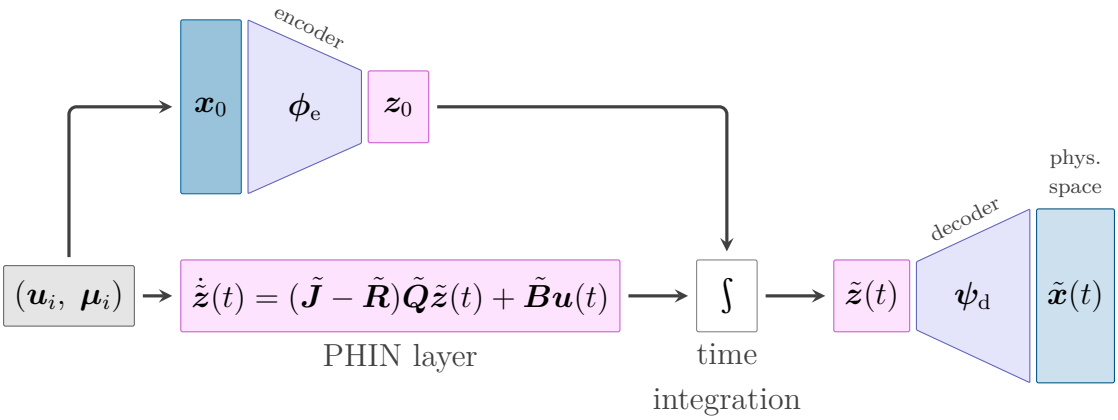


Figure 7.4: Model evaluation of the identified system: The identified (parametric) PH system is evaluated by performing time integration using the given input signals and encoded initial values. The resulting latent state trajectory is then decoded back into the high-dimensional physical space.

System-theoretic Properties of the Identified Model in the State Space It can be shown that, under mild assumptions, the system-theoretic properties and PH structure identified in the latent space $\mathcal{Z}$ are also valid in the original physical state space $\mathcal{X}$. Thus, the structural advantages inherent to $\tilde{\boldsymbol{z}}(t) \in \mathcal{Z}$ extend to the reconstructed physical states, given by $\tilde{\boldsymbol{x}}(t) = \boldsymbol{\psi}_{\mathrm{d}}(\tilde{\boldsymbol{z}}(t)) \in \mathcal{X}$. A necessary condition for transferring the PH structure from the latent space to the physical space is that the autoencoder satisfies the projection property

$$(\boldsymbol{\phi}_{\mathrm{e}} \circ \boldsymbol{\psi}_{\mathrm{d}})(\tilde{\boldsymbol{z}}) = \tilde{\boldsymbol{z}} \quad \forall \tilde{\boldsymbol{z}} \in \mathcal{Z}. \tag{7.21}$$

This condition ensures that the autoencoder is invariant under function composition, meaning that applying the encoder after the decoder does not alter the state.

If this assumption holds, and if it also remains valid after derivation with respect to the latent state, the PH system identified by APHIN in the latent space $\mathcal{Z}$ defines a nonlinear PH system on a submanifold of the physical state space, so that

i) The dissipation inequality (7.7) holds on the submanifold of the physical state space,

ii) The reconstructed Hamiltonian $\tilde{\mathcal{H}} = \mathcal{H} \circ \boldsymbol{\phi}_{\mathrm{e}}$ remains bounded from below, ensuring that the system retains passivity,

iii) A decoded Lyapunov stable equilibrium point from the latent space $\tilde{\boldsymbol{x}}_{\mathrm{e}} \coloneqq \boldsymbol{\psi}_{\mathrm{d}}(\tilde{\boldsymbol{z}}_{\mathrm{e}}) \in \tilde{\mathcal{X}}$, is also Lyapunov stable,

iv) The solutions in the submanifold of the state space remain bounded if the solutions in the latent space are bounded and the decoder adheres to Lipschitz continuity.

For detailed proofs of these properties, refer to [RettbergEtAl24], where we established these statements rigorously.

In practice, the assumption (7.21) is generally not exactly satisfied. One approach to enforce this property is discussed in [OttoMacchioRowley23], where autoencoder architectures are explicitly constrained to ensure the projection property. Alternatively, [BuchfinkEtAl24, Thm. 6.4] argues that this property is approximately satisfied for a generic autoencoder, provided that the autoencoder is Lipschitz continuous and the reconstruction loss (3.14) is sufficiently small. In the following numerical experiments, we adopt the second approach and do not explicitly enforce the projection property (7.21).

7.3 Results

In the following, we demonstrate the effectiveness of our proposed PHIN and APHIN approaches through a series of numerical experiments. We begin with a mechanical PH

benchmark system, specifically a MSD chain, which was introduced at the beginning of this chapter. This serves as a validation case to assess the performance of PHIN in isolation in an input-affine and parametric setting. Next, APHIN is evaluated on a nonlinear low-dimensional system, highlighting its capability to learn structured latent-space representations. Finally, we extend the analysis to a nonlinear high-dimensional system with input-affine and parametric dependencies, demonstrating the scalability and generalization of APHIN in more complex dynamical settings. The hyperparameters used to create the data for the individual numerical examples, as well as the settings for the identification of corresponding latent PH systems, can be found in Table 7.1.

7.3.1 Numerical example I: Mass-spring-damper chain

The first numerical example represents an one-dimensional MSD chain connected by links– is a typical benchmark system in the PH research field, as visualized in Fig. 7.1a. A system with three links can be formulated as a PH system (7.2) using the displacements $\boldsymbol{d} = [d_\mathrm{I}, d_\mathrm{II}, d_\mathrm{III}]^\intercal \in \mathbb{R}^3$ and momenta $\boldsymbol{\varrho} = [\varrho_\mathrm{I}, \varrho_\mathrm{II}, \varrho_\mathrm{III}]^\intercal \in \mathbb{R}^3$ as states $\tilde{\boldsymbol{z}} = [\boldsymbol{d}^\intercal, \boldsymbol{\varrho}^\intercal]^\intercal$. Accordingly, this example represents a low-dimensional system that can be used to test the capability of the PHIN method alone. This setup allows for a direct comparison between the identified system matrices and their corresponding reference matrices.

For the chain with three links and input applied only to the leftmost mass and with masses $\{m_\mathrm{I}, m_\mathrm{II}, m_\mathrm{III}\}$, stiffness values $\{k_\mathrm{I}, k_\mathrm{II}, k_\mathrm{III}\}$, and damping values $\{c_\mathrm{I}, c_\mathrm{II}, c_\mathrm{III}\}$, the according system matrices read

$$\hat{\boldsymbol{J}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}, \qquad \hat{\boldsymbol{R}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_\mathrm{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_\mathrm{II} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_\mathrm{III} \end{bmatrix}, \tag{7.22}$$

$$\hat{\boldsymbol{Q}} = \begin{bmatrix} k_\mathrm{I} & -k_\mathrm{I} & 0 & 0 & 0 & 0 \\ -k_\mathrm{I} & k_\mathrm{I} + k_\mathrm{II} & -k_\mathrm{II} & 0 & 0 & 0 \\ 0 & -k_\mathrm{II} & k_\mathrm{II} + k_\mathrm{III} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{m_\mathrm{I}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{m_\mathrm{II}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{m_\mathrm{III}} \end{bmatrix}, \qquad \hat{\boldsymbol{B}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \tag{7.23}$$

Data Generation and Model Settings In a first step, parametric training data are generated by varying the parameter vector $\boldsymbol{\mu} = (m, k, c)$, where, the mass parameters and stiffness parameters are set to $m_\mathrm{I} = m_\mathrm{II} = m_\mathrm{III} = m \in [0.1, 100]\,\mathrm{kg}$ and $k_\mathrm{I} = k_\mathrm{II} = k_\mathrm{III} = k \in [0.1, 100]\,\frac{\mathrm{N}}{\mathrm{m}}$, while the damping parameters are set to zero, $c_\mathrm{I}, c_\mathrm{II}, c_\mathrm{III} = 0\,\frac{\mathrm{Ns}}{\mathrm{m}}$, to obtain a

Table 7.1: Hyperparameters for the different numerical examples.

Hyperparam. Category	**MSD chain**	**Pendulum**	**Discbrake**
AE	-	Number of layers: 3, Neurons per layer: $32 \times 32 \times 32$, Activation: elu	Number of layers: 3, Neurons per layer: $64 \times 32 \times 16$, Activation: elu
Parametric PHIN	Number of layers: 3, Neurons per layer: $16 \times 16 \times 16$, Activation: elu	-	Number of layers: 3, Neurons per layer: $32 \times 32 \times 32$, Activation: elu
Dimensions	Full: $n = 6$ PCA: - Latent: -	Full: $n = 4$ PCA: $n_{\text{PCA}} = -$ Latent $r = 2$	Full: $n = 998$ PCA: $n_{\text{PCA}} = 8$ Latent $r = 3$
Loss Factors	$\lambda_{\text{rec}} = -$ $\lambda_{\text{PH}} = 1$ $\lambda_{\text{con}} = -$ $\lambda_{\text{L1}} = 1 \cdot 10^{-07}$	$\lambda_{\text{rec}} = 1$ $\lambda_{\text{PH}} = 0.1$ $\lambda_{\text{con}} = 0.001$ $\lambda_{\text{L1}} = 1 \cdot 10^{-10}$	$\lambda_{\text{rec}} = 1$ $\lambda_{\text{PH}} = 0.1$ $\lambda_{\text{con}} = 0.001$ $\lambda_{\text{L1}} = 1 \cdot 10^{-09}$
Training Regularization	Select best weights w.r.t. val. data, early stopping	Select best weights w.r.t. val. data, early stopping	Select best weights w.r.t. val. data, early stopping
Optimization	Adam optimizer, Learn. rate: 10^{-3}, Batch size: 64, Epochs: 2000, Loss: MSE	Adam optimizer, Learning rate: 10^{-3}, Batch size: 256 Epochs: 4000, Loss: MSE	Adam optimizer, Learning rate: 10^{-3}, Batch size: 256 Epochs: 2000, Loss: MSE
Data	$n_{\text{sim}}^{\text{train}} = 90$, $n_{\text{sim}}^{\text{test}} = 30$, $n_t{}^{\text{train}} = 2000$, $n_t{}^{\text{test}} = 1600$, $t_{\text{end}}^{\text{train}} = 20\,\text{s}$, $t_{\text{end}}^{\text{test}} = 16\,\text{s}$	$n_{\text{sim}}^{\text{train}} = 12$, $n_{\text{sim}}^{\text{test}} = 6$, $n_t{}^{\text{train}} = 500$, $n_t{}^{\text{test}} = 1000$, $t_{\text{end}}^{\text{train}} = 5\,\text{s}$, $t_{\text{end}}^{\text{test}} = 10\,\text{s}$	$n_{\text{sim}}^{\text{train}} = 48$, $n_{\text{sim}}^{\text{test}} = 20$, $n_t{}^{\text{train}} = 3001$, $n_t{}^{\text{test}} = 3001$, $t_{\text{end}}^{\text{train}} = 3\,s$, $t_{\text{end}}^{\text{test}} = 3\,s$

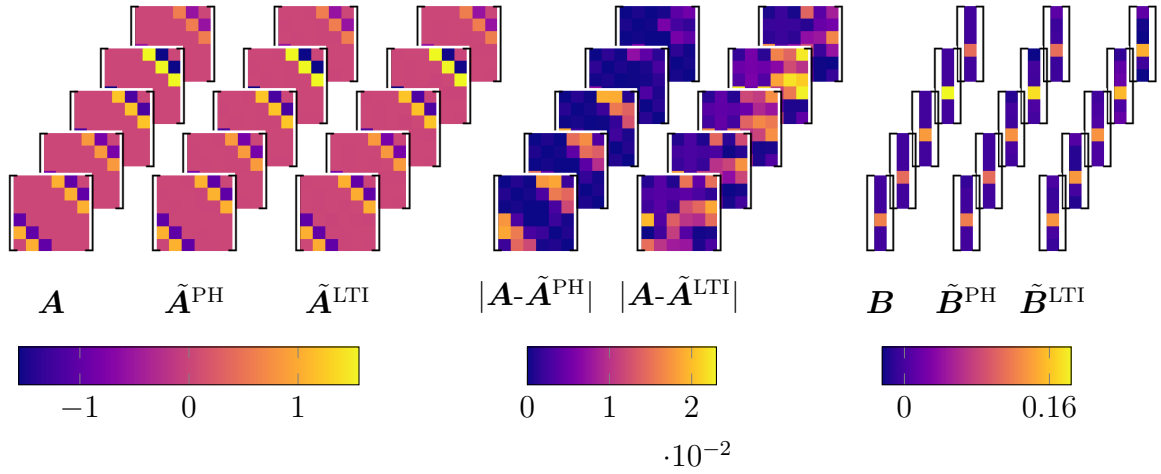
system that is marginally stable. A damped harmonic input signal $u(t) = e^{(-\mu_{\text{I}}^u \cdot t)} \sin(\mu_{\text{II}}^u t^2)$ serves as input to the signal. It depends on the decay rate $\mu_{\text{I}}^u \in [0.125, 2]$ and the frequency $\mu_{\text{II}}^u \in [0.5, 5]\,\text{Hz}$, which are sampled using quasi-random algorithms. Moreover,

each simulation starts from a different initial condition that is sampled in a similar manner.

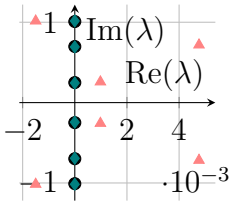
Based on the obtained data, the parametric PH matrices are identified using PHIN, following the methodology detailed in Section 7.2.1. To further elucidate the significance of structure-preserving versus unstructured system identification—briefly introduced at the beginning of this chapter—we additionally derive a second model in the form of an LTI system following the same procedure as for APHIN without enforcing any structural constraints. The identified matrices are subsequently compared with the reference matrices. However, the PH representation is inherently non-unique. Consequently, rather than directly comparing individual PH matrices, we evaluate the reference state matrix $\boldsymbol{A}$ and input matrix $\boldsymbol{B}$ against the identified matrices. In case of PHIN, the approximated state matrix corresponds to the product of the identified matrices, given by $\tilde{\boldsymbol{A}}^{\mathrm{PH}} = (\tilde{\boldsymbol{J}} - \tilde{\boldsymbol{R}})\tilde{\boldsymbol{Q}}$, while $\tilde{\boldsymbol{B}}^{\mathrm{PH}}$ represents the identified input matrix. For the unstructured model, the resulting system matrix $\tilde{\boldsymbol{A}}^{\mathrm{LTI}}$ is a real, dense matrix without any enforced structural properties. Both identified systems are evolved in time using the implicit midpoint rule.

Results As illustrated in Fig. 7.5a, the identified matrices exhibit a strong agreement with their reference counterparts across various parameter samples for both the unstructured LTI system and the structured PH system. However, an analysis of the error in the state and input matrices reveals that the unstructured approach results in significantly higher errors, whereas the sparsity patterns of the matrices are accurately preserved only by the PH method. Furthermore, the identified PH model correctly determines that the input is applied exclusively to the first mass, as indicated by the identification of near-zero values for all remaining entries in $\tilde{\boldsymbol{B}}$.

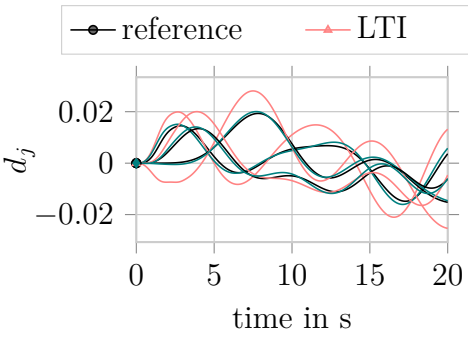
Examining the eigenvalues, visualized in Fig. 7.5b, we observe that PHIN correctly identifies all eigenvalues near the imaginary axis, whereas the unstructured identified LTI system exhibits eigenvalues in the positive real half-plane, confirming its instability as previously discussed. This demonstrates that the PH representation faithfully preserves the system's inherent properties, which is further reflected in the time evolution of various test simulations. As an illustrative example, Fig. 7.5c presents a test trajectory of the displacements of the three masses. The PHIN predictions closely align with the reference solution, whereas the unstructured LTI system exhibits significant deviations. A more quantitative assessment is provided in Fig. 7.5d, where the mean state error across all test trajectories confirms that the PHIN model significantly outperforms the unstructured LTI system, ensuring both accuracy and physical consistency in the learned dynamics. The overall relative errors are listed in Table 7.2.



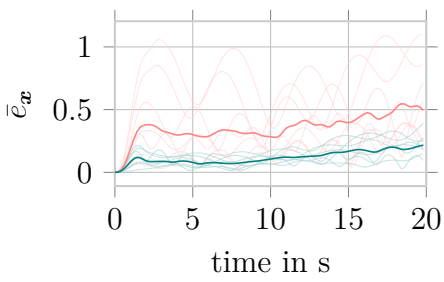
(a) Reference matrices and the corresponding identified PH matrices, $\tilde{\boldsymbol{A}}^{\mathrm{PH}}$ and $\tilde{\boldsymbol{B}}^{\mathrm{PH}}$, and identified LTI matrices, $\tilde{\boldsymbol{A}}^{\mathrm{LTI}}$ and $\tilde{\boldsymbol{B}}^{\mathrm{LTI}}$, for five example parameter vectors $\boldsymbol{\mu}$. The middle block shows the element-wise absolute error for the system matrices.



(b) Eigenvalues $\boldsymbol{\lambda}$ for an example test configuration.



(c) Displacements $\boldsymbol{d}$ of the three masses for an example test configuration.



(d) Mean of state error $\bar{e}_{\boldsymbol{x}}$ over all test simulations. Individual trajectories are drawn opaquely.

Figure 7.5: Results for the MSD chain. The corresponding identified system matrices can be seen in (a). The eigenvalues of the reference and the identified unstructured (LTI) and structured (PHIN) systems, shown in (b), provide information about the identified system behavior. In (c), the displacements of the three masses is visualized, while (d) shows the error between the identified and reference simulations over time.

7.3.2 Numerical example II: Pendulum

In the second example, we demonstrate the advantages and necessity of integrating the identification of a linear PH system within a nonlinear coordinate transformation. To this end, we consider a mathematical pendulum, a well-established nonlinear mechanical system whose dynamics can be explicitly derived using the Hamiltonian formalism, see e.g. [CastañosEtAl13]. A schematic representation of the pendulum is shown in Fig. 7.6a, illustrating its configuration. The system consists of a massless rod of length l, to which

end point a mass is attached. The Cartesian coordinates of the rod's endpoint are given by $(\mathsf{p}_x, \mathsf{p}_y)$. Additionally, the the small-angle approximation is not considered since the maximum deflection of the pendulum is chosen to be $|\omega| \leqslant \frac{\pi}{3}$.

The dynamics of the pendulum in Cartesian coordinates can be formulated as a Differential-Algebraic Equation (DAE) system with the state vector

$$\boldsymbol{x} = \begin{bmatrix} \mathsf{p}_x & \mathsf{p}_y & \dot{\mathsf{p}}_x & \dot{\mathsf{p}}_y \end{bmatrix}^\mathsf{T} , \tag{7.24}$$

which consists of the endpoint positions and their time derivatives. The system is subject to the geometric constraint

$$\mathsf{p}_x^2 + \mathsf{p}_y^2 - \mathsf{l}^2 = 0 , \tag{7.25}$$

which ensures that the motion is restricted to the feasible states. For many mechanical systems, it is more convenient to express the dynamics in generalized coordinates rather than Cartesian coordinates. A natural choice for the pendulum is the angular displacement ω and the angular velocity $\dot{\omega}$. The corresponding second-order nonlinear ODE governing the system dynamics is given by

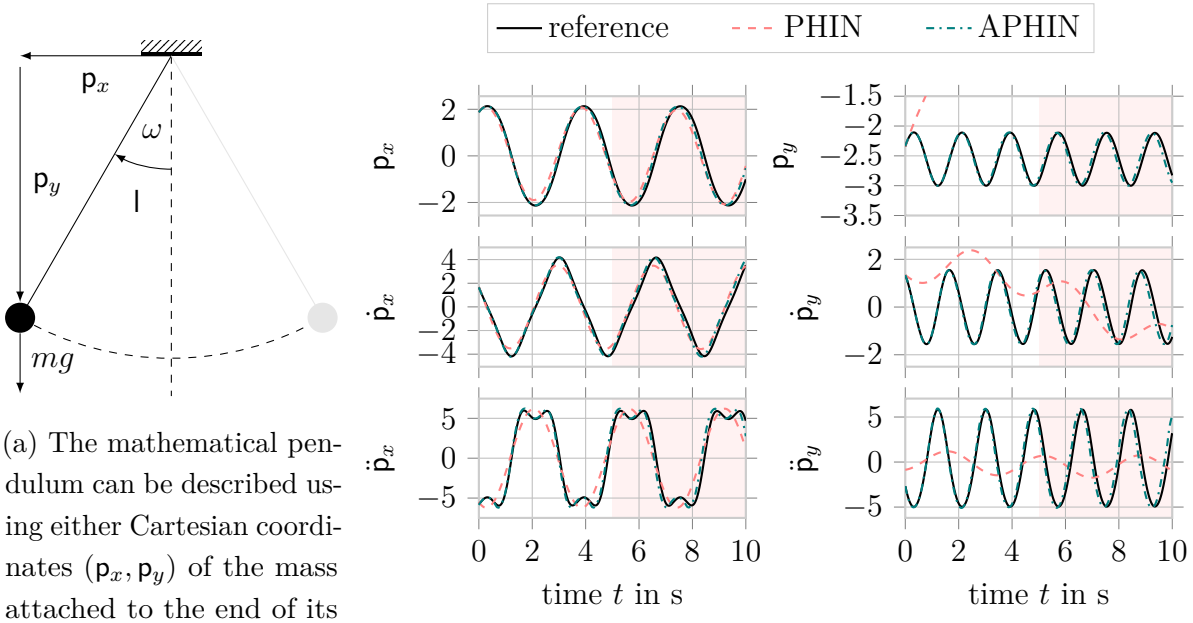
$$\ddot{\omega} = -\frac{g}{\mathsf{l}} \sin \omega. \tag{7.26}$$

The Cartesian coordinates can then be expressed in terms of the generalized coordinates as

$$\boldsymbol{x} = \begin{bmatrix} \mathsf{l} \sin \omega & \mathsf{l} \cos \omega & \mathsf{l}\dot{\omega} \cos \omega & \mathsf{l}\dot{\omega} \sin \omega \end{bmatrix}^\mathsf{T} . \tag{7.27}$$

Data Generation and Model Settings To identify the system dynamics, state data is generated in Cartesian coordinates based on varying initial conditions. The simulations used for training cover a range of 5 s, while the test trajectories capture 10 s to investigate time extrapolation. Further details on the dataset and the implemented model are provided in Table 7.1. To highlight the necessity of nonlinear transformations in the proposed framework, we compare two distinct setups: Plain identification using PHIN and identification embedded in the transformation using APHIN. In the first setup, the system state can only be linearly transformed via $\boldsymbol{Q}$ and the state dimension remains $n = 4$. In the second setup, the system states are nonlinear transformed via the autoencoder and the dimension is reduced to $r = 2$, aligning with the state dimension of the pendulum in generalized coordinates. Each identified system is then solved for unseen initial conditions.

Results The solutions obtained for a test trajectory are depicted in Fig. 7.6b. As expected, just applying PHIN to the state data in an attempt to model the pendulum as a linear PH system results in significant errors. This is caused by inherent nonlinearities present in the pendulum's dynamics. As shown in the results, PHIN alone produces an identified system that does not accurately capture the system's behavior but only the variable p_x, while the second state p_y remains inadequately represented.



(a) The mathematical pendulum can be described using either Cartesian coordinates $(\mathsf{p}_x, \mathsf{p}_y)$ of the mass attached to the end of its rod or as generalized coordinate in form of the angle ω. The system is subject to a gravitational force given by mg, where l is the rod length, g is the gravitational acceleration, and m is the mass at the end of the rod.

(b) Time Evolution of the Pendulum's Cartesian coordinates, velocities, and accelerations for both the reference model and the identified models. The identified systems are based on different coordinate transformations applied to the data. Only the first five seconds of the trajectory were used for training the extrapolation region beyond the training data is highlighted with a red background. Important Note: The velocities $\dot{\mathsf{p}}_x$ and $\dot{\mathsf{p}}_y$ are not computed via numerical differentiation. Instead, they were directly reconstructed from the latent space, as they are explicitly included in the learned state representation.

Figure 7.6: Sketch of the mathematical pendulum (a) along with the results of the PH identification (b).

In contrast, APHIN effectively circumvents this limitation by identifying a set of coordinates in which the pendulum dynamics can be described as a linear PH system. Indeed, employing a nonlinear autoencoder to discover a coordinate transformation where the dynamics exhibit approximate linearity, proves to be successful even though it is accompanied by a dimensionality reduction to $r = 2$. As illustrated in Fig. 7.6b, the temporal evolution of all states and their derivatives closely follows the reference solution, demonstrating not only accurate reconstruction but also the capacity for time extrapolation. These findings are further supported by the performance metrics presented in Table 7.2, which indicate that nonlinear dimensionality reduction yields superior errors in both latent and state space coordinates.

Table 7.2: Performance measures.

	$E_{\boldsymbol{z}}$	$E_{\boldsymbol{x}}$
mass-spring-damper		
train	–	0.08
test	–	0.07
pendulum		
PHIN		
train	0.68	1.32
test	0.70	1.13
APHIN		
train	0.06	0.05
test	0.16	0.15
disc brake		
APHIN		
train	$2.91 \cdot 10^{-4}$	$8.47 \cdot 10^{-3}$
test	$5.44 \cdot 10^{-4}$	$9.04 \cdot 10^{-3}$

7.3.3 Numerical example III: Disc Brake–A Coupled Thermo-Mechanical System

In the final numerical example, the limits of the APHIN framework are assessed by evaluating its suitability for a high-dimensional, parameter-dependent, and input-affine system. This scenario imposes additional challenges in form of a substantial dimensionality reduction with a simultaneous identification of a non-autonomous system with parametric dependencies on material properties. Specifically, we aim to derive a low-dimensional PH model for a disc brake system representing an application of significant practical relevance to vehicle safety. A reliable simulation model of the disc brake enables the investigation of dynamic phenomena that contribute to undesirable brake squeal, which arises due to deformations induced by frictional heat generated during braking.

The HF model is realized using the Finite Element (FE) method in the commercial simulation software *Abaqus*, which employs fully-coupled linear thermoelasticity equations. The model consists of a single layer comprising 60 elements, leading to a total of $n_{\text{node}} = 146$ nodes. The discretized representation of the disc brake model is illustrated in Fig. 7.7, where it is highlighted that the structure is clamped at four nodes. This constraint results in a total of $n = 998$ Degree of Freedoms (DOFs). Each unconstrained node possesses seven DOFs, comprising a single temperature-related DOF, along with three displacement and three velocity DOFs, respectively. Consequently, the state vector for the disc brake,

encompassing all nodes, is defined as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\varpi}^\mathsf{T} & \boldsymbol{d}^\mathsf{T} & \dot{\boldsymbol{d}}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{998}, \tag{7.28}$$

where $\boldsymbol{\varpi} \in \mathbb{R}^{146}$ is the temperature, $\boldsymbol{d} \in \mathbb{R}^{426}$ the displacements, and $\dot{\boldsymbol{d}} \in \mathbb{R}^{426}$ the velocities.

This model exemplifies a common scenario in which access to the underlying operators is restricted due to the closed-source nature of the software, thereby precluding the use of intrusive model reduction techniques (cf. Chall. 4). Simultaneously, the computational demands associated with the FE method can be substantially alleviated through our proposed approach. Consequently, the presented framework provides an effective solution for deriving a computationally efficient ROM solely from data.

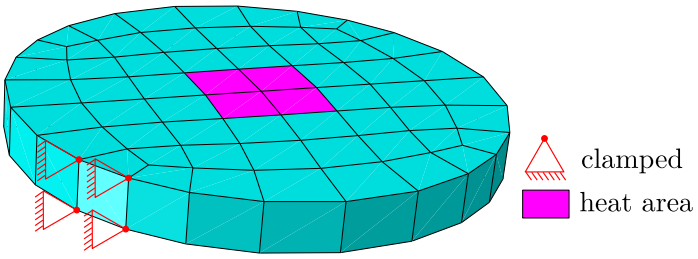


Figure 7.7: Discretized FE model of the disc brake. The areas where heat is applied are colored in pink. The motion of the model is constrained at four nodes on the left, marked in red.

Data Generation and Model Settings A parameterized scenario in which heat is introduced into the disc brake is considered to assess the performance of our proposed method. The heat enters at a designated input area comprising nine nodes (highlighted in pink in Fig. 7.7). The applied heat flux, $\mu^u \in [0.01, 9.99] \cdot 10^6\,\mathrm{W/m^2}$, is imposed as a constant input $u(t) = \mu^u$ on these nodes. Two key material properties, the disc's thermal conductivity, $[\boldsymbol{\mu}]_1 \in [6.5, 86.5]\,\mathrm{W/mK}$, and its density, $[\boldsymbol{\mu}]_2 \in [6850, 8850]\,\mathrm{kg/m^3}$, shape the behavior of the disc brake and form the the parameter vector, $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^2$. Each full-order simulation starts from an identical initial condition, $\boldsymbol{x}_0 = \boldsymbol{0}$, while the parameter vector is varied using a quasi-random sampling approach based on Halton sequences.

The exported simulation results from Abaqus include only the displacements, $\boldsymbol{d}$, and the temperature, $\boldsymbol{\varpi}$. To accommodate the second-order structure inherent in mechanical systems, the dataset is further augmented with velocity components, $\dot{\boldsymbol{d}}$, which are computed using a second-order central difference scheme applied to the displacement data, $\boldsymbol{d}$. Hence the total dataset includes

$$\{\boldsymbol{\varpi}(t_j;\boldsymbol{\mu}_j)\}_{j=1}^{n_s}, \quad \{\boldsymbol{d}(t_j;\boldsymbol{\mu}_j)\}_{j=1}^{n_s}, \quad \{\dot{\boldsymbol{d}}(t_j;\boldsymbol{\mu}_j)\}_{j=1}^{n_s}, \quad \{u[t_j]\}_{j=1}^{n_s}, \text{ and } \quad \{\boldsymbol{\mu}_j\}_{j=1}^{n_s}.$$

Due to the multi-physical nature of the problem, the states in the different domains live on different scales. Hence, each domain is scaled individually to ensure numerical values

remain within a comparable order of magnitude. Additionally, all training inputs and parameters are normalized component-wise to the interval $[0, 1]$ based on the minimum and maximum values observed across all training simulations. The same scaling factors derived from the training data are subsequently applied to the test data. To further enhance computational efficiency during model training, the dataset undergoes dimensionality reduction to a intermediate state dimension through a preprocessing step using PCA computed on the training data, which facilitates the learning process.

Results To account for the multi-physical nature of the problem, the normalized state error (5.3) is decomposed across the different physical domains. Accordingly, Fig. 7.8 presents the mean error in state space over all test trajectories, considering only the respective contributions from temperature, displacements, and velocities. The results demonstrate that the proposed framework effectively captures the heating and expansion behavior of the disc brake. The mean test error remains low for the velocity component, $\dot{\boldsymbol{d}}$, and is even lower for both the temperature, $\boldsymbol{\varpi}$, and displacement, $\boldsymbol{d}$, components. Furthermore, the results summarized in Table 7.2 demonstrate an overall satisfying approximation performance.

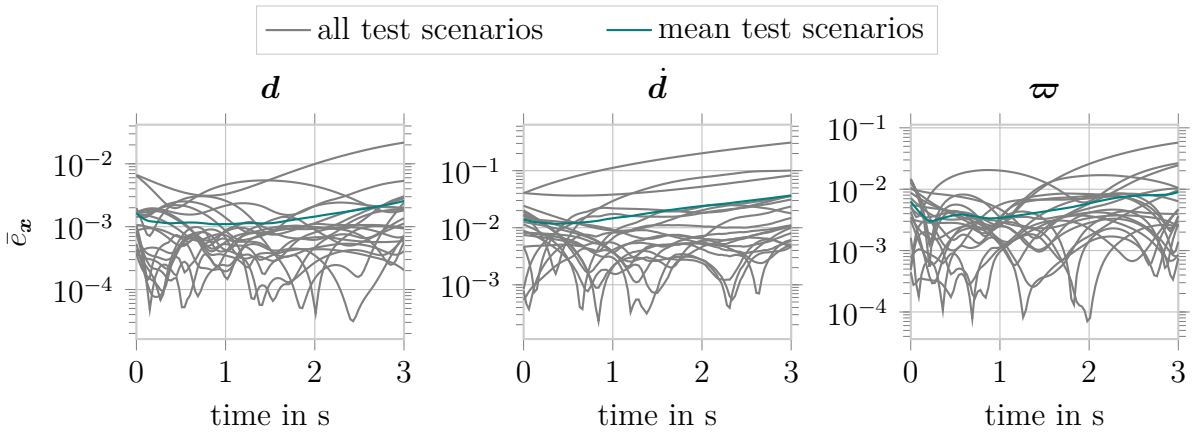


Figure 7.8: Normalized state error $\bar{e}_{\boldsymbol{x}}$ (5.3) divided into the different physical domains for test data.

Beyond the globally defined error metrics, we also aim to provide local errors at the individual nodes of the disc brake in a visually tangible way. To this end, Fig. 7.9 presents a comparative visualization of the 3D model, showcasing example test simulations obtained from the FE software alongside the corresponding approximations generated using our APHIN framework. This visualization illustrates the simulated deformation and temperature distribution while also depicting the absolute error at each node across different domains. As observed, the largest discrepancies in temperature and displacement primarily occur in the vicinity of the heat input area, whereas the highest velocity errors are concentrated on the opposite side of the clamping region. Despite these

localized deviations, the framework accurately captures the overall behavior of the disc brake across a range of parameter values and input variations. From a computational perspective, the entire evaluation process—including encoding the initial condition, solving the identified PH system from $\boldsymbol{z}_0$, and projecting the resulting trajectory back into the state space—requires only approximately 0.1 s, underscoring the computational efficiency of the proposed approach.

7.4 Discussion

The proposed framework introduces a structure-preserving system identification approach within the PH formulation, enabling the identification of low-dimensional linear PH systems solely from data, even in parametric and input-affine scenarios and for potentially high-dimensional systems. The identified PH system is parameterized by a neural network, referred to as Port-Hamiltonian Identification Network (PHIN), which ensures that the learned model retains key system-theoretic properties, including passivity and stability, within the latent space. This formulation makes the identified model particularly well-suited for multi-physics problems. For a low-dimensional input-affine and parametric mechanical system, exemplified by a mass-spring-damper chain, the advantages of this structure-preserving approach over an unstructured identification approach are demonstrated. Specifically, the method accurately reconstructs the system matrices from data while maintaining the desired system-theoretic properties.

For the identification of high-dimensional or inherently nonlinear systems, the PHIN architecture is embedded within a nonlinear coordinate transformation using an autoencoder, resulting in APHIN. This autoencoder maps the original system states onto a lower-dimensional latent space where their dynamics can be effectively described as a linear PH system. Crucially, it can be shown that when the autoencoder satisfies the projection property, the PH structure is preserved not only in the latent space but also in the original state space. The advantages of this approach are demonstrated using a nonlinear mathematical pendulum, where the autoencoder's capability to perform nonlinear transformations of the system state proved crucial for enabling successful system identification as a linear PH system.

Additionally, the framework is applied to a thermomechanical FE model of a disc brake, demonstrating its effectiveness in modeling high-dimensional, parametric, and input-affine dynamic behavior. For this example, a key advantage of the framework is its capability to produce real-time capable ROMs. This efficiency stems from the lightweight structure of the autoencoder, which relies solely on rapidly evaluable artificial neural networks, as well as the low-dimensional nature of the PH system in the latent space, which can be efficiently integrated using structure-preserving integration schemes. Overall, these results demonstrate that the proposed framework successfully reduces system dimensionality to

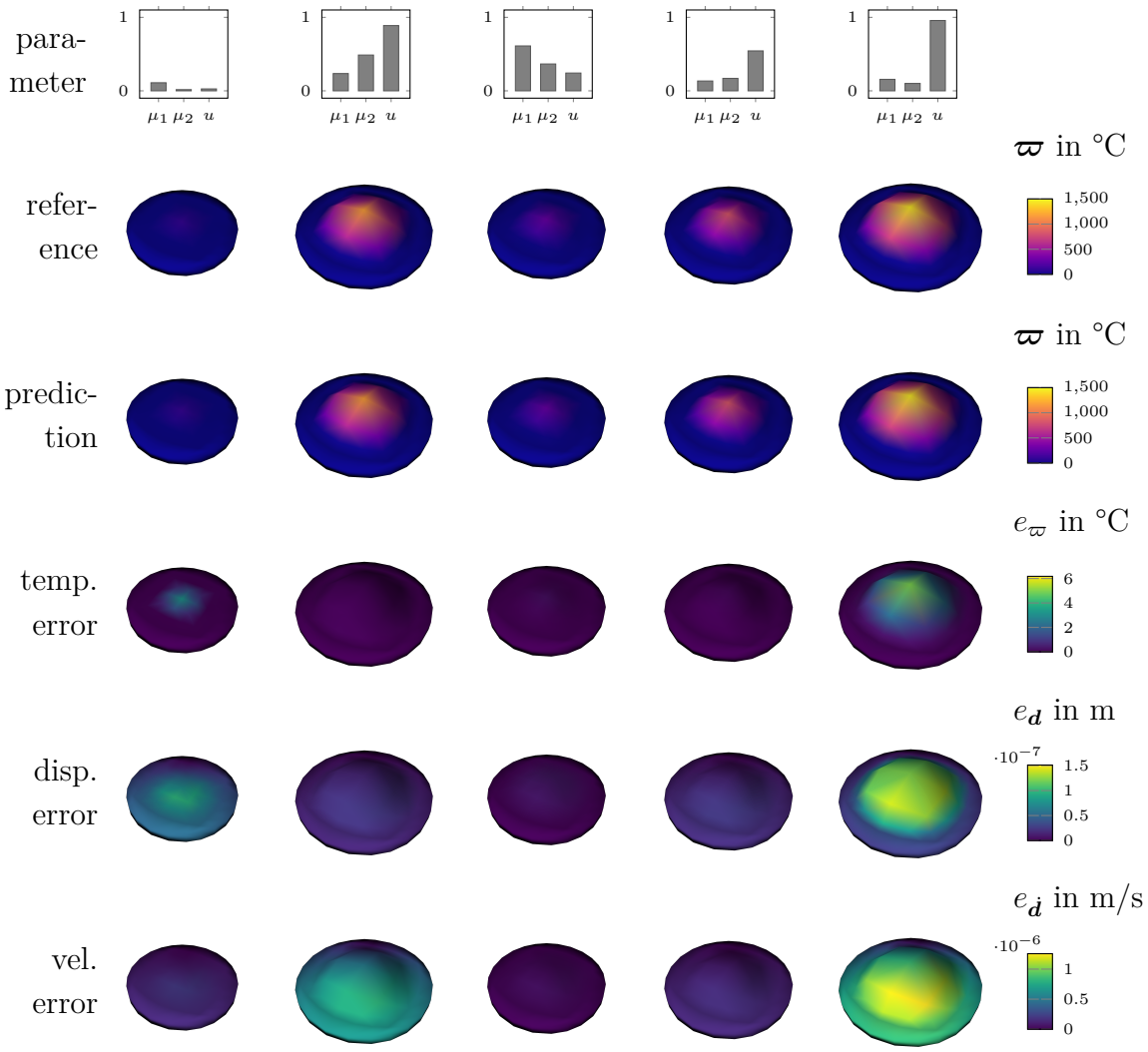


Figure 7.9: Visualization of the displacement and temperature distributions of the disc brake for five representative test simulations, comparing the reference solutions (first row) with their approximations obtained using our APHIN framework (second row). The corresponding simulation parameters $\boldsymbol{\mu}$ and input u are indicated above each disc brake visualization. Beyond displaying the simulation results, the approximation quality for temperature, displacement, and velocity fields is illustrated in rows three to five, respectively, using color-coded discs that represent the local error at each node. For enhanced visual clarity, the displacements are scaled by a factor of 400, and the last time step of each simulation is depicted. The used node-wise errors are $e_\varpi \coloneqq \left\| [\varpi]_l - [\tilde{\varpi}]_l \right\|$, $e_{\boldsymbol{d}} \coloneqq \left\| [\boldsymbol{d}]_l - \left[\tilde{\boldsymbol{d}}\right]_l \right\|_2$, $e_{\dot{\boldsymbol{d}}} \coloneqq \left\| \left[\dot{\boldsymbol{d}}\right]_l - \left[\dot{\tilde{\boldsymbol{d}}}\right]_l \right\|_2$, where l indicates that the corresponding quantity belongs to the l-th node of the model.

its intrinsic order while identifying stable, passive, and approximately linear dynamics in the latent space. Simultaneously, it preserves the ability to reconstruct the original physical quantities, making it a powerful tool for structure-preserving system identification across a wide range of applications.

To further enhance the presented framework, autoencoders that inherently satisfy the projection properties could be integrated to ensure the necessary conditions for transferring the PH properties to physical space are met. Additionally, several promising extensions are conceivable. For instance, incorporating state-dependent system matrices could enable the learning of nonlinear PH systems. Another potential advancement involves encoding each physical domain separately while explicitly learning the coupling dynamics in the latent representation. This approach could improve the interpretability of the latent coordinates and prove particularly beneficial in scenarios where different physical domains operate on different scales, exhibit markedly different behaviors, or vary in the number of associated DOFs. Furthermore, the ability to identify input-affine systems highlights the framework's suitability for control applications, particularly where efficient control of complex systems is required. Moreover, the APHIN framework can already be adapted to identify a higher-dimensional lifted linear system in the latent space, which may offer advantages for certain nonlinear systems.

Despite these advantages, the framework's inherent design imposes limitations on its applicability. Specifically, it is restricted to dynamical systems that are both stable and passive–an intentional constraint that renders it unsuitable for other system classes. Furthermore, while the identified system in the latent space may generalize well to unseen scenarios, the surrounding autoencoder remains susceptible to errors when extrapolating beyond the training regime. Another notable sensitivity lies in the framework's dependence on data scaling and hyperparameter tuning, particularly the weighting of different loss terms, which can significantly impact performance. Additionally, the method has not yet been tested in more realistic scenarios where only partial or low-quality observations, affected by measurement noise, are available. Since the framework relies on high-fidelity data including accurate time-derivative information, its overall accuracy is inherently linked to data quality. Addressing this challenge, i.e. developing a method capable of performing latent model discovery even in a low-data, high-noise regime, is the focus of the following chapter.

The code used to generate the presented results is accessible via

https://github.com/Institute-Eng-and-Comp-Mechanics-UStgt/APHIN

with a persistent release in [KneiflRettbergHerb24] under MIT Common License, while the data can be found in [KneiflRettbergFehr24].

Chapter 8

Generative, Physically Consistent and Uncertainty-Aware Latent Discovery

In der Nacht sind wir frei.

In Verruf, Frei

Many system identification methods inherently constrain the identified model to a predefined class to facilitate efficient learning, such as restricting models to stable linear systems, as demonstrated in Chapter 7. While these constraints can simplify the learning process and ensure structural properties in many applications, they may impose excessive rigidity in scenarios where the assumed model structure is not valid. In such cases, more flexible and adaptive identification approaches are preferable, allowing for a broader range of dynamical behaviors.

Moreover, data-driven Reduced Order Models (ROMs) produced by black-box modeling techniques or system identification methods often lack robust mechanisms for Uncertainty Quantification (UQ) and thus struggle with the challenge of lower quality data, c.f. Chall. 6. Consequently, scenarios characterized by sparse data or the availability of only noisy measurements present a significant challenge, resulting in a lack of adequate estimates of model reliability. Generative models have shown remarkable capabilities in capturing complex patterns and addressing uncertainties in high-dimensional data, notably in natural language processing and computer vision.

In the context of ROMs, they offer transformative potential by enabling efficient synthesis of Partial Differential Equation (PDE) solutions, enhancing noise robustness, and naturally incorporating uncertainty quantification. While many generative models approximate the underlying data distribution through probabilistic learning and thus enable efficient sampling, their reliance on observational data lacks an explicit notion of dynamics. This limitation undermines reliability, explainability, and accuracy when extrapolating to

unseen scenarios in scientific computing. The absence of explicit physical constraints can lead to inaccurate or non-generalizable predictions, restricting their applicability to physical dynamical systems, cf. Chall. 3. Bridging this gap requires balancing the physical consistency of data-driven discovery methods with the flexibility of generative models and the benefits of UQ.

To address the lack of physical consistency in conventional generative methods, we introduce a novel *physical generative model* presented in [ContiEtAl24] and this thesis. It seamlessly integrates dimensionality reduction, dynamical system identification, and UQ within a unified variational framework. This approach eliminates the need for explicit prior knowledge while enabling the construction of flexible and nonlinear model representations through rich, interchangeable function libraries. Within this framework, both the governing dynamics and the low-dimensional latent variables used to describe the system are identified in an interpretable manner. Starting from a limited set of high-dimensional, noisy observations, the proposed method constructs efficient ROMs by leveraging variational autoencoders for dimensionality reduction in conjunction with a newly developed variational extension of Sparse Identification of Nonlinear Dynamics (SINDy). This combination facilitates the discovery of compact yet expressive representations of the system's dynamics.

The proposed method comprises three key components: Variational Encoding of Noisy Inputs (VENI), Variational Identification of Nonlinear Dynamics (VINDy), and Variational Inference with Credibility Intervals (VICI), each fulfilling a distinct role in the identification and uncertainty quantification process. First, VENI is employed to infer the distribution of reduced coordinates, while VINDy identifies the distribution of the coefficients of function terms governing the underlying equations. Once trained offline, the framework enables the rapid generation of full field solutions for previously unseen parameter instances and initial conditions with low computational cost by VICI, which evolves an ensemble of generated ROMs.

The major contributions of the presented framework are as follows:

1. A novel data-driven system identification framework, VINDy, is introduced which leverages variational inference for governing equation discovery while incorporating principled uncertainty quantification.

2. Dimensionality reduction (VENI), system identification (VINDy), and UQ on the model and state predictions (VICI) are unified into a cohesive optimization framework. This integrated approach is efficiently trainable through standard deep learning techniques, ensuring both scalability and robustness.

3. The proposed methodology is validated on both low-dimensional chaotic dynamical systems and high-dimensional PDE benchmarks. The results demonstrate the framework's capability to accurately recover governing equations, thereby enabling the construction of interpretable and physically meaningful latent models.

The following sections systematically introduce the core components of the proposed framework. First, an overview of Variational Autoencoders (VAEs) is provided. This is followed by a detailed explanation of each step within the proposed methodology. Next, we demonstrate the effectiveness of the newly introduced VINDy method in identifying interpretable and accurate dynamical models for the Lorenz system under varying noise intensities and sources. Finally, the overall framework–comprising VENI, VINDy, and VICI–is evaluated on benchmark problems involving applications in structural mechanics and fluid dynamics.

8.1 Variational Autoencoders

Unlike standard autoencoders, VAEs do not map input data to a single reduced coordinate but instead learn a probability distribution over the latent space, as illustrated in Fig. 8.1. Typically, this distribution over the latent variables is assumed to follow a Gaussian distribution. In this formulation, the encoder network parameterizes the distribution over the latent space by producing both a mean $\boldsymbol{\mu}_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x}) \in \mathbb{R}^r$ and a variance $\boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x}) \in \mathbb{R}^r$ estimate. These outputs define a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x})))$, where the covariance matrix is a diagonal matrix $\mathrm{diag}(\boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x})) \in \mathbb{R}^{r\times r}$ with the variances $\boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x})$ along the main diagonal. We explicitly chose a diagonal structure to ensure an independent realization of the latent variables. The output of the encoder, represented as $\left[\boldsymbol{\mu}_{\boldsymbol{\phi}_\mathrm{e}}^\intercal \ \boldsymbol{\vartheta}^{2\,\intercal}_{\boldsymbol{\phi}_\mathrm{e}}\right]^\intercal \in \mathbb{R}^{2r}$, has twice the dimensionality of the reduced latent space r.

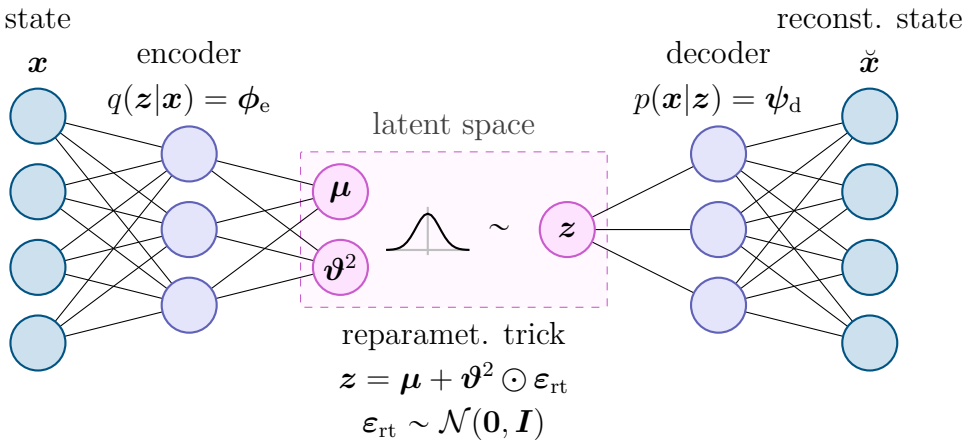


Figure 8.1: Schematic representation of an VAE. For the sake of clarity, $\boldsymbol{\mu}_{\boldsymbol{\phi}_\mathrm{e}}$ and $\boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_\mathrm{e}}$ are abbreviated as $\boldsymbol{\mu}$ and $\boldsymbol{\vartheta}^2$.

A state in the original space can be reconstructed by drawing a sample from the latent distribution, i.e. $\boldsymbol{z} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x}), \mathrm{diag}(\boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x}))\right)$ and processing it with a probabilistic decoder. This probabilistic formulation encourages continuity and smoothness in the latent space meaning that nearby points in the reduced space correspond to similar

reconstructions, and promote disentanglement of reduced coordinates [HigginsEtAl18, AlemiEtAl16]. Due to these properties, VAEs are widely employed in generative tasks, where new data samples can be synthesized by drawing random samples from the latent distribution. However, beyond their generative capabilities, VAEs also serve as powerful tools in dimensionality reduction [MahmudHuangFu20] and dynamical system applications [SoleraRicoEtAl24, SimpsonEtAl24, BotteghiGuoBrune22]. For a more comprehensive theoretical foundation and methodological details, readers may refer to [KingmaWelling13, RezendeMohamedWierstra14].

Variational Inference In particular, VAEs address the problem of variational inference, aiming to approximate the posterior probability distribution (*posterior*) $p(\boldsymbol{z}|\boldsymbol{x})$ over a hidden latent variable $\boldsymbol{z}$, given observations $\boldsymbol{x}$ (in this case, state space data). The fundamental assumption is that each high-fidelity data point $\boldsymbol{x}$ can be generated from a corresponding latent variable $\boldsymbol{z}$ following a probability distribution $p(\boldsymbol{x}|\boldsymbol{z})$. The objective is to determine an optimal output distribution within a selected family of distributions $\mathscr{P}$ over $\mathbb{R}^n$, such that the generative process implemented by the decoder–from $\boldsymbol{z}$ to $\boldsymbol{x}$–effectively models the observed data. The optimal distribution is determined by maximizing the likelihood of each $\boldsymbol{x}$ in the training dataset $\mathcal{D}$ under the generative model, i.e maximize

$$p(\boldsymbol{x}) = \int_{\mathbb{R}^r} p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}, \tag{8.1}$$

where the prior distribution (*prior*) over latent variables, denoted as $p(\boldsymbol{z})$, is predefined and can be selected from various probability distributions. Typically, it is assumed to follow a Gaussian distribution $\mathcal{N}(\mathbf{0}_r, \boldsymbol{I}_r)$ with zero mean and unit variance.

Evidence Lower-Bound However, the integral in (8.1) is in general intractable, making direct optimization infeasible. To circumvent this issue, the Evidence Lower-Bound (ELBO)

$$\log p(\boldsymbol{x}) - \mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}|\boldsymbol{x})) = \mathbb{E}_{\boldsymbol{z}\sim q}[\log p(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})), \tag{8.2}$$

is optimized instead. In this context, $\mathrm{KL}(\cdot)$ represents the Kullback-Leibler (KL) divergence that quantifies the discrepancy between two probability distributions. The goal is to find an approximation of the posterior $q(\boldsymbol{z}|\boldsymbol{x})$ such that samples drawn from it are likely to have generated the observed data $\boldsymbol{x}$. The ELBO serves as a lower bound for (8.1), meaning that maximizing it indirectly maximizes the original objective, i.e. the likelihood of each state $\boldsymbol{x}$ in the training dataset. Consequently, the term $\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}|\boldsymbol{x}))$ acts as an error term, measuring the divergence between the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ and the true posterior $p(\boldsymbol{z}|\boldsymbol{x})$. Minimizing this divergence improves the approximation quality, thereby tightening the ELBO on $\log p(\boldsymbol{x})$ and bringing it closer to the actual objective function.

While the left-hand side of (8.2) remains intractable, the right-hand side provides a computationally feasible alternative, enabling practical optimization through a reformulation that can be directly evaluated. The first term $\mathbb{E}_{\boldsymbol{z}\sim q}[\log p(\boldsymbol{x}|\boldsymbol{z})]$ in the right-hand side corresponds to the reconstruction objective, ensuring that the log-likelihood of $\boldsymbol{x}$ given a latent variable $\boldsymbol{z}$ sampled from the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ is maximized. In other words, this term enforces that the input data can be effectively reconstructed from a sample in the latent space. The second term $\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))$ serves as a regularization constraint, encouraging the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ to align with the prior distribution $p(\boldsymbol{z})$. This ensures that the latent representation learned by the encoder $\boldsymbol{\phi}_\mathrm{e}$ remains structured and adheres to the assumed prior distribution over the latent variables. For a detailed derivation of the ELBO, obtained by applying Bayes' theorem to the KL divergence between the approximate and true posterior distributions, we refer to [Yu20].

In VAEs, the variational inference problem is addressed by parameterizing both the likelihood and the posterior distributions using neural networks. The likelihood function is modeled by the decoder

$$\boldsymbol{\psi}_\mathrm{d} : \mathbb{R}^r \to \mathscr{P}, \quad \text{such that} \quad \boldsymbol{z} \mapsto \boldsymbol{\psi}_\mathrm{d}(\boldsymbol{z}; \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}) = p(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}), \tag{8.3}$$

mapping the latent variables $\boldsymbol{z}$ into an output probability distribution $\mathscr{P}$ over $\mathbb{R}^n$, thereby reconstructing the input data. Similarly, the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ is modeled by the encoder

$$\boldsymbol{\phi}_\mathrm{e} : \mathbb{R}^n \to \mathscr{Q}, \quad \text{such that} \quad \boldsymbol{x} \mapsto \boldsymbol{\phi}_\mathrm{e}(\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}) = q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}), \tag{8.4}$$

where the input data $\boldsymbol{x}$ is mapped into a probability distribution within a family of distributions $\mathscr{Q}$ on $\mathbb{R}^r$ in the latent space, approximating the true posterior. The weights $\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}$ and $\boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}$ of the encoder and decoder networks are optimized to identify suitable candidates for these distributions, ensuring that the learned representations effectively capture the underlying structure of the data. Correspondingly, the training objective (8.2) can be reformulated as loss function

$$L_\mathrm{VAE}(\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}) = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log \boldsymbol{\psi}_\mathrm{d}(\boldsymbol{z}; \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}})] - \mathrm{KL}(\boldsymbol{\phi}_\mathrm{e}(\boldsymbol{x}; \boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}) \parallel p(\boldsymbol{z})). \tag{8.5}$$

Reparameterization Trick Direct sampling from the distribution defined by the encoder $\boldsymbol{\phi}_\mathrm{e}$ is inherently non-differentiable, rendering gradient-based optimization techniques inapplicable. A solution arises if the family of distributions for the encoder is closed under linear transformations. The key idea is to introduce an auxiliary noise variable that is sampled from the same family of distributions but remains independent of the network parameters. Specifically, in the case of a Gaussian distribution, the noise variable $\boldsymbol{\varepsilon}_\mathrm{rt}$ is sampled as $\boldsymbol{\varepsilon}_\mathrm{rt} \sim \mathcal{N}(\mathbf{0}_r, \boldsymbol{I}_r)$, and a sample from the encoder's posterior distribution is obtained as

$$\boldsymbol{z}_i = \boldsymbol{\mu}_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x}_i) + \boldsymbol{\varepsilon}_\mathrm{rt} \odot \boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x}_i), \tag{8.6}$$

where $\odot$ represents the Hadamard product (element-wise multiplication).

This trick is known as *reparameterization trick* [KingmaWelling13] and also holds also for Laplace distributions in which case $\boldsymbol{\varepsilon}_{\mathrm{rt}} \sim \mathcal{L}(\mathbf{0}_r, \boldsymbol{I}_r)$. By restructuring the sampling process in this manner, the stochastic operation is effectively decoupled from the network parameters, shifting it outside the computational graph. This transformation ensures that the latent variable $\boldsymbol{z}$ remains differentiable with respect to the network parameters, allowing the ELBO function to be efficiently optimized using gradient-based methods.

8.2 VENI, VINDy, VICI – a variational reduced-order modeling framework with uncertainty quantification

The framework presented in the following leverages Latent Discovery of Dynamics (LDD) and variational methods to construct a generative model capable of producing physically consistent and interpretable ROMs. Additionally, it incorporates UQ into both model training and inference, embedding uncertainties in model parameters and state evolution. A distinguishing feature of our method is its preservation of the computational efficiency and scalability that are intrinsic to variational techniques circumventing the substantial computational overhead associated with alternative Bayesian methods [HirshBarajasSolanoKutz22, MarsGaoKutz24, NivenEtAl24] and weak-form approaches [MessengerBortz21, WangHuanGarikipati19]. Hence, it maintains computational feasibility without compromising the expressiveness and accuracy of the learned models. The integration of data-driven discovery of reduced variables and governing equations with uncertainty quantification enables the development of automated, interpretable, and uncertainty-aware models that can reliably capture the dynamics of complex systems.

The method consists of three components:

i) **VENI** (Variational Encoding of Noisy Inputs). A VAE is employed to identify the distribution of latent states from high-dimensional, noisy snapshots.

ii) **VINDy** (Variational Identification of Nonlinear Dynamics). A probabilistic dynamical model is discoverd based on a sparse linear combination of a predefined library of candidate functions. Hence, this component is a variational extension to SINDy [BruntonProctorKutz16] with the key distinction that it replaces the multiplicative deterministic coefficients with probability distributions. This modification enables the model to quantify uncertainty in the inferred governing equations.

iii) **VICI** (Variational Inference with Credibility Intervals). During VICI the generative model produces ROMs that are utilized to produce full-time solutions for previ-

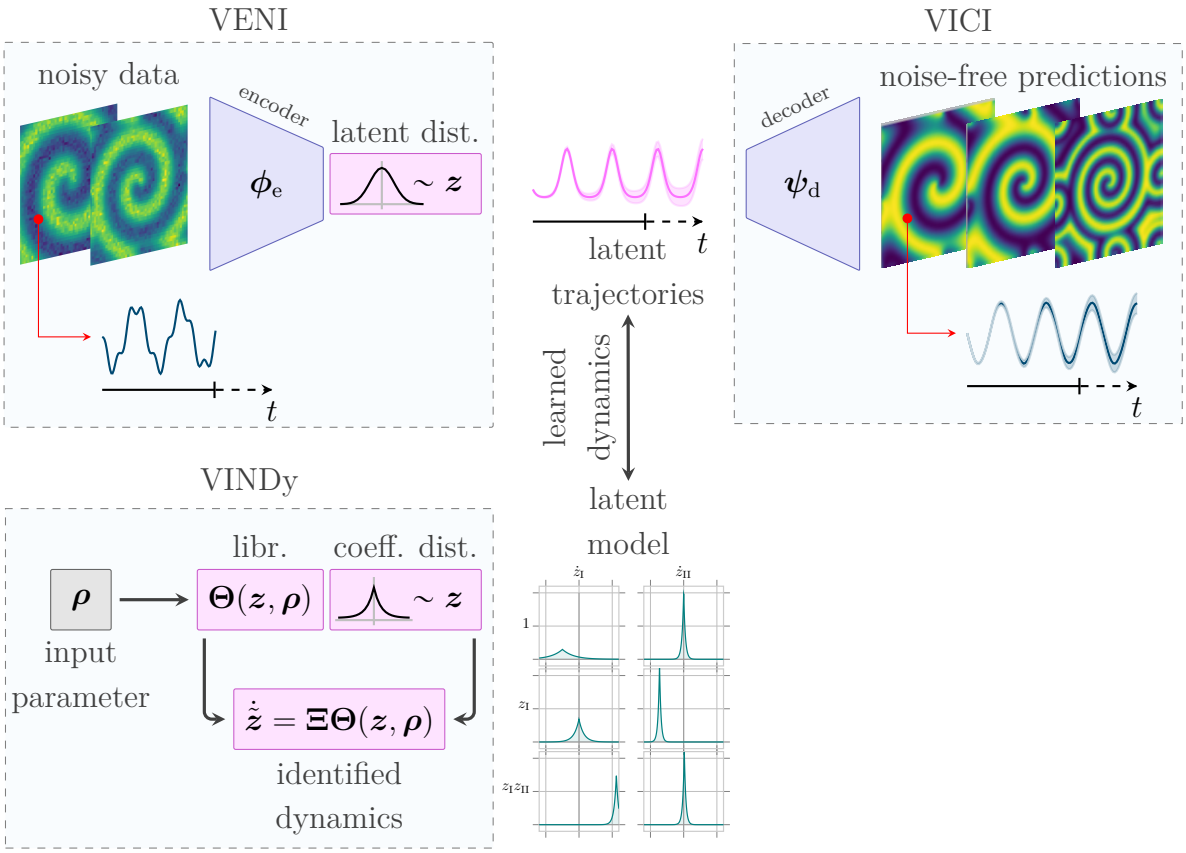


Figure 8.2: The VENI, VINDy, VICI framework. A variational autoencoder transforms high-dimensional, noisy data into low-dimensional, latent random variables (VENI). The dynamics of the latent variables are identified by VINDy in a joint training procedure. In the subsequent online phase, VICI produces noise-free, full-field solutions with uncertainty bounds.

ously unseen parameters and/or initial conditions. Simultaneously, VICI provides uncertainty quantification by leveraging the learned probabilistic representations and ensemble-based predictions.

In the offline phase, VENI and VINDy are trained simultaneously. This ensures that the low-dimensional embedding found by the VAE does not only account for reconstruction qualities but also for the dynamics discovered of VINDy. In the online phase, VICI can be utilized to produce physically consistent ROMs that efficiently and reliably approximate full-time solutions. The complete workflow is illustrated in Fig. 8.2.

8.2.1 VENI – Variational Encoding of Noisy Inputs

In VENI, a VAE is employed to encode the high-dimensional, noisy data into a low-dimensional latent representation. The general procedure follows the former explana-

tions on VAEs. An encoder $\boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x}\,;\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}) : \mathcal{X} \to \mathscr{Q}$ maps $\boldsymbol{x}$ onto the low-dimensional approximated posterior distribution $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}) = \boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}})$ within a preselected family of distributions $\mathscr{Q}$ on the low-dimensional, latent space $\mathcal{Z} \subset \mathbb{R}^r$; and a decoder $\boldsymbol{\psi}_{\mathrm{d}}(\boldsymbol{z}\,;\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}) : \mathcal{Z} \to \mathscr{P}$ transforms samples from the latent coordinates $\boldsymbol{z}$ into an output probability distribution $p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}) = \boldsymbol{\psi}_{\mathrm{d}}(\boldsymbol{z};\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}})$ within a chosen family of distributions $\mathscr{P}$ on the high-dimensional, physical state space $\mathcal{X} \subset \mathbb{R}^n$.

The families of distributions $\mathscr{Q}$ and $\mathscr{P}$ must be chosen specifically so that the weights, $\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}$ and $\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}$, can be optimized using standard machine learning techniques. A common approach in VAEs is to choose the output distribution family $\mathscr{P}$ for the decoder to be Gaussian. Specifically, given a latent variable $\boldsymbol{z}$, the likelihood function is modeled as

$$\boldsymbol{\psi}_{\mathrm{d}}(\boldsymbol{z};\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}) = p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\psi}_{\mathrm{d}}}(\boldsymbol{z};\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}), \vartheta^2 \boldsymbol{I}_n\right). \tag{8.7}$$

The mean function $\boldsymbol{\mu}_{\boldsymbol{\psi}_{\mathrm{d}}}(\boldsymbol{z};\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}) \in \mathbb{R}^n$ is implemented via a multi-layer feed-forward neural network, while the covariance matrix is assumed to be isotropic depending on the scalar hyperparameter ϑ^2. While other continuous probability distribution that are parameterizable by weights $\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}}$ could be used as well, adopting an isotropic Gaussian distribution offers a significant advantage. Specifically, the expectation term $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log \boldsymbol{\psi}_{\mathrm{d}}(\boldsymbol{z};\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}})]$ in the loss function (8.5) becomes proportional to the squared Euclidean distance $\|\boldsymbol{x} - \breve{\boldsymbol{x}}\|_2^2$ between the neural network's reconstructed mean output $\breve{\boldsymbol{x}} \coloneqq \boldsymbol{\mu}_{\boldsymbol{\psi}_{\mathrm{d}}}(\boldsymbol{z};\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}})$ and the original data $\boldsymbol{x}$, see [Yu20]. Consequently, the decoder can be trained using the classical reconstruction loss of standard autoencoders (3.14). This simplifies the training process while a probabilistic interpretation of the model is maintained.

For the approximated posterior distribution family $\mathscr{Q}$ of the encoder $\boldsymbol{\phi}_{\mathrm{e}}$, the same family is selected as assumed for the prior over the latent variables. In the following, this prior is assumed to follow a multivariate Gaussian distribution with independent latent variables, i.e. $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}_r, \boldsymbol{I}_r)$. Consequently, the encoder parameterizes a Gaussian distribution following

$$\boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}) = q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\phi}_{\mathrm{e}}}(\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}), \operatorname{diag}(\boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_{\mathrm{e}}}(\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}))\right), \tag{8.8}$$

where both, the mean vector $\boldsymbol{\mu}_{\boldsymbol{\phi}_{\mathrm{e}}}(\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}) \in \mathbb{R}^r$ and the variance vector $\boldsymbol{\vartheta}^2_{\boldsymbol{\phi}_{\mathrm{e}}}(\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}) \in \mathbb{R}^r_+$ are computed via a multi-layer feed-forward neural network. This choice offers a significant computational advantage, as the KL divergence term $\mathrm{KL}(\boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x};\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}}) \parallel p(\boldsymbol{z}))$ in (8.5) can be expressed in closed form for certain distributions, such as Gaussian and Laplacian, see e.g. [Meyer21]. This closed-form solution simplifies the optimization process and reduces computational overhead, making the variational autoencoder more efficient to train.

8.2.2 VINDy – Variational Identification of Nonlinear Dynamics

Despite VENI's capabilities in identifying low-dimensional variables from which the high-dimensional distribution can be generated, they do neither account for time nor dynamics.

Consequently, introducing a VAE in a reduced-order modeling setting is just the first step in the task of approximating the full dynamics. The second step must involve a strategy to describe the latent dynamics and evolve them in time.

In VINDy, we aim to identify a system of Ordinary Differential Equations (ODEs)

$$\begin{aligned} \dot{\boldsymbol{z}}(t, \boldsymbol{\mu}) &= \tilde{\boldsymbol{f}}(t, \boldsymbol{z}(t; \boldsymbol{\mu}); \boldsymbol{\mu}), t \in \mathcal{T}, \\ \boldsymbol{z}(0, \boldsymbol{\mu}) &= \boldsymbol{z}_0, \end{aligned} \tag{8.9}$$

that govern the evolution of the latent variables, where $\boldsymbol{z}_0 \sim \boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x}_0)$ and $\dot{\boldsymbol{z}}$ represents the latent states' time derivatives. In particular, we are interested in identifying the unknown function $\tilde{\boldsymbol{f}}$ that encodes the dynamics of the low-dimensional system. For this objective, we extend the principles of SINDy [BruntonProctorKutz16] into a probabilistic framework, allowing for uncertainty-aware discovery of the governing equations. Similarly to SINDy, we model the right-hand side of the latent governing equations,

$$\dot{\boldsymbol{z}}(t, \boldsymbol{\mu}) = \tilde{\boldsymbol{f}}(t, \boldsymbol{z}, \boldsymbol{\mu}) = \boldsymbol{\Xi}\boldsymbol{\Theta}(\boldsymbol{z}, \boldsymbol{\mu}), \tag{8.10}$$

as a linear combination of candidate basis functions taken from a predefined library $\boldsymbol{\Theta}(\boldsymbol{z}, \boldsymbol{\mu}) \in \mathbb{R}^{n_{\boldsymbol{\Theta}}}$. This library consists of predefined (nonlinear) functions constructed from the latent variables $\boldsymbol{z}$ and input parameters $\boldsymbol{\mu}$, see Section 4.2.1. The associated coefficients $\boldsymbol{\Xi} \in \mathbb{R}^{r \times n_{\boldsymbol{\Theta}}}$ determine the contribution of each candidate function to the system's dynamics.

Unlike the classical deterministic approach, we model $\dot{\boldsymbol{z}}$ and $\boldsymbol{\Xi}$ as random variables to account for uncertainty in the learned dynamics. Please note that without loss of generality the parameters $\boldsymbol{\mu}$ are assumed to be deterministic in the following. Consequently, the dependency on them for all random variables is dropped. Moreover, we assume that the system dynamics are generated by an underlying random process following formulation (8.10), defined by the unknown probability distribution of the coefficients $p(\boldsymbol{\Xi}|\dot{\boldsymbol{z}})$ with a prior distribution $p(\boldsymbol{\Xi})$ over $\mathbb{R}^{n \times n_{\boldsymbol{\Theta}}}$. In particular, we assume

$$\dot{\boldsymbol{z}}|\boldsymbol{\Xi}, \boldsymbol{z} \sim \mathcal{N}\left(\boldsymbol{\Xi}\boldsymbol{\Theta}(\boldsymbol{z}, \boldsymbol{\mu}),\, \tilde{\boldsymbol{\vartheta}}^2 \boldsymbol{I}\right), \tag{8.11}$$

where $\tilde{\boldsymbol{\vartheta}}^2$ is a scalar hyperparameter. Each entry $[\boldsymbol{\Xi}]_{ij}$ represents an independent scalar random variable, determining the contribution of the i-th candidate function to the j-th equation of system (8.10), which defines the dynamics of $\boldsymbol{z}_j$.

To learn the latent dynamics within a variational framework, the unknown distribution $p(\boldsymbol{\Xi}|\dot{\boldsymbol{z}})$ must be approximated by a posterior distribution $q(\boldsymbol{\Xi}|\dot{\boldsymbol{z}}; \boldsymbol{W}_{\boldsymbol{\Xi}})$. This posterior is sought within a predefined family of distributions, $\mathscr{Q}_{\mathrm{VINDy}}$, over $\mathbb{R}^{r \times n_{\boldsymbol{\Theta}}}$. The posterior is parameterized by trainable weights $\boldsymbol{W}_{\boldsymbol{\Xi}}$ and incorporates a prior distribution $p(\boldsymbol{\Xi})$, allowing the model to capture uncertainties in the system coefficients. Specifically, we define $\mathscr{Q}_{\mathrm{VINDy}}$ as a distribution family that can be parameterized by a set of learnable weights $\boldsymbol{W}_{\boldsymbol{\Xi}} = \{\boldsymbol{W}_{\boldsymbol{\Xi}_{\boldsymbol{\mu}}} \in \mathbb{R}^{r \times n_{\boldsymbol{\Theta}}}, \boldsymbol{W}_{\boldsymbol{\Xi}_{\tilde{\boldsymbol{\vartheta}}^2}} \in \mathbb{R}^{r \times n_{\boldsymbol{\Theta}}}\}$, where each entry $\left[\boldsymbol{W}_{\boldsymbol{\Xi}_{\boldsymbol{\mu}}}\right]_{ij}$ and $\left[\boldsymbol{W}_{\boldsymbol{\Xi}_{\tilde{\boldsymbol{\vartheta}}^2}}\right]_{ij}$ directly defines the distribution parameters of $[\boldsymbol{\Xi}]_{ij}$. Consequently,

- the weights represent the mean and variance $[\boldsymbol{\Xi}]_{ij} \sim \mathcal{N}\left(\left[\boldsymbol{W}_{\boldsymbol{\Xi}_{\boldsymbol{\mu}}}\right]_{ij}, \left[\boldsymbol{W}_{\boldsymbol{\Xi}_{\boldsymbol{\vartheta}^2}}\right]_{ij}\right)$ if the posterior family is chosen to be Gaussian.

- the weights correspond to the localization parameter and scale factor $[\boldsymbol{\Xi}]_{ij} \sim \mathcal{L}\left(\left[\boldsymbol{W}_{\boldsymbol{\Xi}_{\boldsymbol{\mu}}}\right]_{ij}, \left[\boldsymbol{W}_{\boldsymbol{\Xi}_{\boldsymbol{\vartheta}^2}}\right]_{ij}\right)$ if the posterior family is chosen to be Laplacian.

Further details on the selection of distribution families can be found in [ContiEtAl24].

It is important to note that, unlike the posteriors approximated by the encoder and decoder distributions in VAEs, the approximated posterior for $\boldsymbol{\Xi}$ does not exhibit an explicit dependence on the conditioning variable $\dot{\boldsymbol{z}}$. While the encoder and decoder posteriors explicitly take their respective conditioning variables, $\boldsymbol{x}$ and $\boldsymbol{z}$, as inputs (see (8.7)–(8.8)), the coefficients solely depend on the weights defining the corresponding distribution. To emphasize this distinction, we denote the posterior $q(\boldsymbol{\Xi}|\dot{\boldsymbol{z}}; \boldsymbol{W}_{\boldsymbol{\Xi}})$ simply as $q(\boldsymbol{\Xi})$ in the subsequent formulation.

The next step is to determine suitable posterior distributions that accurately capture the observed dynamics while minimizing the approximation error for $\boldsymbol{z}$. To achieve this, the objective in VINDy is to maximize the probability of the latent time derivatives, $p(\dot{\boldsymbol{z}})$, analogous to the objective in VAEs, which seeks to maximize the data likelihood $p(\boldsymbol{x})$. To derive a practically computable loss function from the theoretical objective, we start with the KL divergence between the approximated posterior $q(\boldsymbol{\Xi})$ and the true one $p(\boldsymbol{\Xi}|\dot{\boldsymbol{z}})$ formulated as

$$
\begin{aligned}
\mathrm{KL}(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi}|\dot{\boldsymbol{z}})) &= \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}[\log q(\boldsymbol{\Xi})] - \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}[\log p(\boldsymbol{\Xi}|\dot{\boldsymbol{z}})] \\
&= \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}[\log q(\boldsymbol{\Xi})] - \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}[\log p(\boldsymbol{\Xi}|\dot{\boldsymbol{z}},\, \boldsymbol{z})] \\
&= \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}[\log q(\boldsymbol{\Xi})] - \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log \frac{p(\boldsymbol{\Xi},\, \boldsymbol{z})p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi},\, \boldsymbol{z})}{p(\dot{\boldsymbol{z}}|\boldsymbol{z})p(\boldsymbol{z})}\right] \\
&\overset{\text{(by indep. of } \boldsymbol{\Xi} \text{ and } \boldsymbol{z})}{=} \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}[\log q(\boldsymbol{\Xi})] - \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log \frac{p(\boldsymbol{\Xi})p(\boldsymbol{z})p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi},\, \boldsymbol{z})}{p(\dot{\boldsymbol{z}}|\boldsymbol{z})p(\boldsymbol{z})}\right] \\
&= \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}[\log q(\boldsymbol{\Xi})] - \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log p(\boldsymbol{\Xi})\right] - \\
&\qquad \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log(p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi},\, \boldsymbol{z}))\right] + \log p(\dot{\boldsymbol{z}}|\boldsymbol{z}) \\
&= \mathrm{KL}(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi})) - \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log(p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi},\, \boldsymbol{z}))\right] + \log p(\dot{\boldsymbol{z}}|\boldsymbol{z}).
\end{aligned}
\tag{8.12}
$$

The resulting expression is derived by applying the definition of the KL divergence, leveraging Bayes' theorem for the second term, and exploiting the assumed independence between $\boldsymbol{\Xi}$ and $\boldsymbol{z}$.

The second component required to arrive at the final loss is the KL divergence between the true conditional distribution $p(\boldsymbol{z}|\dot{\boldsymbol{z}})$ and its variational approximation $q(\boldsymbol{z})$, which can

be expressed as

$$\begin{aligned}\mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}|\dot{\boldsymbol{z}})) &= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log q(\boldsymbol{z})] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log p(\boldsymbol{z}|\dot{\boldsymbol{z}})] \\ &= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log q(\boldsymbol{z})] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log p(\boldsymbol{z})] - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log p(\dot{\boldsymbol{z}}|\boldsymbol{z})] + \log p(\dot{\boldsymbol{z}}) \\ &= \mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z})) - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log p(\dot{\boldsymbol{z}}|\boldsymbol{z})] + \log p(\dot{\boldsymbol{z}}).\end{aligned} \tag{8.13}$$

Summing up (8.12) and (8.13) results in

$$\begin{aligned}&\mathrm{KL}(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi}|\dot{\boldsymbol{z}})) + \mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}|\dot{\boldsymbol{z}})) \\ &= \mathrm{KL}(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi})) - \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log(p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi},\, \boldsymbol{z}))\right] + \log p(\dot{\boldsymbol{z}}|\boldsymbol{z}) + \\ &\quad \mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z})) - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log p(\dot{\boldsymbol{z}}|\boldsymbol{z})] + \log p(\dot{\boldsymbol{z}}) \\ &= \mathrm{KL}(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi})) + \mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z})) - \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log(p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi},\, \boldsymbol{z}))\right] + \log p(\dot{\boldsymbol{z}}).\end{aligned} \tag{8.14}$$

The final equality is justified by a standard assumption in stochastic gradient descent-based methods: During each optimization step, a single sample of $\boldsymbol{z}$ is drawn, and $\log p(\dot{\boldsymbol{z}}|\boldsymbol{z})$ approaches the expectation $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log p(\dot{\boldsymbol{z}}|\boldsymbol{z})]$ so that their difference $\log p(\dot{\boldsymbol{z}}|\boldsymbol{z}) - \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log p(\dot{\boldsymbol{z}}|\boldsymbol{z})] \approx 0$ is approximately zero.

Finally, we can reformulate (8.14) into an equation

$$\begin{aligned}&\log p(\dot{\boldsymbol{z}}) - \mathrm{KL}(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi}|\dot{\boldsymbol{z}})) - \mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}|\dot{\boldsymbol{z}})) \\ &\qquad = \mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log(p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi}, \boldsymbol{z}))\right] - \mathrm{KL}(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi})) - \mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z})),\end{aligned} \tag{8.15}$$

where the actual objective to maximize $p(\dot{\boldsymbol{z}})$ minus an error term (left-hand side) is equivalent to the right-hand side that is a tractable objective function. Hence, this equation can be seen as the ELBO equivalent in VINDy.

8.2.3 Joint Optimization of VENI and VINDy

In the following, we want to combine the two key components of our framework for a simultaneous identification of reduced variables via VENI and latent governing equations through VINDy. Thus far, these tasks have been considered in isolation by independently optimizing the respective objective functions (8.5) and (8.15). However, a joint optimization strategy can enhance the identification of structured and dynamically meaningful latent representations, particularly since the optimization objectives of VENI and VINDy partially align. Both methods aim to approximate a suitable posterior distribution $q(\boldsymbol{z})$ for the latent variables, indicating a natural interdependence between the two components.

Hence, VENI and VINDy are simultaneously solved in an unified offline phase by maxi-

mizing

$$\max_{\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}, \boldsymbol{W}_{\boldsymbol{\Xi}}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \Bigg[\underbrace{\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log \boldsymbol{\psi}_\mathrm{d}(\boldsymbol{z})]}_{\text{reconstruction}} - \underbrace{2\mathrm{KL}(\boldsymbol{\phi}_\mathrm{e}(\boldsymbol{x}) \parallel p(\boldsymbol{z}))}_{\text{posterior for } \boldsymbol{z}} + \underbrace{\mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log(p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi}, \boldsymbol{z}))\right]}_{\text{latent dynamics}} - \underbrace{\mathrm{KL}(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi}))}_{\text{posterior for } \boldsymbol{\Xi}} \Bigg]. \tag{8.16}$$

This optimization objective represents a combination of all terms appearing in (8.5) and (8.15) for a given training dataset $\mathcal{D}$. In this context, the VINDy latent posterior $q(\boldsymbol{z})$ is substituted with the VAE latent posterior $\boldsymbol{\phi}_\mathrm{e}(\boldsymbol{x})$. This adjustment is made in order to obtain a unified approximation for the posterior from which samples can be drawn.

By adopting this joint learning strategy, we simultaneously learn the distribution of the coefficients $\boldsymbol{\Xi}$ along with the encoder $\boldsymbol{\phi}_\mathrm{e}$ and decoder $\boldsymbol{\psi}_\mathrm{d}$ distributions. Since both the encoder and decoder are modeled as neural networks parameterized by weights $\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}$ and $\boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}$, and the distribution of $\boldsymbol{\Xi}$ is also parameterized by trainable weights $\boldsymbol{W}_{\boldsymbol{\Xi}}$, it becomes straightforward to optimize all the unknown parameters $\boldsymbol{W}_{\boldsymbol{\phi}_\mathrm{e}}, \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}}, \boldsymbol{W}_{\boldsymbol{\Xi}}$ through backpropagation using standard gradient-based methods.

Reformulation of the loss In order to transform the probabilistic loss function (8.16), into a data-driven practical optimization problem, a series of steps must be taken. These steps include the addition of a full dynamics regularization term, the replacement of expectation terms, the application of the reparameterization trick for differentiable sampling, and the use of closed-form expressions for the KL divergence terms. The reparameterization trick is applied to ensure the differentiability of the sampling process for latent variables from the encoder's approximated posterior distribution during training, c.f. Section 8.1. In addition to samples of the latent states $\boldsymbol{z}_i$, samples of the latent time derivatives $\dot{\boldsymbol{z}}_i$ are also required for computing the loss function (8.18). Based on the reparameterization trick (8.6), these samples can be obtained by applying the chain rule to the derivatives of the full-state variables $\boldsymbol{x}$. Specifically, the latent state derivative samples $\dot{\boldsymbol{z}}_i$ can be expressed as

$$\dot{\boldsymbol{z}}_i = \nabla_{\boldsymbol{x}} \boldsymbol{z}(\boldsymbol{x}_i)\dot{\boldsymbol{x}}_i + \boldsymbol{\varepsilon}_\mathrm{rt} \odot \nabla_{\boldsymbol{x}} \boldsymbol{\mu}_{\boldsymbol{\phi}_\mathrm{e}}(\boldsymbol{x}_i)\dot{\boldsymbol{x}}_i. \tag{8.17}$$

As detailed in Section 8.2.1, the reconstruction loss in (8.16) becomes proportional to the squared Euclidean distance between the data $\boldsymbol{x}_i$ and the mean output of the decoder $\hat{\boldsymbol{x}}_i$ when an isotropic Gaussian distribution is assumed for the decoder, as in (8.7). This implies that $\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log \boldsymbol{\psi}_\mathrm{d}(\boldsymbol{z}; \boldsymbol{W}_{\boldsymbol{\psi}_\mathrm{d}})] \propto \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2$. Similarly, the latent dynamics loss is proportional to the squared Euclidean distance between the latent time derivatives $\dot{\boldsymbol{z}}_i$ and their approximation $\boldsymbol{\Xi}_i \boldsymbol{\Theta}(\boldsymbol{z}_i, \boldsymbol{\mu})$ when the posterior distribution of $\dot{\boldsymbol{z}}$ is assumed to be Gaussian, as in (8.11). This leads to $\mathbb{E}_{\boldsymbol{\Xi}\sim q(\boldsymbol{\Xi})}\left[\log(p(\dot{\boldsymbol{z}}|\boldsymbol{\Xi}, \boldsymbol{z}))\right] \propto \|\dot{\boldsymbol{z}}_i - \boldsymbol{\Xi}_i \boldsymbol{\Theta}(\boldsymbol{z}_i, \boldsymbol{\mu})\|_2^2$.

As already mentioned, under the assumption that the posterior distributions $q(\boldsymbol{\Xi})$ and $q(\boldsymbol{z}|\boldsymbol{x}_i) = \boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x})$ belong to the same family of distributions as the corresponding priors, the KL divergences can be expressed in closed form for Gaussian or Laplace distributions, see e.g. as described in [ContiEtAl24, Appendix A.2.]. By minimizing these KL divergences, the posterior distributions are pushed closer to the selected priors.

All these steps lead to the reformulated data-driven loss

$$\begin{aligned}\min_{\boldsymbol{W}_{\boldsymbol{\phi}_{\mathrm{e}}},\boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}},\boldsymbol{W}_{\boldsymbol{\Xi}}} \Bigg(\frac{1}{n_s}\sum_{i=1}^{n_s} & \underbrace{\lambda_{\mathrm{rec}} \left\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\right\|_2^2}_{\text{reconstruction}} + \underbrace{\lambda_{q(\boldsymbol{z})} \mathrm{KL}\left(\boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x}_i) \parallel p(\boldsymbol{z})\right)}_{\text{posterior for } \boldsymbol{z}} \\ & + \underbrace{\lambda_{\mathrm{dyn}} \|\dot{\boldsymbol{z}}_i - \boldsymbol{\Xi}_i \boldsymbol{\Theta}(\boldsymbol{z}_i, \boldsymbol{\mu})\|_2^2}_{\text{latent dynamics}} + \underbrace{\lambda_{q(\boldsymbol{\Xi})} \mathrm{KL}\left(q(\boldsymbol{\Xi}) \parallel p(\boldsymbol{\Xi})\right)}_{\text{posterior for } \boldsymbol{\Xi}} \\ & + \underbrace{\lambda_{\mathrm{con}} \left\|\dot{\boldsymbol{x}}_i - \nabla_{\boldsymbol{z}} \hat{\boldsymbol{x}}_i(\boldsymbol{z}_i) \boldsymbol{\Theta}(\boldsymbol{z}_i, \boldsymbol{\mu}) \boldsymbol{\Xi}_i\right\|_2^2}_{\text{full dynamics}} \Bigg)\end{aligned} \tag{8.18}$$

that can be utilized in standard deep-learning frameworks. In (8.18), the expectation is approximated by the mean over all samples in the training set. Moreover, $\boldsymbol{x}_i$ and $\dot{\boldsymbol{x}}_i$ represent the full state variables and their time derivatives for the i-th training snapshot, while $\boldsymbol{z}_i$ and $\dot{\boldsymbol{z}}_i$ denote their latent counterparts. The variable $\boldsymbol{\Xi}_i$ is a sample drawn from the coefficient's approximate posterior distribution $q(\boldsymbol{\Xi})$ and $\hat{\boldsymbol{x}}_i = \boldsymbol{\mu}_{\boldsymbol{\psi}_{\mathrm{d}}}(\boldsymbol{z}_i; \boldsymbol{W}_{\boldsymbol{\psi}_{\mathrm{d}}})$ refers to the mean of the decoder output distribution generated from the corresponding latent sample $\boldsymbol{z}_i$. A visual representation of the combined VENI and VINDy procedure is provided in Fig. 8.3, illustrating how these components interact to deliver interpretable, uncertainty-aware dynamical models.

The loss function (8.18) consists of five key terms, each playing a distinct role in the training process:

i) Reconstruction loss: Ensures that the full-state variables $\boldsymbol{x}$ can be accurately reconstructed from the latent variables $\boldsymbol{z}$ through the decoder $\boldsymbol{\psi}_{\mathrm{d}}$. This term minimizes the discrepancy between the original data and its reconstruction.

ii) Posterior for latent variables: Encourages the approximated posterior distribution of the latent variables, represented by the encoder $\boldsymbol{\phi}_{\mathrm{e}}(\boldsymbol{x})$, to align with the selected prior $p(\boldsymbol{z})$. This regularization promotes a structured and well-behaved latent space.

iii) Latent dynamics loss: Ensures that the identified dynamical system in the latent space accurately matches the observed data. This term minimizes the difference between the latent time derivatives $\dot{\boldsymbol{z}}$ and their predicted values from the identified dynamics model.

iv) Posterior for VINDy coefficients: Compels the approximated posterior distribution

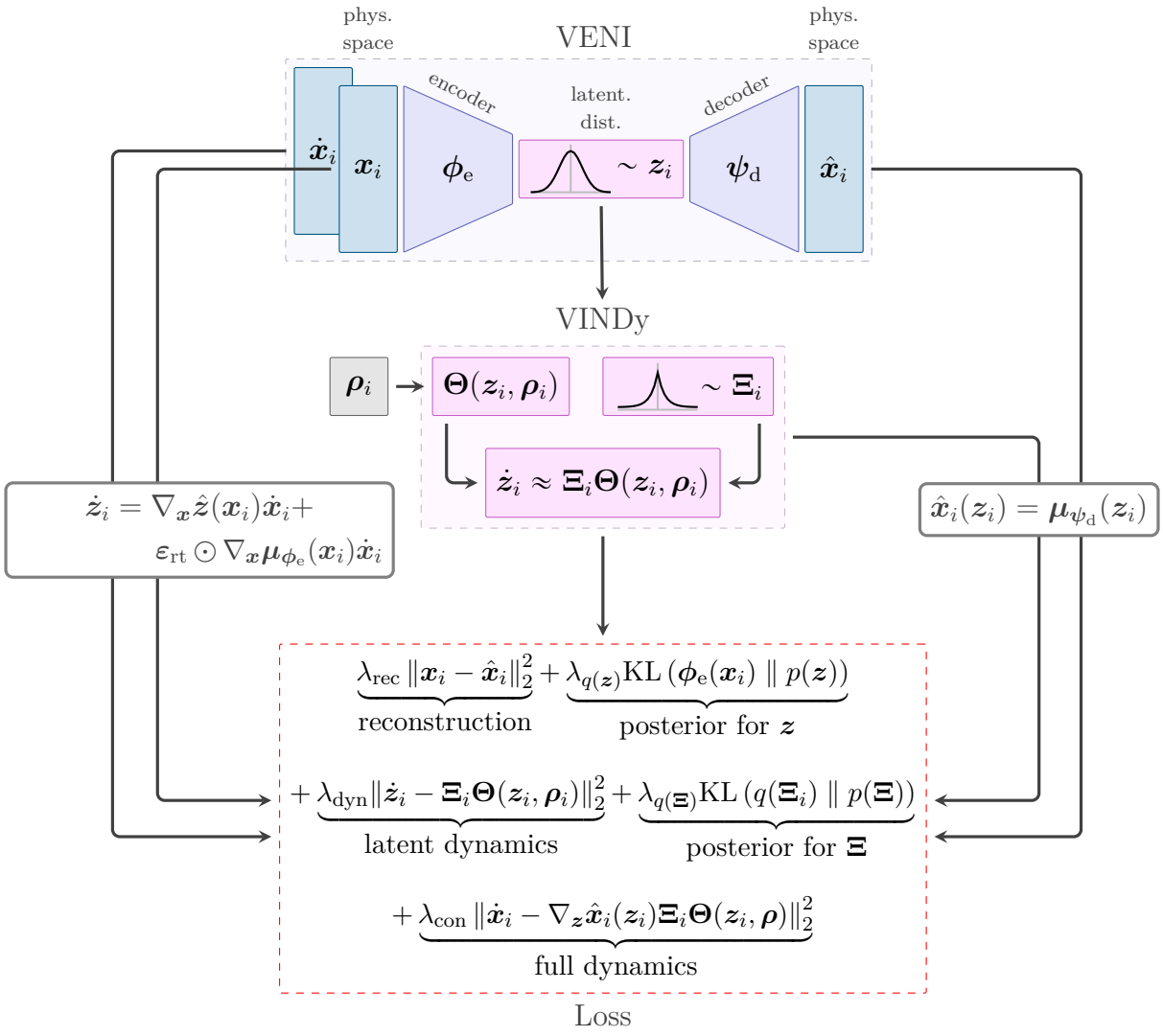


Figure 8.3: A schematic representation of the VENI and VINDy steps. The encoder maps the snapshot data $\boldsymbol{x}$ into the corresponding posterior distributions $\boldsymbol{\phi}_\mathrm{e}(\boldsymbol{x})$, from which we sample the latent states $\boldsymbol{z}$. Latent states are passed to both the decoder and VINDy setup. Within the decoder, the sampled latent states $\boldsymbol{z}$ are used to reconstruct the full-state distributions. Simultaneously, in the VINDy framework, the coefficients $\boldsymbol{\Xi}$ governing the dynamical model are sampled from the posterior $q(\boldsymbol{\Xi})$. These coefficients, in combination with the sampled latent states $\boldsymbol{z}$ and system parameters $\boldsymbol{\mu}$ are used for the identification of the dynamical system representation.

of the VINDy coefficients $q(\boldsymbol{\Xi})$ to remain close to the selected prior $p(\boldsymbol{\Xi})$. This regularization helps maintain sparsity and robustness in the identified dynamics.

v) Full dynamics loss: An additional regularization term as proposed in [ChampionEtAl19], ensuring consistency between the reference time derivatives of

the data $\dot{\boldsymbol{x}}_i$ and the mean value of their reconstruction from the approximated latent time derivatives $\nabla_{\boldsymbol{z}}\hat{\boldsymbol{x}}_i(\boldsymbol{z}_i)\boldsymbol{\Xi}_i\boldsymbol{\Theta}(\boldsymbol{z}_i, \boldsymbol{\mu})$. This term aligns the latent dynamics with the observed full-state dynamics.

The weighting coefficients $\{\lambda_{\text{rec}},\ \lambda_{q(\boldsymbol{z})},\ \lambda_{\text{dyn}},\ \lambda_{q(\boldsymbol{\Xi})},\ \lambda_{\text{con}}\} \in \mathbb{R}^+$ are hyperparameters that control the relative contribution of each term in the loss function. As a general guideline, the most dominant terms are related to the reconstruction loss and the identification of latent dynamics, i.e. λ_{rec} and λ_{dyn}. These weights should be orders of magnitude larger than the remaining coefficients $\lambda_{q(\boldsymbol{z})}$, $\lambda_{q(\boldsymbol{\Xi})}$, and λ_{con}, which are associated with regularization terms. This prioritization ensures that the primary focus remains on accurate state reconstruction and reliable dynamics identification. In cases where the learned latent variables exhibit very small magnitudes, the latent dynamics loss might appear deceptively low due to scaling effects rather than accurate dynamics identification. To mitigate this issue, the coefficient λ_{con} can be increased, emphasizing the full-dynamics consistency loss, which ensures that the identified dynamics align with the original time derivatives of the full state. This adjustment helps maintain consistency across scales and prevents underestimation of the latent dynamics loss.

For the KL divergence terms ($\lambda_{q(\boldsymbol{z})}$ and $\lambda_{q(\boldsymbol{\Xi})}$), a prioritized weighting strategy similar to the one used in β-VAEs [HigginsEtAl17] can be applied. This approach encourages more efficient latent encoding and promotes disentanglement of the latent variables, resulting in a more structured and interpretable latent space representation. Ultimately, the choice of hyperparameters is problem-dependent and must be carefully adjusted to the specific dataset. Iterative tuning and cross-validation can significantly improve training stability and convergence performance. Proper regulation of these hyperparameters ensures a balanced optimization, where both primary objectives and regularization effects contribute to a well-behaved and robust model.

8.2.4 Adaptive Sparsity Promotion via PDF Thresholding

Following the joint optimization of VENI and VINDy, a post-processing step can be applied to promote sparsity in the identified dynamical model. Specifically, coefficients with large uncertainty centered around zero–indicating an unclear or inconsistent contribution to the dynamics–are pruned. The core idea is to discard coefficients whose Probablity Density Functions (PDFs) at zero exceeds a predefined threshold, i.e.

$$\text{pdf}(0) > \tau. \tag{8.19}$$

The threshold value τ acts as a hyperparameter that controls the level of sparsity in the identified system, providing a flexible and effective mechanism for model refinement. This adaptive approach ensures that coefficients with small magnitudes are retained if their

contribution remains reliable, unlike more rigid methods such as sequential thresholding [BruntonProctorKutz16], where all coefficients below a fixed magnitude are removed indiscriminately. By selectively pruning coefficients based on uncertainty rather than magnitude alone, this method preserves essential components while enhancing the interpretability and robustness of the identified dynamics.

8.2.5 Variational Inference with Credibility Intervals

Once the offline training phase is complete, VICI can be employed during the online phase to query the fitted model and generate new solutions for unseen initial values $\boldsymbol{x}_0$ and input parameters $\boldsymbol{\mu}$. By doing so, the evolution of the system states is approximated and uncertainty intervals for the estimated solution trajectories are provided, as illustrated in Fig. 8.4. This probabilistic approach ensures that the generated predictions come with reliable uncertainty estimates, offering robust and interpretable solutions for new scenarios.

In detail, we sample n_{sim} pairs of latent initial values and model coefficients from their respective approximated posterior distributions, i.e. $\{(\boldsymbol{z}_{0i}, \boldsymbol{\Xi}_i)\}_{i=1}^{n_{\text{sim}}}$ with $\{\boldsymbol{z}_{0i}\}_{i=1}^{n_{\text{sim}}} \overset{\text{iid}}{\sim} \boldsymbol{\phi}_{\text{e}}(\boldsymbol{x}_0)$ and $\{\boldsymbol{\Xi}_i\}_{i=1}^{n_{\text{sim}}} \overset{\text{iid}}{\sim} q(\boldsymbol{\Xi})$ for a given initial value $\boldsymbol{x}_0$ in the full state space. This approach differs from standard generative methods in VAEs, where the encoder is typically ignored during testing, and new solutions are generated directly by sampling latent variables from the prior distribution. In our framework, however, the inclusion of time and dynamics requires generating physics-consistent initial conditions in the latent space. Therefore, the encoder is essential during the online phase to map initial conditions from the full state space into a latent representation that accurately reflects the dynamics of the underlying physical system.

Each pair $(\boldsymbol{z}_{0i}, \boldsymbol{\Xi}_i)$ defines an initial value problem. For each of these pairs, we integrate the corresponding system of ODEs $\dot{\boldsymbol{z}} = \boldsymbol{\Xi}_i \boldsymbol{\Theta}(\boldsymbol{z}, \boldsymbol{\mu})$ from the latent initial value $\boldsymbol{z}_{0i}$ using standard time-stepping schemes, such as Runge-Kutta methods, over a set of discrete time steps $\mathbb{T} = \{t_0, \dots t_{n_t-1}\}$. This results in a collection of latent trajectories $\left\{\{\boldsymbol{z}^{(i)}(t_0), \dots, \boldsymbol{z}^{(i)}(t_{n_t-1})\}\right\}_{i=1}^{n_{\text{sim}}}$, where the superscript $\square^{(i)}$ indicates the values belonging to the i-th initial value problem. Finally, the computed latent trajectories are passed through the decoder to obtain the mean prediction of the corresponding full-state solution trajectories $\left\{\{\hat{\boldsymbol{x}}(\boldsymbol{z}^{(i)}(t_0)), \dots, \hat{\boldsymbol{x}}(\boldsymbol{z}^{(i)}(t_{n_t-1}))\}\right\}_{i=1}^{n_{\text{sim}}}$. It is important to note that we utilize the decoder's mean prediction $\hat{\boldsymbol{x}}(\boldsymbol{z})$ instead of sampling from its posterior at each time step following $\boldsymbol{x}(t) \sim \boldsymbol{\psi}_{\text{d}}(\boldsymbol{z}(t))$. This is done to preserve the regularity in time of the latent trajectories in the full state space, ensuring that the resulting trajectories remain smooth and consistent over time. Uncertainty bounds for the approximated solution are computed from the statistical moments of the set of n_{sim} full-state trajectories, providing uncertainty estimates for the predicted solutions.

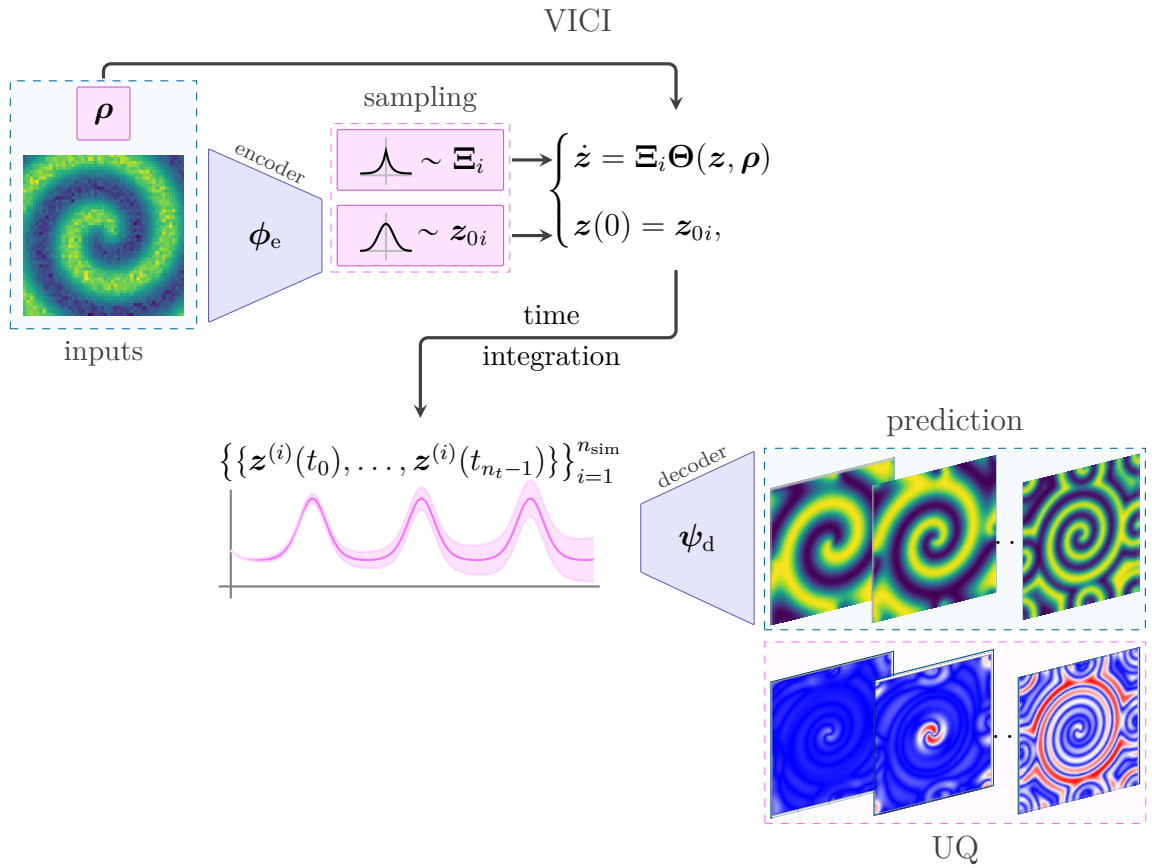


Figure 8.4: Schematic representation of the VICI procedure: new solutions are generated for a given initial condition $\boldsymbol{x}_0$ and parameter set $\boldsymbol{\mu}$. The process begins by sampling multiple instances of the corresponding latent initial values and coefficients of the dynamical model, each defining an ODE system in the latent space. Each sampled dynamical system is then integrated over time using standard time-stepping schemes, resulting in a set of latent trajectories. These latent trajectories are subsequently mapped back to the full state space through the decoder's mean output, producing full-state trajectories. Finally, predictions and UQ are derived directly from the statistical properties of the approximated solution trajectories.

In summary, the trained model naturally provides UQ for the identified dynamics, the latent states, and their time evolution, owing to its probabilistic formulation. The approximated posterior distributions of the model coefficients $q(\boldsymbol{\Xi})$ offer valuable insights into the active terms in the candidate function library, revealing their contribution to the system's dynamics, as well as the variability and reliability of the estimates. Furthermore, representing the coefficients as random variables enables a principled approach to sparsity promotion, where coefficients with large probability density values at zero are pruned. This selective pruning approach ensures that only significant terms are retained, improving the interpretability and robustness of the identified model.

8.3 Results

In the following, results on VINDy alone are presented for a low-dimensional chaotic system. Additionally, the complete framework is tested on high-dimensional examples from the field of fluid dynamics and structural mechanics. Hyperparameters regarding the individual numerical examples and settings used to discover the corresponding latent models are provided in Table 8.1.

8.3.1 Numerical Example I: Lorenz Equations – Identification of a Low-dimensional Chaotic System under Varying Noise-levels

In order to assess the potential of our novel VINDy approach alone (without VENI), we present results for a low-dimensional example. Therefore, the Lorenz equations

$$\begin{aligned} \dot{z}_\mathrm{I} &= \xi_\mathrm{I}^\mathrm{true}(z_\mathrm{II} - z_1), \\ \dot{z}_\mathrm{II} &= z_\mathrm{I}(\xi_\mathrm{II}^\mathrm{true} - z_\mathrm{III}) - z_\mathrm{II}, \\ \dot{z}_\mathrm{III} &= z_1 z_\mathrm{II} - \xi_\mathrm{III}^\mathrm{true} z_\mathrm{III}, \end{aligned} \tag{8.20}$$

originally introduced as a simplified mathematical model for atmospheric convection by mathematician and meteorologist Edward Lorenz [Lorenz63], are considered. This system exhibits chaotic behavior and has become a fundamental example in the study of nonlinear dynamical systems. In this context, $\boldsymbol{z} = [z_\mathrm{I},\ z_\mathrm{II},\ z_\mathrm{III}]^\intercal$ represents the vector of the system states and $\boldsymbol{\xi}^\mathrm{true} = [\xi_\mathrm{I}^\mathrm{true} = 10,\ \xi_\mathrm{II}^\mathrm{true} = 28,\ \xi_\mathrm{III}^\mathrm{true} = 8/3]^\intercal$ contains the chosen system coefficients.

Data Generation and Model Settings The settings to generate training and testing data as well as the hyperparameters for the model creation can be taken from Table 8.1. Each simulation of the Lorenz System (8.20) starts from normal randomly distributed initial values $\boldsymbol{z}_0 \sim \mathcal{N}([-8, 7, 27]^\intercal,\ 2\boldsymbol{I}_3)$. We consider two primary sources of uncertainty: measurement noise and model noise. To account for measurement noise, we introduce multiplicative noise to the state measurements, modeled as $\boldsymbol{z}_\mathrm{noise}(t) = \boldsymbol{\varepsilon} \odot \boldsymbol{z}(t)$ where $\boldsymbol{\varepsilon}$ is drawn from a log-normal distribution $\boldsymbol{\varepsilon} \sim \mathcal{LN}(\boldsymbol{0}_3, \varepsilon_\mathrm{meas}\boldsymbol{I}_3)$ independently for each sample. The parameter $\varepsilon_\mathrm{meas} \geqslant 0$ modulates the amount of measurement noise. The time derivatives are then numerically computed based on these noisy state measurements.

The model uncertainty is incorporated by introducing random disturbances in the model coefficients $\boldsymbol{\xi}^\mathrm{true} \sim \mathcal{N}\left([\xi_\mathrm{I}^\mathrm{true},\ \xi_\mathrm{II}^\mathrm{true},\ \xi_\mathrm{III}^\mathrm{true}]^\intercal,\ \mathrm{diag}([\vartheta^2_{\xi_\mathrm{I}^\mathrm{true}},\ \vartheta^2_{\xi_\mathrm{II}^\mathrm{true}},\ \vartheta^2_{\xi_\mathrm{III}^\mathrm{true}}]^\intercal)\right)$, which are sampled from a normal distribution. The mean values of this distribution correspond to the true coefficient values, while the standard deviations $\vartheta_{\xi_\mathrm{I}^\mathrm{true}} = \varepsilon_\mathrm{mod}\xi_\mathrm{I}^\mathrm{true}$, $\vartheta_{\xi_\mathrm{II}^\mathrm{true}} = \varepsilon_\mathrm{mod}\xi_\mathrm{II}^\mathrm{true}$,

Table 8.1: Hyperparameters for the different numerical examples.

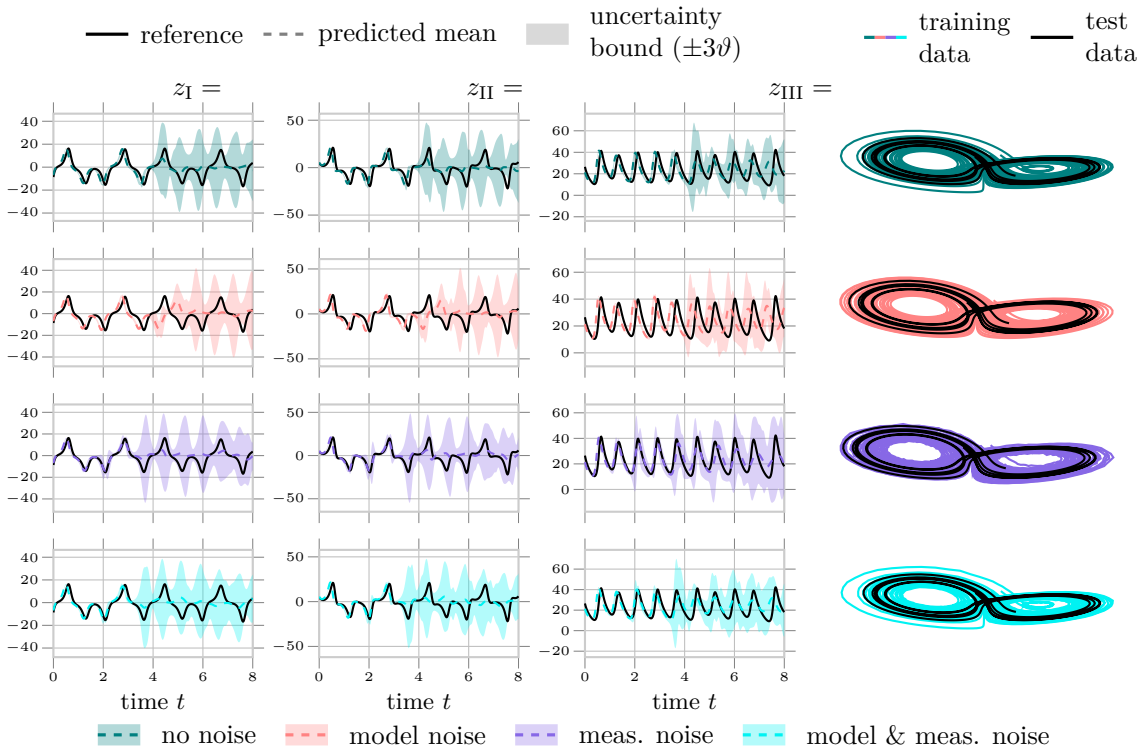
Hyperparam. Category	**Lorenz**	**Reaction-Diffusion**	**Beam**
VINDy	Polynomial library, Order: 2, Bias: Yes, Interactions: Yes, Prior: $p(\boldsymbol{\Xi}) = \mathcal{L}(\mathbf{0}, \boldsymbol{I})$	Polynomial library, Order: 3, Bias: Yes, Interactions: Yes, Prior: $p(\boldsymbol{\Xi}) = \mathcal{L}(\mathbf{0}, \boldsymbol{I})$	Polynomial library, Order: 3, Bias: Yes, Interactions: Yes, Prior: $p(\boldsymbol{\Xi}) = \mathcal{L}(\mathbf{0}, \boldsymbol{I})$
VENI	-	Number of layers: 3, Hidden units per layer: $32 \times 16 \times 8$, Activation: selu, Prior: $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$	Number of layers: 3, Neurons per layer: $32 \times 32 \times 32$, Activation: selu, Prior: $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
Dimensions	Full: $n = 3$, PCA: -, Latent: -	Full: $n = 2500$, PCA: $n_{\text{PCA}} = 32$, Latent $r = 2$	Full: $n = 7821$, PCA: $n_{\text{PCA}} = 3$, Latent $r = 1$
Training Regularization	Select best weights w.r.t. validation data, PDF thresh. $\tau = 1$	Select best weights w.r.t. validation data, PDF thresh. $\tau = 1$	Select best weights w.r.t. validation. data, PDF thresh. $\tau = 0.1$
Optimization	Adam optimizer, Learn. rate: $2.5 \cdot 10^{-3}$, Batch size: 256, Epochs: 1500, Loss: Huber	Adam optimizer, Learning rate: 10^{-3}, Batch size: 256, Epochs: 3000, Loss: MSE	Adam optimizer, Learning rate: 10^{-3}, Batch size: 32, Epochs: 2500, Loss: MSE
Data	$n_{\text{sim}}^{\text{train}} = 70$, $n_{\text{sim}}^{\text{test}} = 5$, ${n_t}^{\text{train}} = 800$, ${n_t}^{\text{test}} = 8$, $t_{\text{end}}^{\text{train}} = 800$, $t_{\text{end}}^{\text{test}} = 8$	$n_{\text{sim}}^{\text{train}} = 16$, $n_{\text{sim}}^{\text{test}} = 4$, ${n_t}^{\text{train}} = 400$, ${n_t}^{\text{test}} = 800$, $t_{\text{end}}^{\text{train}} = 20$, $t_{\text{end}}^{\text{test}} = 40$	$n_{\text{sim}}^{\text{train}} = 28$, $n_{\text{sim}}^{\text{test}} = 28$, ${n_t}^{\text{train}} = 14001$, ${n_t}^{\text{test}} = 22499$, $t_{\text{end}}^{\text{train}} = 1091\,\mu\text{s}$, $t_{\text{end}}^{\text{test}} = 1753\,\mu\text{s}$
Noise	Model: multiplicative log-normal, Measurement: multiplicative log-normal	Model: -, Measurement: multiplicative log-normal	Model: -, Measurement: multiplicative log-normal

and $\vartheta_{\xi_{\mathrm{III}}^{\mathrm{true}}} = \varepsilon_{\mathrm{mod}} \xi_{\mathrm{III}}^{\mathrm{true}}$ are set proportional to their respective magnitudes, ensuring a relative perturbation in each coefficient. The noise level is controlled by the parameter $\varepsilon_{\mathrm{mod}} \geqslant 0$, analogous to the treatment of measurement noise.

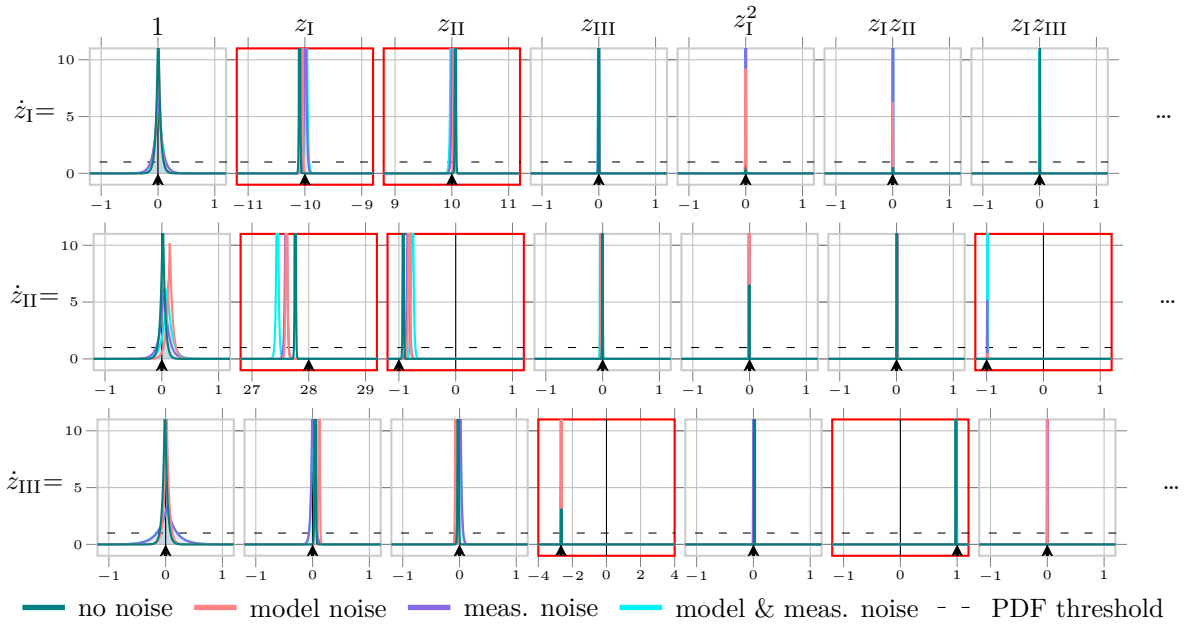
For the following investigations, we identify governing equations using VINDy under model and measurement noise. In detail, measurement noise factors of $\varepsilon_{\mathrm{meas}} \in \{0, 0.01\}$ are considered, which correspond to mean noise levels of approximately 0% and 18% in the time derivatives. Similarly, model noise factors of $\varepsilon_{\mathrm{mod}} \in \{0, 0.01\}$ are applied to the training data, introducing varying levels of coefficient uncertainty in the system. For testing, we generate a set of simulations that differ in their initial conditions from the training data while using the correct coefficient values. This ensures that the evaluation assesses the model's generalization capability to unseen initial conditions while holding the underlying system dynamics fixed.

Results The results for varying noise levels and sources are presented in Fig. 8.5a. As shown, the resulting trajectories follow the ground truth for large parts of the simulation and even in cases where the mean trajectory deviates, the uncertainty bounds mostly cover the true solution. The disparities among the approximated and reference trajectories are predominantly attributable to the chaotic nature of the Lorenz system. Concurrently, this phenomenon is manifesting as a rapid expansion in the uncertainty bounds. Hence, the bounds serve as a reliable metric for assessing the system's dynamics. The results obtained for the various noise levels and intensities are consistent with one another and do not exhibit significant variations demonstrating the robustness of the approach. However, model noise leads to the largest errors, which can be attributed to the sensitivity to model coefficients of the system. Notably, higher levels of noise lead to the uncertainty limits growing earlier, thereby underscoring the augmentation of uncertainty.

This behavior is also evident in the identified posterior distributions of the model coefficients, as shown in Fig. 8.5b. Across all noise conditions, the method consistently identifies all relevant terms present in the underlying equations, demonstrating its robustness in recovering the true governing dynamics. Simultaneously, irrelevant coefficients remain centered around zero, with their corresponding PDFs exhibiting high values, reinforcing their insignificance in the learned model. As expected, higher noise levels are reflected in broader distributions, capturing the increased uncertainty in the data. In contrast, for lower noise levels, the estimated coefficient distributions remain tightly centered around the true reference values, reflecting high confidence in the identified model. As noise increases, the model naturally produces more conservative estimates, leading to wider distributions that appropriately reflect the uncertainty introduced by the noisy data. These findings highlight the method's ability to provide accurate UQ while preserving both interpretability and reliability across different levels of observational and model noise.



(a) Temporal evolution of the system states including UQ for a test scenario. The corresponding training data for each noise scenario is shown at the right of the state trajectories.



(b) Distributions of identified coefficients; true values are indicated with a triangle at the bottom of each axis; coefficients that appear in the original equations are outlined in red.

Figure 8.5: Approximation results for the Lorenz system. Different types of noisy data generated by the Lorenz system serve as basis for the system identification. The approximated temporal state evolutions are shown in (a), while the corresponding distributions of coefficients are shown in (b).

8.3.2 Numerical Example II: Reaction-diffusion – Identification of an Oscillator

As second example, we consider a reaction-diffusion system governed by the PDEs

$$\begin{aligned}\dot{x}_{\mathrm{I}} &= \left(1 - \left({x_{\mathrm{I}}}^2 + {x_{\mathrm{II}}}^2\right)\right) x_{\mathrm{I}} + \xi_{\mathrm{I}}^{\mathrm{true}} \left({x_{\mathrm{I}}}^2 + {x_{\mathrm{II}}}^2\right) x_{\mathrm{II}} + \xi_{\mathrm{II}}^{\mathrm{true}} \left(x_{\mathrm{I}_{xx}} + x_{\mathrm{I}_{yy}}\right) \\ \dot{x}_{\mathrm{II}} &= -\xi_{\mathrm{I}}^{\mathrm{true}} \left({x_{\mathrm{I}}}^2 + {x_{\mathrm{II}}}^2\right) x_{\mathrm{II}} + \xi_{\mathrm{II}}^{\mathrm{true}} \left({x_{\mathrm{I}}}^2 + {x_{\mathrm{II}}}^2\right) x_{\mathrm{I}} \\ &\quad + \left(1 - \left({x_{\mathrm{I}}}^2 + {x_{\mathrm{II}}}^2\right)\right) x_{\mathrm{II}} \xi_{\mathrm{III}}^{\mathrm{true}} \left(x_{\mathrm{II}_{\chi_1\chi_1}} + x_{\mathrm{II}_{\chi_2\chi_2}}\right),\end{aligned} \tag{8.21}$$

where the coefficients $\xi_{\mathrm{I}}^{\mathrm{true}} = 1.0$ and $\xi_{\mathrm{II}}^{\mathrm{true}} = \xi_{\mathrm{III}}^{\mathrm{true}} = 0.01$ regulate the reaction and diffusion behaviors of the system. The problem is defined for the time interval $\mathcal{T} = [0, t_{\mathrm{end}} = 40]$ and over the spatial domain $[-10, 10]^2$ with periodic boundary conditions. The initial condition is given by

$$\begin{aligned}x_{\mathrm{I}}(\chi_1, \chi_2, 0; \mu) &= x_{\mathrm{II}}(\chi_1, \chi_2, 0; \mu) \\ &= \tanh\left(\mu\sqrt{\chi_1^2 + \chi_2^2}\cos\left((\chi_1 + i\chi_2) - \mu\sqrt{\chi_1^2 + \chi_2^2}\right)\right),\end{aligned} \tag{8.22}$$

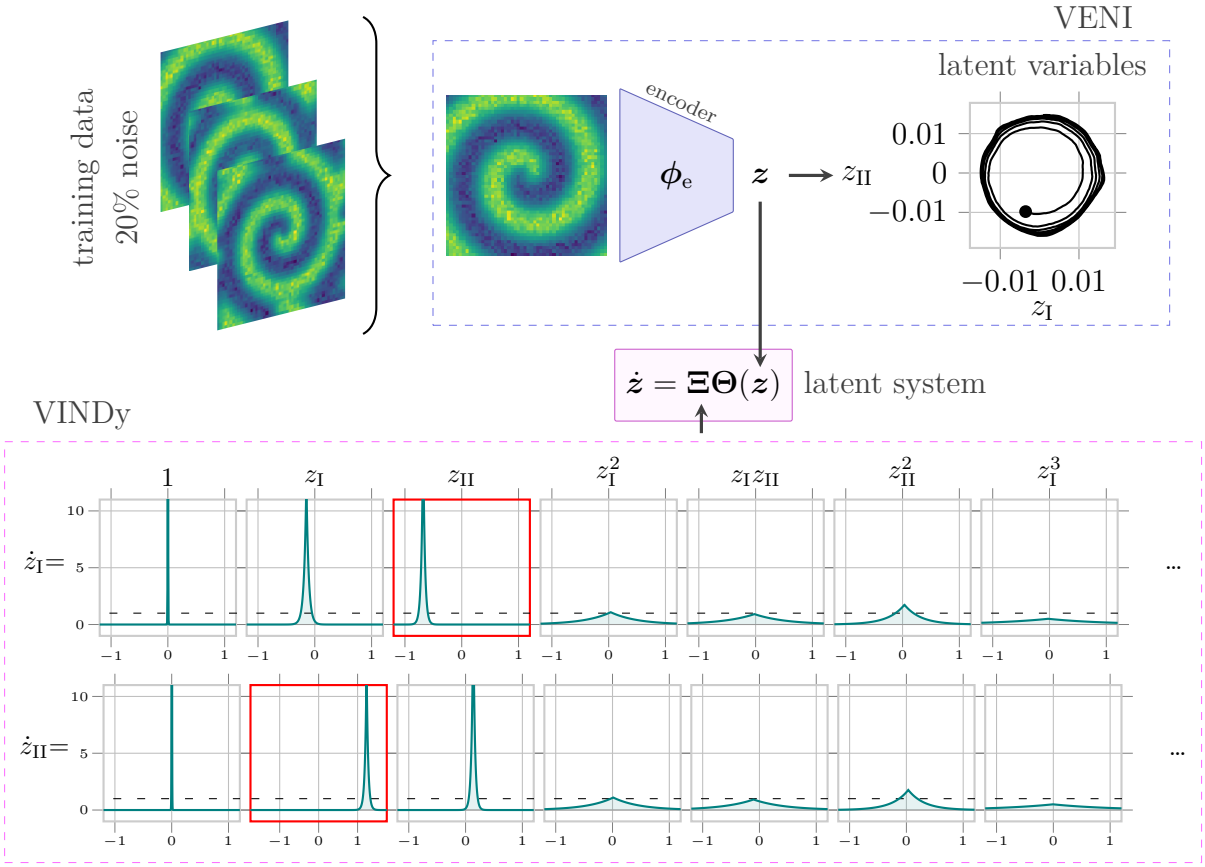
where $\mu \in [0.7, 1.1]$ is a parameter that modulates the system dynamics. In particular, μ controls the radius of spiral waves that are present in the solutions of the system (8.21). Those waves correspond to an attracting limit cycle in state space [FloryanGraham22] and can be effectively approximated using two oscillating spatial modes [ChampionEtAl19]. Our objective is to develop a generative model capable of efficiently computing the entire space-time solution for a new instance of the parameter μ, allowing for rapid and reliable prediction of the system behavior across different parameter values. We set the number of latent variables to $r = 2$ matching the number of oscillating spatial modes required to capture the spiral waves, ensuring a compact yet expressive representation of the system dynamics in the latent space.

Data Generation and Model Settings The parameter μ that modulates the system dynamics is sampled uniformly across the specified parameter domain. For each sampled parameter instance, numerical solutions to (8.21) are obtained by solving the underlying PDE with the initial condition prescribed in (8.22). The numerical integration is performed using the Fourier spectral method [Trefethen00] with a time step of $\Delta t = 0.05$ on an equispaced spatial grid, where the spatial resolution step is set to $\Delta\chi = 0.4$ leading to $n = 2500$ Degree of Freedoms (DOFs). The settings used to generate the training and testing datasets, along with the hyperparameters for model creation, are detailed in Table 8.1. For the training dataset, solutions are computed over a limited time window to restrict the temporal extent of available data. Additionally, measurement noise is introduced by applying 20% log-normal multiplicative noise to the training data, simulating realistic observational uncertainties. Before training, a dimensionality reduction step is applied to both the training and testing data. Specifically, we project the dataset

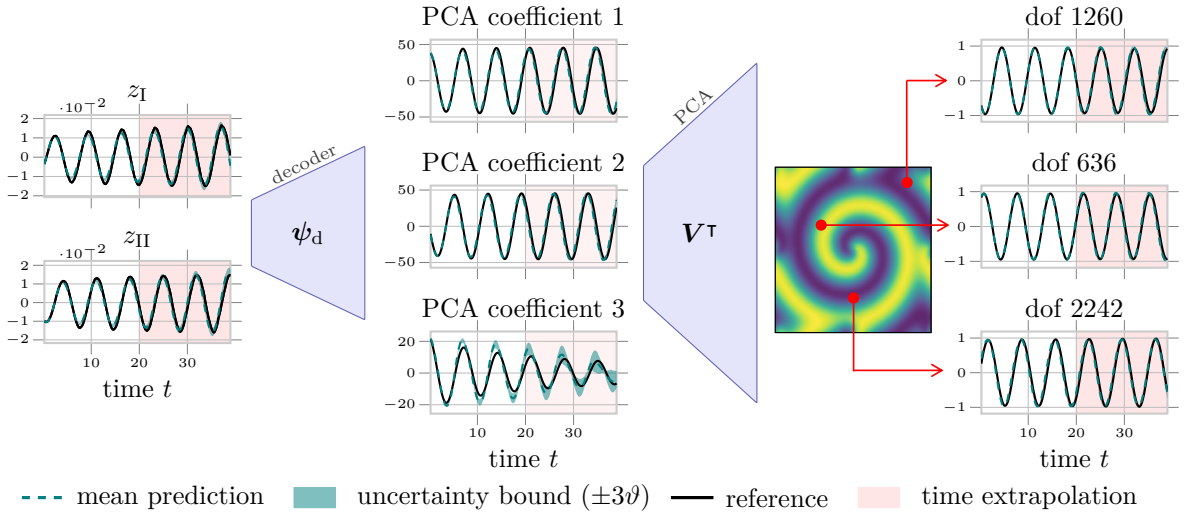
onto a reduced basis of size $n_{\text{PCA}} = 32$, which is obtained via Principal Component Analysis (PCA) on the training data. Regarding the VINDy library, no explicit parametric dependency is included, since the parameter μ affects only the initial conditions and does not appear explicitly in the governing equations.

Results The posterior distributions identified by our method for the model coefficients are shown in Fig. 8.6a. Notably, the mixed linear terms $\dot{z}_{\text{I}} = z_{\text{II}}$ and $\dot{z}_{\text{II}} = z_{\text{I}}$, which play a dominant role in the observed oscillatory dynamics, are accurately identified as significant, exhibiting high confidence in their estimates. Additionally, our approach inherently promotes sparsity in the identified dynamics. This is evidenced by the fact that most nonlinear terms are represented by distributions with high PDFs at zero, indicating that they do not contribute meaningfully to the system's behavior. Furthermore, Fig. 8.6a presents the phase space representation of the system in the identified latent coordinates, highlighting the characteristic oscillatory pattern and the presence of an attracting limit cycle. This visualization demonstrates the method's ability to preserve interpretable dynamical structures in the latent space, ensuring that the learned representation remains both compact and physically meaningful.

To evaluate the accuracy and reliability of VICI, we assess its performance on unseen initial conditions and test parameter instances that were not included during training. Specifically, we generate approximate solution predictions up to a final time of $t_{\text{end}} = 40$, which extends 20 time units beyond the training regime, effectively doubling the prediction horizon compared to the training data. These predictions, which represent the mean and standard deviation over 20 realizations of the reconstructed full solution field, are then compared against the noise-free test data in Fig. 8.6b. The results demonstrate that the method accurately captures the spatial dynamics, even when extrapolating over a time window twice as long as the training range. This highlights the robustness and generalization capability of the proposed approach in predicting long-term system behavior while maintaining reliable uncertainty quantification.



(a) Offline Phase Results: The VENI box shows how the method effectively learns the expected oscillatory behavior from noisy data. Notably, the VINDy box highlights the accurate identification of significant mixed linear terms governing the observed dynamics.



(b) A comparison of trajectories between the approximation and the reference solution in the latent as well as in the physical space. The inference time extrapolates twice the time.

Figure 8.6: Approximation results for the for the reaction-diffusion problem obtained from the offline phase (a) and online phase (b).

8.3.3 Numerical Example III: Beam MEMs Resonator – Identification of a Structural Mechanical Second-order System

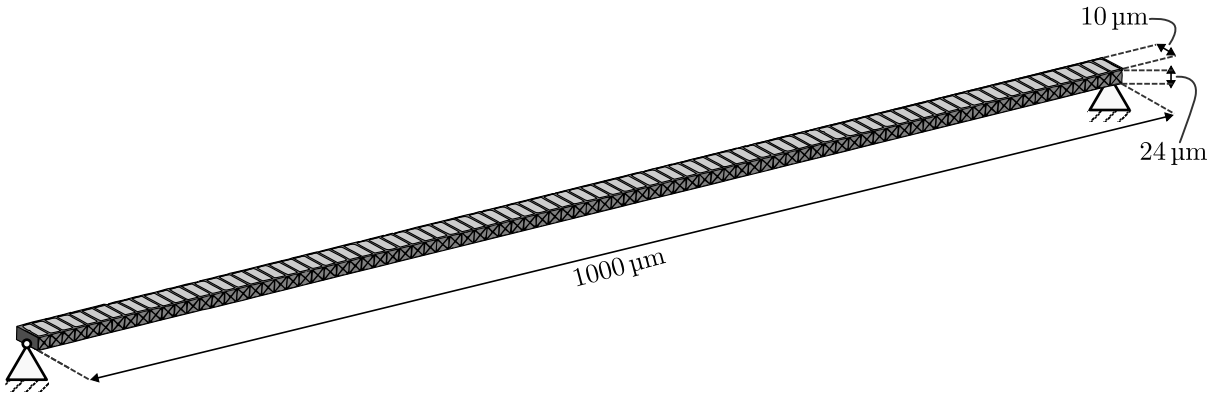
The last example represents a straight beam Micro-Electromechanical Systems (MEMs) resonator that is excited at resonance by a periodic body load that is proportional to the first vibrational eigenfunction, with an amplitude μ_{I} and frequency μ_{II}. The input parameter vector is defined as $\boldsymbol{\mu} = [\mu_{\mathrm{I}}, \mu_{\mathrm{II}}]$ and lies in a closed and bounded two-dimensional parameter space. The system is modeled as a double-clamped beam and an illustration with relevant information on the design of the model is given in Fig. 8.7a. The considered beam is composed of isotropic polysilicon [CoriglianoEtAl04], a commonly used material in MEMs applications. The governing PDEs for this structural dynamics problem under large transformations are formulated in terms of the displacements $\boldsymbol{d}$. For a more detailed discussion of the mathematical formulation, numerical implementation, and application context, please refer to [ContiEtAl23a].

To obtain a computationally feasible representation of the system, the underlying PDE model is discretized using the Finite Element Method (FEM), resulting in a high-dimensional Full Order Model (FOM). The clamped-clamped beam is a well-known structure exhibiting Duffing-like hardening behavior, which has been extensively studied in the literature [Corigliano18, FrangiGobat19]. This nonlinear behavior can be accurately described by the following simple model

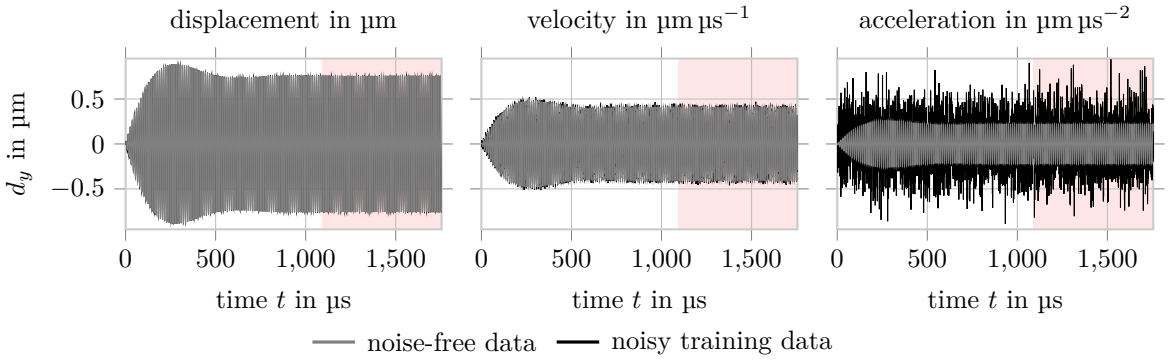
$$\ddot{z} = -{\xi_{\mathrm{I}}^{\mathrm{true}}}^2 z - 2\xi_{\mathrm{II}}^{\mathrm{true}}\xi_{\mathrm{I}}^{\mathrm{true}}\dot{z} - \xi_{\mathrm{III}}^{\mathrm{true}} z^3 - \xi_{\mathrm{IV}}^{\mathrm{true}}\mu_{\mathrm{I}}\cos(\mu_{\mathrm{II}} t), \tag{8.23}$$

where the cubic nonlinear term accounts for the hardening behavior of the system with $\xi_{\mathrm{III}}^{\mathrm{true}} > 0$. It is important to note the absence of quadratic terms, which is characteristic of the underlying physics of the beam's oscillatory response.

Data Generation and Model Settings For data generation, we consider excitation frequencies μ_{II} within the range from $0.526\,\mathrm{rad\,\mu s^{-1}}$ to $0.564\,\mathrm{rad\,\mu s^{-1}}$, with a finer sampling near the natural frequency $0.5475\,\mathrm{rad\,\mu s^{-1}}$ to capture the resonance behavior accurately. The forcing amplitude is considered for two values, $\mu_{\mathrm{I}} \in \{0.125, 0.250\}\,\mu\mathrm{N}$. The settings used for generating displacement snapshots, which are collected into the training and testing datasets, are provided in Table 8.1 along with the hyperparameters of the model identification. The training data consists exclusively of trajectories up to $t_{\mathrm{end}}^{\mathrm{train}} = 1091\,\mu\mathrm{s} < t_{\mathrm{end}}$, providing a limited time window for model learning. Additionally, the data is corrupted by additive noise, which follows a normal distribution with a scale proportional to the mean displacement level across all training simulations. We model the beam as a second-order system, following the classical structural mechanics formulation. Hence, we require to have access or compute the second time derivatives $\ddot{\boldsymbol{x}}$ in addition to the system states $\boldsymbol{x}$ and their time derivatives $\dot{\boldsymbol{x}}$.



(a) Schematic illustration of the beam MEMs resonator showing the FEM mesh. The beam is made of isotropic polysilicon with a density 2330 kg/m^3, Young modulus 167 GPa and Poisson coefficient 0.22. The first bending eigenfrequency is 0.5475 rad µs^{-1}. Dirichlet boundaries are highlighted.



(b) Noise-free and noisy trajectories of the displacement in y-direction d_y and the corresponding velocity and acceleration for a node in the center of the beam for an example training simulation. Only the first 1091 µs of the noisy data are used for training; the remaining time for which the identified model must extrapolate is highlighted in red.

Figure 8.7: Illustration of beam MEMs resonator (a) and corresponding examples of noisy training data (b).

In the context of second-order dynamical systems, noise constitutes a substantial challenge, as each numerical differentiation step amplifies it. This effect is particularly pronounced for acceleration computations, where even small levels of noise in the displacement data lead to substantial noise amplification at the acceleration level, as illustrated in Fig. 8.7b. For the beam data, the ratio between the maximum acceleration amplitude in the noisy training data and the noise-free reference data across all simulations is $4{,}52 \pm 2{,}27$. This quantifies the strong noise amplification effect in the numerical computation of acceleration, emphasizing the importance of robust modeling approaches that account for uncertainty.

We enforce the second-order structure in the identified ODE of the reduced dynamics by augmenting the reduce states with the first time derivatives $\dot{\boldsymbol{z}}$ following

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{z}\\ \dot{\boldsymbol{z}}\end{bmatrix} = \mathbf{f}(\boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{\mu}) \approx \begin{bmatrix}\dot{\boldsymbol{z}}\\ \boldsymbol{\Xi}\boldsymbol{\Theta}(\boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{\mu})\end{bmatrix}. \tag{8.24}$$

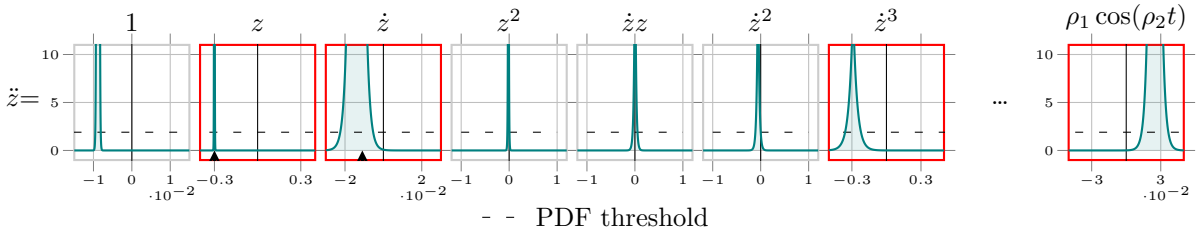
The required accelerations in the latent space $\ddot{\boldsymbol{z}}$, can be derived using the chain rule, similar to how velocities $\dot{\boldsymbol{z}}$ are computed. Alternatively, the system could be modeled as a first-order ODE system with $r = 2$ latent variables if no prior knowledge of the second-order structure of the problem is to be exploited.

With respect to the VINDy library $\boldsymbol{\Theta}$, polynomial functions up to the third degree are considered with respect to both $\boldsymbol{z}$ and $\dot{\boldsymbol{z}}$. Since the beam resonator is subjected to harmonic forcing with amplitude and frequency parameterized by μ_{I} and μ_{II}, respectively, we account for this dependency by including the library term $\mu_{\mathrm{I}}\cos(\mu_{\mathrm{II}}t)$. Moreover, the data, i.e. displacements, velocities, and accelerations, are preliminarily projected onto a lower-dimensional space using PCA derived from the noisy displacement training data as in the previous example. Although the dynamics of the system naturally evolve on a two-dimensional manifold in phase space, we successfully reduce the latent space to a single variable, setting $r = 1$. This is possible because the velocity dependence is minimal for the dominant bending mode under investigation.

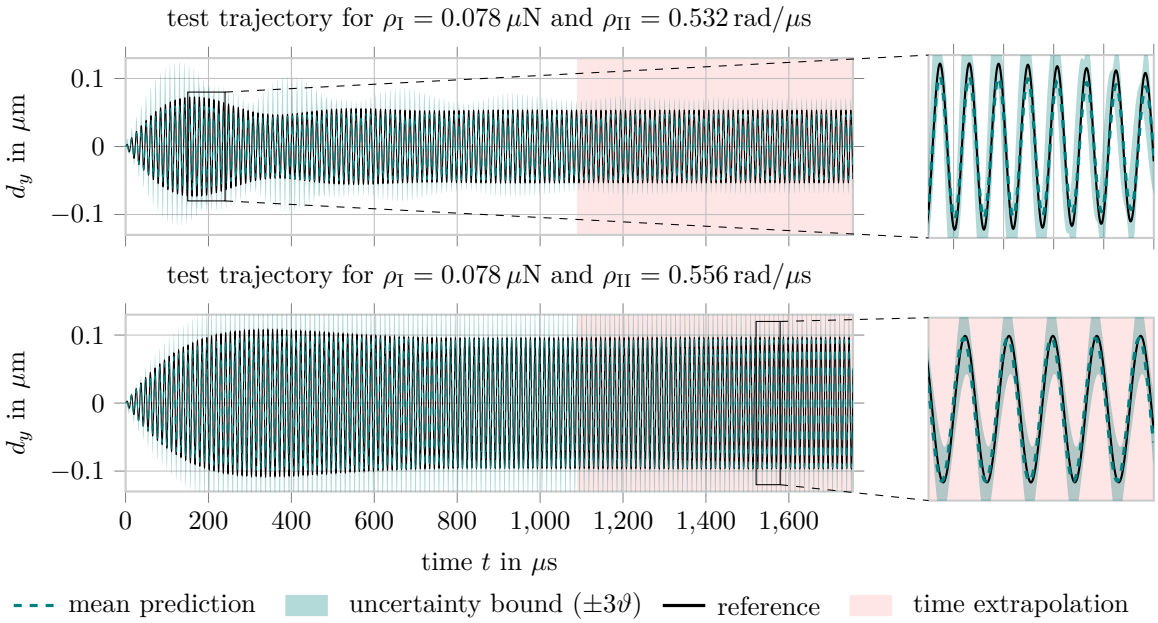
Results One of the primary objectives of VINDy is to derive a sparse, reduced-order representation of the system. To achieve this, we aim to reconstruct the underlying normal form described by (8.23) directly from observational data, circumventing the need for access to the full-order model or related structural insights. In the analyzed example, Fig. 8.8a demonstrates that the terms identified as significant by our methodology coincide with those present in the normal form, apart from a minor bias term of negligible magnitude. This result underscores the precision and reliability of our approach in capturing the system's intrinsic structure. Moreover, our method accurately recovers the coefficients governing the linear terms associated with the system's natural frequency and damping properties, specifically $-{\xi_{\mathrm{I}}^{\mathrm{true}}}^2$ and $-2\xi_{\mathrm{II}}^{\mathrm{true}}\xi_{\mathrm{I}}^{\mathrm{true}}$. Notably, this identification solely relies on noisy measurement data. Additionally, the VICI-generated mean trajectories for new parameter configurations exhibit strong agreement with reference solutions, while the associated standard deviations effectively encapsulate deviations, as illustrated in Fig. 8.8b.

8.4 Discussion

The results highlight the efficacy of the proposed VENI, VINDy, and VICI framework in constructing ROMs that do not only address the challenge of physicality (Chall. 3) but also UQ in noisy or data-scarce scenarios (Chall. 6). In particular, this is achieved by



(a) Distributions of identified coefficients; true values are indicated with a triangle at the bottom of each axis; coefficients that appear in the normal form of the observed high-dimensional system are outlined in red.



(b) A comparison of the reconstructed of latent mean trajectories and corresponding uncertainties with the reference solution for a node in the middle of the beam where the most dynamics occur.

Figure 8.8: Approximation results for the for the beam MEMs resonator obtained from the offline phase (a) and online phase (b).

integrating probabilistic modeling with data-driven discovery within a unified *physical generative* framework. A key strength of VINDy lies in its ability to identify the essential terms governing the system's dynamics, thereby facilitating the discovery of governing equations that offer deeper insight into complex physical phenomena. Accordingly, the identified ROMs reveal model descriptions that are consistent with the underlying physical principles while maintaining computational efficiency and providing robust UQ, even in scenarios characterized by high noise levels.

A distinguishing feature of the proposed approach is that it treats both state variables and model coefficients as probability distributions. This allows to effectively account for measurement noise and model uncertainty, thereby enhancing the reliability and robustness of the inferred ROMs. Several advantages arise from this probabilistic formulation. On the

one hand, the estimated probability distributions of model coefficients reveal uncertainties in the corresponding governing equation terms, enabling informed model selection and refinement. On the other hand, the incorporation of sparsity-promoting priors, such as Laplacian priors, in conjunction with PDF thresholding, leads to the construction of parsimonious models. Particularly, PDF thresholding allows that small-magnitude coefficients are retained if they significantly influence the system's dynamics. Conventional methods lack this property as they solely discard coefficients based on their magnitude. For example, in the numerical example of the beam MEMs resonator example, the forcing term–despite its small amplitude–obviously plays a crucial role in capturing the non-autonomous behavior of the system. While purely magnitude-based approaches might erroneously eliminate this term, the probabilistic formulation ensures its retention, thereby preserving the accuracy of the inferred model.

In the case of the Lorenz system, the method accurately reconstructs the governing equations by selecting the essential terms and estimating their coefficients. In the reaction-diffusion system, the primary dynamics are effectively captured as a linear oscillator, where the latent phase-space structure closely approximates the attracting limit cycle. In the MEMs resonator application, the framework autonomously discovers the governing equations, unveiling the system's *normal form*, which represents the sparsest possible description of the dynamics while explicitly revealing key coefficients (e.g., natural frequency and damping) alongside critical nonlinear contributions.

Beyond UQ on latent states and dynamical models, the framework extends uncertainty quantification to physical solution trajectories using an ensemble strategy within VICI. Due to its generative nature, the method can efficiently construct an ensemble of dynamical models via sampling, incurring virtually no additional computational cost in contrast to conventional ensemble-based techniques that require training of multiple model instances. By propagating these sampled models forward in time and mapping the resulting ensemble of latent trajectories back to the physical space, we gain direct insight into the evolution of uncertainty in the physical solution over time.

Despite its advantages, the framework, like most deep learning approaches, remains sensitive to hyperparameters such as the relative weighting of different loss terms, the choice of priors, and the selection of candidate functions within the VINDy library. While the framework effectively forecasts solution trajectories using the identified dynamical models, the feed-forward networks in the autoencoder may not generalize well to spatial patterns entirely outside the training distribution. Hence, further improvements could be achieved by leveraging more advanced neural network architectures as components of the VAEs. For instance, Graph Convolutional Neural Networks (GCNNs) could enhance the reconstruction of spatial patterns and improve generalization to previously unseen configurations.

The method's robustness to noise and its non-intrusive nature make it particularly well-suited for real-world applications involving sensor and video data [MarsGaoKutz24, ChenEtAl22a]. Furthermore, its inherent flexibility suggests strong potential for online learning, where models could dynamically update as new data become available. This could be achieved by utilizing previous posterior estimates as priors and fine-tuning the model to iteratively refine the posterior distributions. Additionally, the approach could incorporate constrained learning techniques [ChamonRibeiro20] to restrict the learning process to regions where the reconstruction error remains within acceptable limits. This would ensure that the learned dynamics are both reliable and physically meaningful, further enhancing the practical applicability of the framework.

The code used to generate the presented results is accessible via

https://github.com/jkneifl/VENI-VINDy-VICI

with a persistent release in [KneiflConti25] under MIT Common License.

Chapter 9

Conclusion and Outlook

Oye mi consejo
Que el que no oye consejos
No llega a viejo

Celia Cruz, Oye Mi Consejo

This thesis introduces multiple approaches for constructing efficient yet accurate surrogate models for structural dynamical systems, emphasizing the pivotal role of Reduced Order Models (ROMs) in bridging the gap between computational cost and modeling fidelity. In particular, the focus is placed on non-intrusive data-driven modeling of complex, high-dimensional, nonlinear, and parametric dynamical systems.

Conventional numerical simulation models serve as the foundation for a wide range of applications in science and engineering, owing to their predictive capabilities and ability to reveal insights into complex physical phenomena. However, their practical use is often restricted by high computational demands, especially in multi-query, real-time, or hardware-constrained environments. The methods proposed throughout this thesis aim to extend the reach of high-fidelity simulations into such domains—delivering their benefits without incurring the prohibitive cost.

By building on concepts from both Model Order Reduction (MOR) and Machine Learning (ML), a unified latent modeling framework is developed, where system dynamics are described in a low-dimensional latent space. Essential system behavior is first compressed into compact representations using a range of linear and nonlinear dimensionality reduction techniques. The dynamics are then approximated within this reduced space, enabling faster yet expressive simulations.

Both black-box and model discovery strategies are explored to approximate the latent dynamics. Black-box approaches offer ease of implementation and computational speed, while model discovery techniques enable interpretability, physical consistency, and extrap-

olation capabilities beyond the training domain. Notably, the proposed techniques allow for extracting governing behavior from data alone—without requiring access to internal simulation operators or source code.

Throughout the thesis, several severe challenges in surrogate modeling of structural dynamical systems are addressed. In the black-box part, various dimensionality reduction and regression strategies are benchmarked for their ability to approximate crash dynamics in a kart simulation. This is extended to a multi-hierarchical framework that combines multiple surrogates of varying resolution to capture system behavior at micro- as well as macroscale under varying hardware demands.

In the model discovery part, emphasis is placed on selecting suitable latent coordinates in which system dynamics could be described by Ordinary Differential Equations (ODEs). With the introduction of the Autoencoder-based Port-Hamiltonian Identification Network (APHIN) method, structure-preserving ROMs are learned that retain essential system-theoretic properties such as energy conservation and passivity. In a subsequent generative modeling framework—comprising Variational Encoding of Noisy Inputs (VENI), Variational Identification of Nonlinear Dynamics (VINDy), and Variational Inference with Credibility Intervals (VICI)—uncertainty and noise in the data are explicitly addressed using variational techniques. These methods enable robust surrogate models capable of handling noisy input data while remaining interpretable and physically meaningful.

Limitations Despite the potential and versatility of the proposed methods, they naturally exhibit several limitations that must be acknowledged and critically discussed.

Data Quantity and Quality As with most data-driven approaches, the quality and quantity of available data are critical factors that significantly influence the success of surrogate modeling. However, in practice, data is often subject to noise and uncertainty. While noise is typically associated with real-world experimental setups, where it stems from limitations in measurement technology, it can also be present in purely computational contexts, arising from numerical artifacts such as discretization errors, solver tolerances, or interpolation inaccuracies. This challenge became particularly evident in two key areas of this thesis.

First, in the model discovery setting, derivative information is required to identify governing dynamics. Numerical differentiation, particularly of noisy data, is notoriously ill-posed, with noise amplification increasing with each successive derivative. This challenge must explicitly be addressed as in the generative modeling framework comprising VENI, VINDy, and VICI. Second, numerical artifacts, such as discretization-induced discontinuities, posed significant challenges when creating holistic surrogates that approximate multiple physical quantities. In particular, stress fields—derived from displacement fields—lack inter-element continuity in standard Finite Element (FE) formulations. These fields are often highly non-smooth, irregularly distributed, and

significantly harder to approximate and reduce without losing critical information. This highlights a fundamental limitation in constructing multi-output surrogates from noisy or inconsistent data representations.

Hyperparameter Tuning Another practical limitation concerns hyperparameter sensitivity. Many of the proposed methods rely on careful selection of loss function weights, data scaling, and training strategies. Although general guidance and heuristics are provided throughout the thesis, optimal configurations are highly problem-dependent and often require manual tuning or the use of automated hyperparameter optimization methods.

Extrapolation Finally, while the model discovery approaches demonstrated promising extrapolation capabilities, they remain embedded in neural architectures. Consequently, generalization beyond the training regime is limited by the capacity of the autoencoder to represent out-of-distribution system states. In scenarios where the training data does not sufficiently cover the relevant solution space, latent representations may fail to meaningfully encode system dynamics, leading to poor surrogate performance.

Outlook The methods presented in this thesis lay a strong foundation for data-driven surrogate modeling of structural dynamical systems. Building on this foundation, several future research directions are envisioned to further expand the applicability, interpretability, and physical consistency of surrogate models.

Holistic Latent Surrogates Across Physical Quantities A promising extension lies in the development of holistic surrogate models that jointly approximate multiple physical quantities within a shared latent framework and provide discretization-free reconstructions. Instead of focusing on a single quantity of interest, future models could capture the coupled behavior of different physical observables using a unified latent representation and incorporate knowledge about dependencies among the quantities of interest. This may be achieved through parallel multi-channel architectures (e.g., graph-based representations) or shared-latent-space autoencoders that allow each physical field to be reconstructed on demand.

Surrogate-Based Full-Field Reconstruction from Sparse Measurements An important step toward real-time monitoring and digital twins involves reconstructing full-field dynamic behavior from sparse sensor data. Extending the presented surrogate modeling techniques, future work could explore the integration of measurement-driven latent state estimation. In this setting, sensor readings act as latent variables from which surrogate decoders reconstruct the full state of the system, even in the presence of noise, limited sampling, or unmeasured regions. This opens up the possibility of sensor-informed simulation models capable of delivering data-consistent physical states with reduced sensor counts—relevant in applications where sensor placement is constrained due to physical, financial, or privacy limitations.

Multi-Scale and Multi-Physics Surrogates Another avenue for future research is the incorporation of multi-scale effects and multi-resolution surrogate models. The multi-hierarchical framework introduced in this thesis could be extended to support multi-physics systems, where different components of a structure interact across spatial and temporal scales. Efficient surrogate coupling strategies and latent transfer learning techniques may be explored to propagate information across scales, balancing speed with physical fidelity. Such an approach is particularly valuable for systems that exhibit both global dynamics and localized nonlinearities, such as structures undergoing contact, fracture, or fatigue.

Real-World Demonstrators and Applications To validate and mature the proposed modeling concepts, future work may involve real-world demonstrators and implementation on edge-devices. Applying the methods developed in this thesis to such demonstrators will provide practical insights into system integration, real-time state estimation, and predictive maintenance.

In conclusion, this work highlights the potential of uniting physical insight with modern data-driven techniques to address long-standing challenges in computational modeling. The proposed methods pave the way for real-time, interpretable, and efficient simulation tools—greatly broadening the range of users who can benefit from valuable insights. This represents a step toward intelligent engineering systems that are not only fast but also faithful to the underlying physics.

Abbreviations, Symbols, and Notation

Blackboard Bold Latin Capitals

$\mathbb{E}$	expectation
$\mathbb{M}$	operator
$\mathbb{N}$	natural numbers
$\mathbb{P}$	set of parameter combinations
$\mathbb{R}$	real numbers
$\mathbb{T}$	set of timesteps
$\mathbb{V}$	set of planes at a node of a mesh

Calligraphic Latin Capitals

$\mathcal{A}$	adjacency matrix
$\mathcal{D}$	dataset
$\mathcal{E}$	edges in a graph
$\mathcal{F}$	class of surrogate models
$\mathcal{G}$	graph
$\mathcal{H}$	Hamiltonian
$\mathcal{I}$	stress space
$\mathcal{J}$	computational costs
$\mathcal{K}$	kernel matrix
$\mathcal{L}$	Laplace distribution
$\mathcal{M}$	mesh
$\mathcal{N}$	normal distribution
$\mathcal{O}$	order of
$\mathcal{P}$	parameter space
$\mathcal{Q}$	displacement space
$\mathcal{S}$	solution manifold
$\mathcal{T}$	time interval
$\mathcal{V}$	set of nodes/vertices of a mesh/-graph
$\mathcal{X}$	state space
$\mathcal{X}_{\mathrm{N}}$	bounded neighborhood
$\mathcal{Z}$	latent state space

Greek Capitals

$\boldsymbol{\Omega}$	right singular vectors
$\boldsymbol{\Psi}$	neural network
$\boldsymbol{\Sigma}$	singular value matrix
$\boldsymbol{\Theta}$	SINDy library of Ansatzfunctions
$\boldsymbol{\Upsilon}$	eigenvector matrix
$\boldsymbol{\Xi}$	SINDy coefficients

Greek Minuscules

$\boldsymbol{\alpha}$	inputs/features
β	beta in β-Variational Autoencoders (VAEs)
γ_{lr}	learning rate
ϵ	parameter modulating the number of selected columns in CUR decom-
	position
ε	noise factor
$\varepsilon_{\mathrm{meas}}$	measurement noise factor
$\epsilon_{\mathrm{pos,def}}$	regularization term
$\boldsymbol{\varepsilon}_{\mathrm{rt}}$	artificial noise term for reparameterization trick in VAEs
$\boldsymbol{\zeta}$	forces

$\boldsymbol{\theta}$	regression model
$\boldsymbol{\vartheta}^2$	variance
$\boldsymbol{\vartheta}$	standard deviation
ω	pendulum angle
$\boldsymbol{\kappa}$	kernel function
$\boldsymbol{\lambda}$	eigenvalues
λ	loss factor
$\acute{\boldsymbol{\mu}}$	mean function
$\boldsymbol{\mu}$	mean
$\boldsymbol{\nu}$	nodes/vertices in a mesh/graph
$\boldsymbol{\xi}$	coefficients of function terms
$\boldsymbol{\varphi}$	feature mapping
$\boldsymbol{\varpi}$	temperature
$\boldsymbol{\varrho}$	momenta
$\boldsymbol{\mu}$	parameter
$\boldsymbol{\varsigma}$	singular values
$\boldsymbol{\sigma}$	stress
τ	threshold for PDF thresholding
$\boldsymbol{\upsilon}$	eigenvectors
$\boldsymbol{\phi}_{\mathrm{e}}$	encoder
$\boldsymbol{\phi}$	reduction mapping
$\boldsymbol{\psi}_{\mathrm{d}}$	decoder
$\boldsymbol{\psi}$	reconstruction mapping
$\boldsymbol{\chi}$	spatial coordinates

Latin Capitals

$\boldsymbol{A}$	state matrix
$\boldsymbol{B}$	input matrix
$\boldsymbol{C}$	damping matrix
$\boldsymbol{D}$	degree matrix
$E_{\boldsymbol{d}}^{\mathrm{mean}}$	mean displacement error over all nodes
$E_{\boldsymbol{d}}^{\mathrm{max}}$	maximum displacement error over all nodes
$E_{\breve{\boldsymbol{x}}}$	reconstruction error on state space
$E_{\boldsymbol{x}}$	error on state space
$E_{\boldsymbol{z}}$	error on latent space
$\boldsymbol{F}$	flow of a vector field
$\tilde{\boldsymbol{F}}$	surrogate model
GP	Gaussian process
$\boldsymbol{J}$	structure matrix
$\boldsymbol{K}$	stiffness matrix
KL	Kullback-Leibler divergence
L	loss function
$\boldsymbol{M}$	mass matrix
$\boldsymbol{P}$	projector
$\boldsymbol{Q}$	energy matrix
$\boldsymbol{R}$	dissipation matrix
T	Chebyshev polynomial
$\boldsymbol{U}$	left singular vectors
$\boldsymbol{V}$	projection matrix
$\boldsymbol{\mathcal{W}}$	projection matrix
$\boldsymbol{W}$	neural network weight matrix
$\boldsymbol{X}$	state data/snapshot matrix

Other Symbols

C	matrix in CUR decomposition
$\mathscr{D}$	downsampling matrix
f	filter
$\mathscr{F}$	Fourier transform
K	covariance matrix
$\mathscr{L}$	Laplace matrix
l	pendulum length
$\mathscr{O}$	quadric
$\mathscr{P}$	family of distributions
p	position of pendulum bob
$\mathscr{Q}$	family of distributions
R	matrix in CUR decomposition
U	matrix in CUR decomposition
$\mathscr{U}$	upsampling matrix
y	output of an system

Latin Minuscules

a	activation function
$\boldsymbol{b}$	bias
c	damping
$\boldsymbol{d}$	displacements
d_{Kol}	Kolmogorov distance

d^r_{Kol}	Kolmogorov n-wdith
e	error over time
$\bar{e}$	normalized error over time
$e^{\text{mean}}_{\boldsymbol{d}}$	mean displacement error over all nodes over time
$e^{\text{max}}_{\boldsymbol{d}}$	maximum displacement error over all nodes over time
$\bar{e}_{\breve{\boldsymbol{x}}}$	normalized reconstruction error on state space over time
$\bar{e}_{\boldsymbol{x}}$	normalized error on state space over time
$\bar{e}_{\boldsymbol{z}}$	normalized error on latent space over time
$\hat{\boldsymbol{e}}_i$	standard basis vector
$\boldsymbol{f}$	right hand side of an ODE
$\boldsymbol{g}$	gate in an LSTM
g	graviational acceleration
$\boldsymbol{h}$	hidden state
i	iterator index
j	iterator index
k	iterator index
$\boldsymbol{k}$	stiffness
l	iterator index
$\boldsymbol{m}$	mass
n	state dimension
$n_{\boldsymbol{\alpha}}$	dimension of input variable in ML
n_{c}	number of channels/DOFs per node
n_{cheb}	order of Chebyshev polynomials
$n_{\downarrow}$	number of nodes in downsampled mesh
n_{e}	number of edges
n_{elem}	number of elements
n_{h}	number of hidden states
n_{l}	number of layers
n_ℓ	number of levels in multi-hierarchic modeling
n_{n}	number of neurons
n_{node}	number of nodes
n_{PCA}	intermediate state dimension
n_μ	parameter dimension
n_s	number of samples
n_{sim}	number of simulations
n_t	number of timesteps
n_s^{test}	number of test samples
n_s^{train}	number of training samples
$n_{\boldsymbol{u}}$	input dimension
n_w	time horizon
$n_{\boldsymbol{y}}$	dimension of output variable in ML
$n_{\mathbf{y}}$	output dimension
$p(\boldsymbol{z}\|\boldsymbol{x})$	posterior probability
$p(\boldsymbol{z})$	prior probability
p	probability
q	approximated posterior
r	latent state dimension
$\boldsymbol{s}$	cell state in LSTM
s^{CUR}	statistical leverage score
Δt	computational time
t	time
t_{end}	final time of a simulation
$\boldsymbol{u}$	input
$\boldsymbol{v}$	reduced basis vector
$\boldsymbol{w}$	neural network weights
$\boldsymbol{x}$	system state
$\boldsymbol{y}$	outputs/target values
$\boldsymbol{z}$	latent state

Notation Unless stated otherwise, the following notational conventions are used throughout this thesis:

- Scalars are denoted by lowercase letters in regular font, e.g. a
- Vectors are denoted by bold lowercase letters, e.g. $\boldsymbol{a}$.

- Matrices are denoted by bold uppercase letters, e.g. $\boldsymbol{A}$

In general, subscripts using lowercase Latin letters or Arabic numerals (e.g., $\boldsymbol{x}_i$, $\boldsymbol{x}_1$) are used to denote different instances or samples of the same quantity, with the exception of $\boldsymbol{x}_0$, which is reserved for initial values. In cases where such indexing would conflict with semantic distinctions—such as in systems of ODEs where x_1, x_2 might refer to distinct state variables—we instead use Roman numerals (e.g. x_{I}, x_{II}) to indicate structurally different quantities.

If a vector is defined as $\boldsymbol{a} = [a_1, \ldots, a_n]$, then the notation $[\boldsymbol{a}]_i = a_i$ denotes its i-th entry. Similarly, for a matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]$, the i-th column is denoted by $[\boldsymbol{A}]_i = \boldsymbol{a}_i$. An individual matrix entry at row i and column j is written as $[\boldsymbol{A}]_{ij} = [\boldsymbol{a}_i]_j = a_{ij}$.

Additional symbols and operators used throughout this thesis are listed in Table 9.1.

Table 9.1: Common symbols and operators used throughout this thesis.

Symbol	**Meaning**
$\mathbb{R}$	The set of real numbers
$\boldsymbol{a}^\intercal$, $\boldsymbol{A}^\intercal$	Transpose of matrix or vector
$\boldsymbol{A}^{-1}$	Inverse of a matrix
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{A}^\dagger$	Moore–Penrose pseudoinverse of a matrix
$\|\boldsymbol{a}\|$	Norm of a vector or matrix
$\mathrm{diag}(\boldsymbol{a})$	Diagonal matrix with entries of $\boldsymbol{a}$ on its diagonal
$\odot$	Hadamard (element-wise) product
$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{a} = \dot{\boldsymbol{a}}$	Total time derivative
$\tilde{a}$	Approximate or estimated value
$\breve{a}$	Reconstructed quantity
$\bar{a}$	Mean value of a variable
$[\boldsymbol{a}]_i$, $[\boldsymbol{A}]_{ij}$	Entry of vector or matrix
a_i	i-th instance of a quantity

Remarks Unless stated otherwise, all results presented in this thesis—except for the finite element simulations—were generated on an Apple M1 Max system with a 10-core CPU, 24-core GPU, and 64 GB of RAM.

Statement on the Use of Language Models and Digital Tools During the preparation of this thesis, I made use of modern language models (LLMs) and neural machine translation service, such as ChatGPT and DeepL, to support various aspects

of the writing process. These tools were employed exclusively for non-technical tasks, including:

- Refinement of language and grammar
- Structural and stylistic improvements of academic text
- Translation assistance (primarily between German and English)
- Clarification and rephrasing of existing content to improve readability

All technical content, scientific reasoning, original results, and interpretations were independently developed and verified by the author. The use of such tools was limited to enhancing the clarity, coherence, and presentation quality of the text, without influencing the scientific rigor or originality of the work.

Acronyms

AE Autoencoder 49–52, 54–56, 73, 74, 81, 84, 88, 89, 100, 111, 128, 138, 143

AI Artificial Intelligence 26

APHIN Autoencoder-based Port-Hamiltonian Identification Network 128, 129, 133, 141, 142, 144, 146–148, 150–153, 186

CAD Computer-Aided Design 16

CMS Component Mode Synthesis 40

CNN Convolutional Neural Network 27, 100

CUR CUR Decomposition 51, 52, 54, 55, 73–75, 77, 83–85, 87, 89

DAE Differential-Algebraic Equation 146

DL Deep Learning 27, 35

DMD Dynamic Mode Decomposition 26, 59

DOF Degree of Freedom 23, 31, 38, 148, 153, 176

ELBO Evidence Lower-Bound 158–160, 165

FCAE Fully-Connected Autoencoder 111–113

FE Finite Element 3, 15–20, 22, 23, 31, 34, 38, 39, 41, 46, 76, 80, 88, 89, 100, 105, 110, 120, 148–151, 186

FEM Finite Element Method 12, 14–16, 22, 38, 39, 179, 180

FOM Full Order Model 14, 133, 179

GAE Graph Convolutional Autoencoder 111–114, 117

GCNN Graph Convolutional Neural Network 7, 98, 100, 101, 103, 108, 120, 183

GP Gaussian Process 27, 30, 58, 61, 75, 81, 82, 84, 87, 88, 90, 126

HF High-Fidelity 2–4, 6, 11, 24, 31, 35, 42, 51, 57, 72, 78, 88, 92, 94, 98, 124, 125, 148, XI, XIII

KL Kullback-Leibler 158, 159, 162, 167

KPCA Kernel Principal Component Analysis 48, 49, 51, 52, 54, 73–75, 77, 81, 84, 85, 87–89

LADy Latent Approximation of Dynamics 5, 73–75, 124

LDD Latent Discovery of Dynamics 5, 73, 124, 125, 160

LSTM Long Short-Term Memory 27, 62–64, 67, 68, 90–92, 95

LTI Linear Time-Invariant 127, 130, 134, 140, 144, 145

MB Multi-Body 16, 21, 22

MBS Multi-Body System 16

MEM Micro-Electromechanical System 73, 179, 180, 182, 183

MH Multi-Hierarchical 7, 98, 99, 106, 107, 109–121
ML Machine Learning 1, 2, 4–8, 26, 27, 33–35, 185, XIII
MLP Multi Layer Perceptron 27, 67, 75, 81, 98, 106, 108, 110–112, 118
MOR Model Order Reduction 3, 4, 6, 7, 19, 25, 26, 37, 40, 41, 90, 91, 100, 101, 132, 133, 185, XIII
MSD Mass-Spring-Damper 127, 142, 145
MSE Mean Squared Error 29, 67

NN Neural Network 27, 28, 58, 61, 75, 81, 82, 88, 90, 103, 111, 112

ODE Ordinary Differential Equation 11, 14, 57–59, 64, 66–69, 73, 101, 125, 126, 132, 146, 163, 170, 171, 181, 186
OI Operator Inference 26

PCA Principal Component Analysis 41, 43, 48, 51, 52, 54, 56, 73–75, 77, 81, 84, 85, 87–93, 95, 111–113, 115, 117, 119, 133, 139, 143, 150, 177, 181
PDE Partial Differential Equation 11, 12, 14, 33, 37, 57–59, 73, 99, 101, 155, 156, 176, 179
PDF Probablity Density Function 169, 173, 174, 177, 183
PH Port-Hamiltonian 7, 124, 128–134, 136–148, 151, 153
PHIN Port-Hamiltonian Identification Network 128, 133, 136–138, 141–146, 148, 151
PINN Physics-Informed Neural Network 58
POD Proper Orthogonal Decomposition 43

RNN Recurrent Neural Network 27, 62, 63, 75, 91
ROM Reduced Order Model 7, 8, 19, 26, 40, 73, 83, 117, 124, 125, 149, 151, 155, 156, 160, 161, 181, 182, 185, 186

SINDy Sparse Identification of Nonlinear Dynamics 8, 26, 59, 64–68, 126, 156, 160, 163
STLSQ Sequentially Thresholded Least Squares 65, 67
SVD Singular Value Decomposition 43, 44

UQ Uncertainty Quantification 35, 124, 155, 156, 160, 171, 174, 181–183

VAE Variational Autoencoder 50, 51, 54, 157–164, 166, 170, 183, 189
VENI Variational Encoding of Noisy Inputs 156, 157, 160–162, 165, 167–169, 172, 173, 178, 181, 186
VI Variational Inference 30
VICI Variational Inference with Credibility Intervals 156, 157, 160, 161, 170, 171, 181, 183, 186
VINDy Variational Identification of Nonlinear Dynamics 156, 157, 160, 161, 163–169, 172–174, 177, 178, 181–183, 186

Bibliography

[AbdarEtAl21] Abdar, M.; Pourpanah, F.; Hussain, S.; Rezazadegan, D.; Liu, L.; Ghavamzadeh, M.; Fieguth, P.; Cao, X.; Khosravi, A.; Acharya, U.R.; Makarenkov, V.; Nahavandi, S.: A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion, Vol. 76, pp. 243–297, 2021.

[Aggarwal23] Aggarwal, C.C.: Neural Networks and Deep Learning: A Textbook. Springer International Publishing, 2023.

[AlemiEtAl16] Alemi, A.A.; Fischer, I.; Dillon, J.V.; Murphy, K.: Deep variational information bottleneck. arXiv preprint arXiv:1612.00410, 2016.

[AlonYahav20] Alon, U.; Yahav, E.: On the Bottleneck of Graph Neural Networks and its Practical Implications. arXiv, 2020.

[Antoulas05] Antoulas, A.C.: Approximation of large-scale dynamical systems. No. 6 in Advances in Design and Control. Philadelphia, Pa.: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2005. Restricted to subscribers or individual electronic text purchasers.

[ArioliGazzola15] Arioli, G.; Gazzola, F.: A new mathematical explanation of what triggered the catastrophic torsional mode of the Tacoma Narrows Bridge. Applied Mathematical Modelling, Vol. 39, No. 2, pp. 901–912, 2015.

[BabuskaSzabo82] Babuska, I.; Szabo, B.: On the rates of convergence of the finite element method. International Journal for Numerical Methods in Engineering, Vol. 18, No. 3, pp. 323–341, 1982.

[BabuškaGuo92] Babuška, I.; Guo, B.: The h, p and h-p version of the finite element method; basis theory and applications. Advances in Engineering Software, Vol. 15, No. 3–4, pp. 159–174, 1992.

[Bai02] Bai, Z.: Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. Applied Numerical Mathematics, Vol. 43, No. 1–2, pp. 9–44, 2002.

[BakarjiEtAl23] Bakarji, J.; Champion, K.; Nathan Kutz, J.; Brunton, S.L.: Discovering governing equations from partial measurements with deep delay autoencoders. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 479, No. 2276, 2023.

[BakırWestonSchölkopf04] Bakır, G.H.; Weston, J.; Schölkopf, B.: Learning to find pre-images. Advances in neural information processing systems, Vol. 16, pp. 449–456, 2004.

[BarraultEtAl04] Barrault, M.; Maday, Y.; Nguyen, N.C.; Patera, A.T.: An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. Comptes Rendus. Mathématique, Vol. 339, No. 9, pp. 667–672, 2004.

[BassEtAl99] Bass, C.R.; Crandall, J.R.; Bolton, J.R.; Pilkey, W.D.; Khaewpong, N.; Sun, E.: Deployment of Air Bags into the Thorax of an Out-of-Position Dummy. SAE Transactions, Vol. 108, pp. 1468–1482, 1999.

[Bathe14] Bathe, K.J. (Ed.): Finite element procedures. Watertown, MA: K.J. Bathe, 2nd Edn., 2014.

[BattagliaEtAl18] Battaglia, P.W.; Hamrick, J.B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gulcehre, C.; Song, F.; Ballard, A.; Gilmer, J.; Dahl, G.; Vaswani, A.; Allen, K.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; Pascanu, R.: Relational inductive biases, deep learning, and graph networks. arXiv, 2018.

[BeattieGugercin05] Beattie, C.; Gugercin, S.: Krylov-based model reduction of second-order systems with proportional damping. In Proceedings of the 44th IEEE Conference on Decision and Control, pp. 2278–2283, 2005.

[BeattieGugercin11] Beattie, C.; Gugercin, S.: Structure-preserving Model Reduction for Nonlinear Port-Hamiltonian Systems. Proceedings of the IEEE Conference on Decision and Control, pp. 6564–6569, 2011.

[BeattieGugercinMehrmann22] Beattie, C.; Gugercin, S.; Mehrmann, V.: Structure-preserving interpolatory model reduction for port-Hamiltonian differential-algebraic systems. In Beattie, C.; Benner, P.; Embree, M.; Gugercin, S.; Lefteriu, S. (Eds.): Realization and Model Reduction of Dynamical Systems: A Festschrift in Honor of the 70th Birthday of Thanos Antoulas, pp. 235–254. Cham: Springer International Publishing, 2022.

[BechmannSchmid24] Bechmann, R.; Schmid, A.: Challenges in structural design and execution: Stuttgart's new central station. Structural Concrete, Vol. 25, No. 3, pp. 1508–1527, 2024.

[BeckEtAl24] Beck, M.; Pöppel, K.; Spanring, M.; Auer, A.; Prudnikova, O.; Kopp, M.; Klambauer, G.; Brandstetter, J.; Hochreiter, S.: xLSTM: Extended Long Short-Term Memory. arXiv, 2024.

[BelytschkoEtAl14] Belytschko, T.; Liu, W.K.; Moran, B.; Elkhodary, K.: Nonlinear Finite Elements for Continua and Structures. New York Academy of Sciences Series. Newark: John Wiley & Sons, Incorporated, 1st Edn., 2014. Description based on publisher supplied metadata and other sources.

[Benner17] Benner, P.; Cohen, A.; Ohlberger, M.; Willcox, K. (Eds.): Model reduction and approximation. No. 15 in Computational science and engineering. Philadelphia: SIAM, Society for Industrial and Applied Mathematics, [2017], 2017. Includes bibliographical references and index.

[BennerEtAl21] Benner, P.; Schilders, W.; Grivet-Talocia, S.; Quarteroni, A.; Rozza, G.; Miguel Silveira, L.: Model Order Reduction: Volume 3 Applications. Berlin: De Gruyter, 2021.

[BerkoozHolmesLumley93] Berkooz, G.; Holmes, P.; Lumley, J.L.: The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. Annual Review of Fluid Mechanics, Vol. 25, No. 1, pp. 539–575, 1993.

[BesselinkEtAl13] Besselink, B.; Tabak, U.; Lutowska, A.; van de Wouw, N.; Nijmeijer, H.; Rixen, D.; Hochstenbach, M.; Schilders, W.: A comparison of model reduction techniques from structural dynamics, numerical mathematics and systems and control. Journal of Sound and Vibration, Vol. 332, No. 19, pp. 4403–4422, 2013.

[Bishop19] Bishop, C.M.: Pattern recognition and machine learning. Information Science and Statistics. New York, NY: Springer Science+Business Media, LLC, 2019.

[BongardLipson07] Bongard, J.; Lipson, H.: Automated reverse engineering of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, Vol. 104, No. 24, pp. 9943–9948, 2007.

[BonnevilleEtAl23] Bonneville, C.; Choi, Y.; Ghosh, D.; Belof, J.L.: GPLaSDI: Gaussian Process-based Interpretable Latent Space Dynamics Identification through Deep Autoencoder. Computer Methods in Applied Mechanics and Engineering, 418A, 116535, 2024, Vol. 418, p. 116535, 2023.

[BorjaScherpenFujimoto23] Borja, P.; Scherpen, J.M.A.; Fujimoto, K.: Extended balancing of continuous LTI systems: A structure-preserving approach. IEEE Transactions on Automatic Control, Vol. 68, No. 1, pp. 257–271, 2023.

[BotteghiGuoBrune22] Botteghi, N.; Guo, M.; Brune, C.: Deep kernel learning of dynamical models from high-dimensional noisy data. Scientific reports, Vol. 12, No. 1, p. 21530, 2022.

[BreimanEtAl17] Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J.: Classification And Regression Trees. Routledge, 2017.

[BreitenMorandinSchulze22] Breiten, T.; Morandin, R.; Schulze, P.: Error bounds for port-Hamiltonian model and controller reduction based on system balancing. Computers & Mathematics with Applications, Vol. 116, pp. 100–115, 2022.

[BreitenUnger22] Breiten, T.; Unger, B.: Passivity preserving model reduction via spectral factorization. Automatica, Vol. 142, p. 110368, 2022.

[BronsteinEtAl21] Bronstein, M.M.; Bruna, J.; Cohen, T.; Veličković, P.: Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. arXiv, 2021.

[BruntonEtAl22] Brunton, S.L.; Budišić, M.; Kaiser, E.; Kutz, J.N.: Modern Koopman Theory for Dynamical Systems. SIAM Review, Vol. 64, No. 2, pp. 229–340, 2022.

[BruntonKutz22] Brunton, S.L.; Kutz, J.N.: Data-driven science and engineering: machine learning, dynamical systems, and control. Cambridge: Cambridge University Press, 2nd Edn., 2022.

[BruntonKutz23] Brunton, S.L.; Kutz, J.N.: Machine Learning for Partial Differential Equations, 2023.

[BruntonProctorKutz16] Brunton, S.L.; Proctor, J.L.; Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, Vol. 113, No. 15, pp. 3932–3937, 2016.

[BryutkinEtAl24] Bryutkin, A.; Huang, J.; Deng, Z.; Yang, G.; Schönlieb, C.B.; Aviles-Rivero, A.: HAMLET: Graph Transformer Neural Operator for Partial Differential Equations. arXiv, 2024.

[BuchfinkEtAl24] Buchfink, P.; Glas, S.; Haasdonk, B.; Unger, B.: Model reduction on manifolds: A differential geometric framework. Physica D: Nonlinear Phenomena, Vol. 468, p. 134299, 2024.

[BuchfinkGlasHaasdonk23] Buchfink, P.; Glas, S.; Haasdonk, B.: Approximation bounds for model reduction on polynomially mapped manifolds. arXiv, 2023.

[BuiThanhDamodaranWillcox03] Bui-Thanh, T.; Damodaran, M.; Willcox, K.: Proper Orthogonal Decomposition Extensions for Parametric Applications in Compressible Aerodynamics. In 21st AIAA Applied Aerodynamics Conference, American Institute of Aeronautics and Astronautics, 2003.

[CastañosEtAl13] Castaños, F.; Gromov, D.; Hayward, V.; Michalska, H.: Implicit and explicit representations of continuous-time port-Hamiltonian systems. Systems & Control Letters, Vol. 62, No. 4, pp. 324–330, 2013.

[ChamonRibeiro20] Chamon, L.; Ribeiro, A.: Probably Approximately Correct Constrained Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; Lin, H. (Eds.): Advances in Neural Information Processing Systems, Vol. 33, pp. 16722–16735, Curran Associates, Inc., 2020.

[ChampionEtAl19] Champion, K.; Lusch, B.; Kutz, J.N.; Brunton, S.L.: Data-driven discovery of coordinates and governing equations. Proceedings of the National Academy of Sciences, Vol. 116, No. 45, pp. 22445–22451, 2019.

[ChangEtAl07] Chang, J.M.; Tyan, T.; El-bkaily, M.; Cheng, J.; Marpu, A.; Zeng, Q.; Santini, J.: Implicit and Explicit Finite Element Methods for Crash Safety Analysis. In SAE Technical Paper Series, ANNUAL, SAE International, 2007.

[ChaturantabutBeattieGugercin16] Chaturantabut, S.; Beattie, C.; Gugercin, S.: Structure-preserving model reduction for nonlinear port-Hamiltonian systems. SIAM Journal on Scientific Computing, Vol. 38, No. 5, pp. B837–B865, 2016.

[ChaturantabutSorensen10] Chaturantabut, S.; Sorensen, D.C.: Nonlinear Model Reduction via Discrete Empirical Interpolation. SIAM Journal on Scientific Computing, Vol. 32, No. 5, pp. 2737–2764, 2010.

[ChenEtAl18] Chen, R.T.Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K.: Neural Ordinary Differential Equations. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; Garnett, R. (Eds.): Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018.

[ChenEtAl20a] Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; Sun, X.: Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 04, pp. 3438–3445, 2020.

[ChenEtAl20b] Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; Li, Y.: Simple and Deep Graph Convolutional Networks. In III, H.D.; Singh, A. (Eds.): Proceedings of the 37th International Conference on Machine Learning, Vol. 119 of Proceedings of Machine Learning Research, pp. 1725–1735, PMLR, 2020.

[ChenEtAl22a] Chen, B.; Huang, K.; Raghupathi, S.; Chandratreya, I.; Du, Q.; Lipson, H.: Automated discovery of fundamental variables hidden in experimental data. Nature Computational Science, Vol. 2, No. 7, pp. 433–442, 2022.

[ChenEtAl22b] Chen, P.Y.; Xiang, J.; Cho, D.H.; Chang, Y.; Pershing, G.A.; Maia, H.T.; Chiaramonte, M.M.; Carlberg, K.; Grinspun, E.: CROM: Continuous Reduced-Order Modeling of PDEs Using Implicit Neural Representations. arXiv, 2022.

[Cherifi20] Cherifi, K.: An overview on recent machine learning techniques for Port Hamiltonian systems. Physica D: Nonlinear Phenomena, Vol. 411, p. 132620, 2020.

[CherifiGoyalBenner22] Cherifi, K.; Goyal, P.; Benner, P.: A non-intrusive method to inferring linear port-Hamiltonian realizations using time-domain data. Electronic Transactions on Numerical Analysis : Special Issue SciML, Vol. 56, pp. 102 – 116, 2022.

[ChinestaEtAl20] Chinesta, F.; Cueto, E.; Abisset-Chavanne, E.; Duval, J.L.; Khaldi, F.E.: Virtual, Digital and Hybrid Twins: A New Paradigm in Data-Based Engineering and Engineered Data. Archives of Computational Methods in Engineering, Vol. 27, No. 1, pp. 105–134, 2020.

[ContiEtAl23a] Conti, P.; Gobat, G.; Fresca, S.; Manzoni, A.; Frangi, A.: Reduced order modeling of parametrized systems through autoencoders and SINDy approach: continuation of periodic solutions. Computer Methods in Applied Mechanics and Engineering, Vol. 411, p. 116072, 2023.

[ContiEtAl23b] Conti, P.; Guo, M.; Manzoni, A.; Frangi, A.; Brunton, S.L.; Kutz, J.N.: Multi-fidelity reduced-order surrogate modeling. arXiv, 2023.

[ContiEtAl24] Conti, P.; Kneifl, J.; Manzoni, A.; Frangi, A.; Fehr, J.; Brunton, S.L.; Kutz, J.N.: VENI, VINDy, VICI: a variational reduced-order modeling framework with uncertainty quantification. arXiv, 2024.

[Corigliano18] Corigliano, A.: Mechanics of microsystems. No. 7646 in Microsystem and nanotechnology series. Hoboken: Wiley, 2018.

[CoriglianoEtAl04] Corigliano, A.; DeMasi, B.; Frangi, A.; Comi, C.; Villa, A.; Marchi, M.: Mechanical Characterization of Polysilicon Through On-Chip Tensile Tests. Journal of Microelectromechanical Systems, Vol. 13, No. 2, pp. 200–219, 2004.

[Craig00] Craig, R. Jr: Coupling of substructures for dynamic analyses-an overview. In 41st structures, structural dynamics, and materials conference and exhibit, p. 1573, 2000.

[CraigBampton68] Craig, R.R.; Bampton, M.C.C.: Coupling of substructures for dynamic analyses. AIAA Journal, Vol. 6, No. 7, pp. 1313–1319, 1968.

[CraigKurdila06] Craig, R.R.; Kurdila, A.: Fundamentals of structural dynamics. Hoboken, NJ: Wiley, 2nd Edn., 2006. Includes bibliographical references and index. - Previous ed.: published as Structural dynamics. New York: Chichester: Wiley, 1981.

[CranmerEtAl19] Cranmer, M.D.; Xu, R.; Battaglia, P.; Ho, S.: Learning Symbolic Physics with Graph Networks. arXiv, 2019.

[CranmerEtAl20] Cranmer, M.; Sanchez Gonzalez, A.; Battaglia, P.; Xu, R.; Cranmer, K.; Spergel, D.; Ho, S.: Discovering Symbolic Models from Deep Learning with Inductive Biases. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; Lin, H. (Eds.): Advances in Neural Information Processing Systems, Vol. 33, pp. 17429–17442, Curran Associates, Inc., 2020.

[DalSantoDeparisPegolotti20] Dal Santo, N.; Deparis, S.; Pegolotti, L.: Data driven approximation of parametrized PDEs by reduced basis and neural networks. Journal of Computational Physics, Vol. 416, p. 109550, 2020.

[DefferrardBressonVandergheynst16] Defferrard, M.; Bresson, X.; Vandergheynst, P.: Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; Garnett, R. (Eds.): Advances in Neural Information Processing Systems, Vol. 29, Curran Associates, Inc., 2016.

[DemoTezzeleRozza23] Demo, N.; Tezzele, M.; Rozza, G.: A DeepONet multi-fidelity approach for residual learning in reduced order modeling. Advanced Modeling and Simulation in Engineering Sciences, Vol. 10, No. 1, 2023.

[DesaiEtAl21] Desai, S.A.; Mattheakis, M.; Sondak, D.; Protopapas, P.; Roberts, S.J.: Port-Hamiltonian neural networks for learning explicit time-dependent dynamical systems. Phys. Rev. E, Vol. 104, p. 034312, 2021.

[DevlinEtAl18] Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

[DuindamEtAl09] Duindam, V.; Macchelli, A.; Stramigioli, S.; Bruyninckx, H.: Modeling and control of complex physical systems: The port-Hamiltonian approach. Berlin, Heidelberg: Springer Berlin, Heidelberg, 2009.

[EberhardSchiehlen06] Eberhard, P.; Schiehlen, W.: Computational Dynamics of Multibody Systems: History, Formalisms, and Applications. Journal of Computational and Nonlinear Dynamics, Vol. 1, No. 1, pp. 3–12, 2006.

[EggerEtAl18] Egger, H.; Kugler, T.; Liljegren-Sailer, B.; Marheineke, N.; Mehrmann, V.: On structure-preserving model reduction for damped wave propagation in transport networks. SIAM Journal on Scientific Computing, Vol. 40, No. 1, pp. A331–A365, 2018.

[EidnesEtAl23] Eidnes, S.; Stasik, A.J.; Sterud, C.; Bøhn, E.; Riemer-Sørensen, S.: Pseudo-Hamiltonian neural networks with state-dependent external forces. Physica D: Nonlinear Phenomena, Vol. 446, p. 133673, 2023.

[Einstein15] Einstein, A.: Die feldgleichungen der gravitation. Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften, pp. 844–847, 1915.

[EuroNCAP23] Euro NCAP: European New Car Assessment Programme (Euro NCAP): ASSESSMENT PROTOCOL – SAFETY ASSIST COLLISION AVOIDANCE, Version 10.4.1. Tech. rep., Euro NCAP, 2 Place du Luxembourg, 1050 Brussels, Belgium, 2023.

[FarhatChapmanAvery15] Farhat, C.; Chapman, T.; Avery, P.: Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models. International Journal for Numerical Methods in Engineering, Vol. 102, No. 5, pp. 1077–1110, 2015.

[FarhatEtAl14] Farhat, C.; Avery, P.; Chapman, T.; Cortial, J.: Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. International Journal for Numerical Methods in Engineering, Vol. 98, No. 9, pp. 625–662, 2014.

[FaselEtAl22] Fasel, U.; Kutz, J.N.; Brunton, B.W.; Brunton, S.L.: Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 478, No. 2260, 2022.

[FeffermanMitterNarayanan16] Fefferman, C.; Mitter, S.; Narayanan, H.: Testing the manifold hypothesis. Journal of the American Mathematical Society, Vol. 29, No. 4, pp. 983–1049, 2016.

[FehrHolzwarthEberhard16] Fehr, J.; Holzwarth, P.; Eberhard, P.: Interface and model reduction for efficient explicit simulations - a case study with nonlinear vehicle crash models. Mathematical and Computer Modelling of Dynamical Systems, Vol. 22, No. 4, pp. 380–396, 2016.

[FeldmannFreund95] Feldmann, P.; Freund, R.: Efficient linear circuit analysis by Pade approximation via the Lanczos process. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 14, No. 5, pp. 639–649, 1995.

[Feller07] Feller, J. (Ed.): Perspectives on free and open source software. Cambridge, Massachusetts: MIT Press, 2007. Includes bibliographical references and index. - Title from title screen. - Description based on PDF viewed 12/23/2015.

[FengEtAl23] Feng, L.; Lombardi, L.; Antonini, G.; Benner, P.: Multi-fidelity error estimation accelerates greedy model reduction of complex dynamical systems. International Journal for Numerical Methods in Engineering, Vol. 124, No. 23, pp. 5312–5333, 2023.

[FloryanGraham22] Floryan, D.; Graham, M.D.: Data-driven discovery of intrinsic dynamics. Nature Machine Intelligence, Vol. 4, No. 12, pp. 1113–1120, 2022.

[FortunatoEtAl22] Fortunato, M.; Pfaff, T.; Wirnsberger, P.; Pritzel, A.; Battaglia, P.: MultiScale MeshGraphNets. 2nd AI4Science Workshop at the 39th International Conference on Machine Learning (ICML), 2022.

[FrancoEtAl23] Franco, N.R.; Fresca, S.; Tombari, F.; Manzoni, A.: Deep Learning-based surrogate models for parametrized PDEs: handling geometric variability through graph neural networks. arXiv, 2023.

[FrangiGobat19] Frangi, A.; Gobat, G.: Reduced order modelling of the non-linear stiffness in MEMS resonators. International Journal of Non-Linear Mechanics, Vol. 116, pp. 211–218, 2019.

[FrescaDedeManzoni21] Fresca, S.; Dede, L.; Manzoni, A.: A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. J. Sci. Comput., Vol. 87, No. 2, 2021.

[FrescaEtAl22] Fresca, S.; Gobat, G.; Fedeli, P.; Frangi, A.; Manzoni, A.: Deep learning-based reduced order models for the real-time simulation of the nonlinear dynamics of microstructures. International Journal for Numerical Methods in Engineering, Vol. 123, No. 20, pp. 4749–4777, 2022.

[FrescaManzoni22] Fresca, S.; Manzoni, A.: POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. Computer Methods in Applied Mechanics and Engineering, Vol. 388, p. 114181, 2022.

[FriesHeChoi22] Fries, W.D.; He, X.; Choi, Y.: LaSDI: Parametric Latent Space Dynamics Identification. Computer Methods in Applied Mechanics and Engineering, Vol. 399, p. 115436, 2022.

[GaoJi21] Gao, H.; Ji, S.: Graph U-Nets. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2021.

[GaoSunWang21] Gao, H.; Sun, L.; Wang, J.X.: PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. Journal of Computational Physics, Vol. 428, p. 110079, 2021.

[GarcíaGonzálezEtAl20] García-González, A.; Huerta, A.; Zlotnik, S.; Díez, P.: A kernel Principal Component Analysis (kPCA) digest with a new backward mapping (pre-image reconstruction) strategy. arXiv, 2020.

[GarlandHeckbert97] Garland, M.; Heckbert, P.S.: Surface simplification using quadric error metrics. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH 7, ACM Press, 1997.

[GillisMehrmannSharma18] Gillis, N.; Mehrmann, V.; Sharma, P.: Computing the nearest stable matrix pairs. Numerical Linear Algebra with Applications, Vol. 25, No. 5, p. e2153, 2018.

[GillisSharma17] Gillis, N.; Sharma, P.: On computing the distance to stability for matrices using linear dissipative Hamiltonian systems. Automatica, Vol. 85, pp. 113–121, 2017.

[GorbanTyukin18] Gorban, A.N.; Tyukin, I.Y.: Blessing of dimensionality: mathematical foundations of the statistical physics of data. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 376, No. 2118, p. 20170237, 2018.

[GoyalPontesDuffBenner23] Goyal, P.; Pontes Duff, I.; Benner, P.: Guaranteed stable quadratic models and their applications in SINDy and operator inference. arXiv Preprint, 2023.

[GrattarolaEtAl21] Grattarola, D.; Zambon, D.; Bianchi, F.M.; Alippi, C.: Understanding Pooling in Graph Neural Networks. arXiv, 2021.

[GreplPatera05] Grepl, M.A.; Patera, A.T.: A posteriorierror bounds for reduced-basis approximations of parametrized parabolic partial differential equations. ESAIM: Mathematical Modelling and Numerical Analysis, Vol. 39, No. 1, pp. 157–181, 2005.

[GreydanusDzambaYosinski19] Greydanus, S.; Dzamba, M.; Yosinski, J.: Hamiltonian Neural Networks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; Garnett, R. (Eds.): Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.

[Grimm56] Grimm, J.: Kinder-und hausmärchen, Vol. 3. Reclam, 1856.

[GruberEtAl22] Gruber, A.; Gunzburger, M.; Ju, L.; Wang, Z.: A comparison of neural network architectures for data-driven reduced-order modeling. Computer Methods in Applied Mechanics and Engineering, Vol. 393, p. 114764, 2022.

[GrunertFehrHaasdonk20] Grunert, D.; Fehr, J.; Haasdonk, B.: Well-scaled, a-posteriori error estimation for model order reduction of large second-order mechanical systems. ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 100, No. 8, 2020.

[Gugercin08] Gugercin, S.: An iterative SVD-Krylov based method for model reduction of large-scale dynamical systems. Linear Algebra and its Applications, Vol. 428, No. 8–9, pp. 1964–1986, 2008.

[GugercinAntoulas04] Gugercin, S.; Antoulas, A.C.: A Survey of Model Reduction by Balanced Truncation and Some New Results. International Journal of Control, Vol. 77, No. 8, pp. 748–766, 2004.

[GugercinEtAl12] Gugercin, S.; Polyuga, R.V.; Beattie, C.; van der Schaft, A.: Structure-preserving tangential interpolation for model reduction of port-Hamiltonian systems. Automatica, Vol. 48, No. 9, pp. 1963–1974, 2012.

[GuiverOpmeer13] Guiver, C.; Opmeer, M.R.: Error bounds in the gap metric for dissipative balanced approximations. Linear Algebra and its Applications, Vol. 439, No. 12, pp. 3659–3698, 2013.

[GuoHesthaven18] Guo, M.; Hesthaven, J.S.: Reduced order modeling for nonlinear structural analysis using Gaussian process regression. Computer Methods in Applied Mechanics and Engineering, Vol. 341, pp. 807–826, 2018.

[GuoHesthaven19] Guo, M.; Hesthaven, J.S.: Data-driven reduced order modeling for time-dependent problems. Computer Methods in Applied Mechanics and Engineering, Vol. 345, pp. 75–99, 2019.

[HallquistEtAl06] Hallquist, J.O.; et al.: LS–DYNA Theory Manual. Livermore software Technology corporation, Vol. 3, pp. 25–31, 2006.

[HanEtAl22] Han, X.; Gao, H.; Pfaff, T.; Wang, J.X.; Liu, L.P.: Predicting Physics in Mesh-reduced Space with Temporal Attention. arXiv, 2022.

[HartmannAuweraer21] Hartmann, D.; Van der Auweraer, H.: Digital Twins. In Cruz, M.; Parés, C.; Quintela, P. (Eds.): Progress in Industrial Mathematics: Success Stories, pp. 3–17, Cham: Springer International Publishing, 2021.

[HastieTibshiraniFriedman17] Hastie, T.; Tibshirani, R.; Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer series in statistics. New York, NY: Springer, 2nd Edn., 2017.

[Hay22] Hay, J.: A Surrogate Model-enhanced Simulation Framework for Safety Performance Assessment of Integrated Vehicle Safety Systems, Vol. 2022,75 of Schriften

aus dem Institut für Technische und Numerische Mechanik der Universität Stuttgart. Düren: Shaker Verlag, 2022.

[HesthavenUbbiali18] Hesthaven, J.; Ubbiali, S.: Non-intrusive reduced order modeling of nonlinear problems using neural networks. Journal of Computational Physics, Vol. 363, pp. 55–78, 2018.

[HeXu19] He, J.; Xu, J.: MgNet: A unified framework of multigrid and convolutional neural network. Science China Mathematics, Vol. 62, No. 7, pp. 1331–1354, 2019.

[HigginsEtAl17] Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; Lerchner, A.: beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In International Conference on Learning Representations, 2017.

[HigginsEtAl18] Higgins, I.; Amos, D.; Pfau, D.; Racaniere, S.; Matthey, L.; Rezende, D.; Lerchner, A.: Towards a definition of disentangled representations. arXiv preprint arXiv:1812.02230, 2018.

[HintonZemel93] Hinton, G.E.; Zemel, R.: Autoencoders, Minimum Description Length and Helmholtz Free Energy. In Cowan, J.; Tesauro, G.; Alspector, J. (Eds.): Advances in Neural Information Processing Systems, Vol. 6, Morgan-Kaufmann, 1993.

[HirshBarajasSolanoKutz22] Hirsh, S.M.; Barajas-Solano, D.A.; Kutz, J.N.: Sparsifying priors for Bayesian uncertainty quantification in model discovery. Royal Society Open Science, Vol. 9, No. 2, p. 211823, 2022.

[HochreiterSchmidhuber97] Hochreiter, S.; Schmidhuber, J.: Long Short-Term Memory. Neural Computation, Vol. 9, No. 8, pp. 1735–1780, 1997.

[HornikStinchcombeWhite89] Hornik, K.; Stinchcombe, M.; White, H.: Multilayer feedforward networks are universal approximators. Neural Networks, Vol. 2, No. 5, pp. 359–366, 1989.

[Hotelling33] Hotelling, H.: Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, Vol. 24, No. 6, pp. 417–441, 1933.

[Hughes00] Hughes, T.J.R. (Ed.): The finite element method. Mineola, NY: Dover Publications, 2000.

[Hurty65] Hurty, W.C.: Dynamic analysis of structural systems using component modes. AIAA Journal, Vol. 3, No. 4, pp. 678–685, 1965.

[IonescuAstolfi13] Ionescu, T.C.; Astolfi, A.: Families of moment matching based, structure preserving approximations for linear port Hamiltonian systems. Automatica, Vol. 49, No. 8, pp. 2424–2434, 2013.

[JainHaghighatNelaturi24] Jain, A.; Haghighat, E.; Nelaturi, S.: LatticeGraphNet: A two-scale graph neural operator for simulating lattice structures. arXiv, 2024.

[Kalman60] Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering, Vol. 82, No. 1, pp. 35–45, 1960.

[KapsCzechDuddeck22] Kaps, A.; Czech, C.; Duddeck, F.: A hierarchical kriging approach for multi-fidelity optimization of automotive crashworthiness problems. Structural and Multidisciplinary Optimization, Vol. 65, No. 4, 2022.

[KaramoozMahdiabadiEtAl21] Karamooz Mahdiabadi, M.; Tiso, P.; Brandt, A.; Rixen, D.J.: A non-intrusive model-order reduction of geometrically nonlinear structural dynamics using modal derivatives. Mechanical Systems and Signal Processing, Vol. 147, p. 107126, 2021.

[KarniadakisEtAl21] Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L.: Physics-informed machine learning. Nature Reviews Physics, Vol. 3, No. 6, pp. 422–440, 2021.

[KastGuoHesthaven20] Kast, M.; Guo, M.; Hesthaven, J.S.: A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems. Computer Methods in Applied Mechanics and Engineering, Vol. 364, p. 112947, 2020.

[KawanoScherpen18] Kawano, Y.; Scherpen, J.M.: Structure preserving truncation of nonlinear port Hamiltonian systems. IEEE Transactions on Automatic Control, Vol. 63, No. 12, pp. 4286–4293, 2018.

[KingmaBa14] Kingma, D.; Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations, 2014.

[KingmaWelling13] Kingma, D.P.; Welling, M.: Auto-Encoding Variational Bayes. arXiv preprint, 2013.

[KipfWelling16] Kipf, T.N.; Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. arXiv, 2016.

[KneiflConti25] Kneifl, J.; Conti, P.: jkneifl/VENI-VINDy-VICI: 0.1.8, 2025.

[KneiflEtAl23] Kneifl, J.; Rosin, D.; Avci, O.; Röhrle, O.; Fehr, J.: Low-dimensional data-based surrogate model of a continuum-mechanical musculoskeletal system based on non-intrusive model order reduction. Archive of Applied Mechanics, Vol. 93, No. 9, pp. 3637–3663, 2023.

[KneiflEtAl24] Kneifl, J.; Fehr, J.; Brunton, S.L.; Kutz, J.N.: Multi-hierarchical surrogate learning for explicit structural dynamical systems using graph convolutional neural networks. Computational Mechanics, 2024.

[KneiflGrunertFehr21] Kneifl, J.; Grunert, D.; Fehr, J.: A nonintrusive nonlinear model reduction method for structural dynamical problems based on machine learning. International Journal for Numerical Methods in Engineering, Vol. 122, No. 17, pp. 4774–4786, 2021.

[KneiflHayFehr22a] Kneifl, J.; Hay, J.; Fehr, J.: Human Occupant Motion in Pre-Crash Scenario, 2022.

[KneiflHayFehr22b] Kneifl, J.; Hay, J.; Fehr, J.: Real-time Human Response Prediction Using a Non-intrusive Data-driven Model Reduction Scheme. IFAC-PapersOnLine, Vol. 55, No. 20, pp. 283–288, 2022.

[KneiflRettbergFehr24] Kneifl, J.; Rettberg, J.; Fehr, J.: Coupled thermo-mechanical simulation results of a finite element discbrake, 2024.

[KneiflRettbergHerb24] Kneifl, J.; Rettberg, J.; Herb, J.: ApHIN – Autoencoder-based port-Hamiltonian Identification Networks (Software Package), 2024.

[Koopman31] Koopman, B.O.: Hamiltonian Systems and Transformation in Hilbert Space. Proceedings of the National Academy of Sciences, Vol. 17, No. 5, pp. 315–318, 1931.

[KotyczkaLefèvre19] Kotyczka, P.; Lefèvre, L.: Discrete-time port-Hamiltonian systems: A definition based on symplectic integration. Systems & Control Letters, Vol. 133, p. 104530, 2019.

[KovachkiEtAl23] Kovachki, N.; Li, Z.; Liu, B.; Azizzadenesheli, K.; Bhattacharya, K.; Stuart, A.; Anandkumar, A.: Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs. Journal of Machine Learning Research, Vol. 24, No. 89, pp. 1–97, 2023.

[Kramer91] Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal, Vol. 37, No. 2, pp. 233–243, 1991.

[KramerPeherstorferWillcox24] Kramer, B.; Peherstorfer, B.; Willcox, K.E.: Learning Nonlinear Reduced Models from Data with Operator Inference. Annual Review of Fluid Mechanics, Vol. 56, No. 1, pp. 521–548, 2024.

[Kruskal60] Kruskal, M.D.: Maximal Extension of Schwarzschild Metric. Physical Review, Vol. 119, No. 5, pp. 1743–1745, 1960.

[Kutz16] Kutz, J.N.; Brunton, S.L.; Brunton, B.W.; Proctor, J.L. (Eds.): Dynamic mode decomposition: Data-Driven Modeling of Complex Systems. No. 149 in Other titles in applied mathematics. Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2016.

[KutzBrunton22] Kutz, J.N.; Brunton, S.L.: Parsimony as the ultimate regularizer for physics-informed machine learning. Nonlinear Dynamics, Vol. 107, No. 3, pp. 1801–1817, 2022.

[Larsen00] Larsen, A.: Aerodynamics of the Tacoma Narrows Bridge - 60 Years Later. Structural Engineering International, Vol. 10, No. 4, pp. 243–248, 2000.

[LavinEtAl21] Lavin, A.; Krakauer, D.; Zenil, H.; Gottschlich, J.; Mattson, T.; Brehmer, J.; Anandkumar, A.; Choudry, S.; Rocki, K.; Baydin, A.G.; Prunkl, C.; Paige, B.; Isayev, O.; Peterson, E.; McMahon, P.L.; Macke, J.; Cranmer, K.; Zhang, J.; Wainwright, H.; Hanuka, A.; Veloso, M.; Assefa, S.; Zheng, S.; Pfeffer, A.: Simulation Intelligence: Towards a New Generation of Scientific Methods. arXiv, 2021.

[LeeCarlberg20] Lee, K.; Carlberg, K.T.: Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. Journal of Computational Physics, Vol. 404, p. 108973, 2020.

[LeeEtAl23] Lee, S.; Lee, S.; Jang, K.; Cho, H.; Shin, S.: Data-driven Nonlinear Parametric Model Order Reduction Framework using Deep Hierarchical Variational Autoencoder. arXiv, 2023.

[LepriBacciuSantina23] Lepri, M.; Bacciu, D.; Santina, C.D.: Neural autoencoder-based structure-preserving model order reduction and control design for high-dimensional physical systems. IEEE Control Systems Letters, pp. 1–1, 2023.

[Le GuennecEtAl18] Le Guennec, Y.; Brunet, J.P.; Daim, F.Z.; Chau, M.; Tourbier, Y.: A parametric and non-intrusive reduced order model of car crash simulation. Computer Methods in Applied Mechanics and Engineering, Vol. 338, pp. 186–207, 2018.

[LiEtAl18] Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; Battaglia, P.: Learning Deep Generative Models of Graphs. arXiv, 2018.

[LiEtAl20a] Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A.: Fourier Neural Operator for Parametric Partial Differential Equations, 2020.

[LiEtAl20b] Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; Anandkumar, A.: Neural operator: Graph kernel network for partial differential equations. arXiv, 2020.

[LiEtAl22] Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J.: A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. IEEE Transactions on Neural Networks and Learning Systems, Vol. 33, No. 12, pp. 6999–7019, 2022.

[LiljegrenSailer20] Liljegren-Sailer, B.: On port-Hamiltonian modeling and structure-preserving model reduction. Doctoral thesis, Universität Trier, 2020.

[LiljegrenSailerMarheineke22] Liljegren-Sailer, B.; Marheineke, N.: On snapshot-based model reduction under compatibility conditions for a nonlinear flow problem on networks. Journal of Scientific Computing, Vol. 92, No. 2, 2022.

[LiuEtAl23] Liu, Y.; Ponce, C.; Brunton, S.L.; Kutz, J.N.: Multiresolution convolutional autoencoders. Journal of Computational Physics, Vol. 474, p. 111801, 2023.

[LiuEtAl24] Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T.Y.; Tegmark, M.: KAN: Kolmogorov-Arnold Networks. arXiv, 2024.

[LiuKutzBrunton22] Liu, Y.; Kutz, J.N.; Brunton, S.L.: Hierarchical deep learning of multiscale differential equation time-steppers. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 380, No. 2229, 2022.

[Loh11] Loh, W.Y.: Classification and regression trees. WIREs Data Mining and Knowledge Discovery, Vol. 1, No. 1, pp. 14–23, 2011.

[Lorenz56] Lorenz, E.N.: Empirical orthogonal functions and statistical weather prediction, Vol. 1. Massachusetts Institute of Technology, Department of Meteorology Cambridge, 1956.

[Lorenz63] Lorenz, E.N.: Deterministic Nonperiodic Flow. Journal of the Atmospheric Sciences, Vol. 20, No. 2, pp. 130–141, 1963.

[LSTC21] LSTC, L.S.T.C.: LS–DYNA Keyword User's Manual Volume I. Livermore Software Technology Corporation (LSTC), an ANSYS company, 2021. LS–DYNA R13.

[LuEtAl21] Lu, L.; Jin, P.; Pang, G.; Zhang, Z.; Karniadakis, G.E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence, Vol. 3, No. 3, pp. 218–229, 2021.

[LuschKutzBrunton18] Lusch, B.; Kutz, J.N.; Brunton, S.L.: Deep learning for universal linear embeddings of nonlinear dynamics. Nature Communications, Vol. 9, No. 1, 2018.

[MahmudHuangFu20] Mahmud, M.S.; Huang, J.Z.; Fu, X.: Variational Autoencoder-Based Dimensionality Reduction for High-Dimensional Small-Sample Data Classification. International Journal of Computational Intelligence and Applications, Vol. 19, No. 01, p. 2050002, 2020.

[MahoneyDrineas09] Mahoney, M.W.; Drineas, P.: CUR matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences, Vol. 106, No. 3, pp. 697–702, 2009.

[MarsGaoKutz24] Mars Gao, L.; Kutz, J.N.: Bayesian autoencoders for data-driven discovery of coordinates, governing equations and fundamental constants. Proceedings of the Royal Society A, Vol. 480, No. 2286, p. 20230506, 2024.

[McCormick87] McCormick, S.F.: Multigrid methods. SIAM, 1987.

[McGeeSchmidt85] McGee, L.A.; Schmidt, S.F.: Discovery of the Kalman filter as a practical tool for aerospace and industry. Tech. rep., NASA Langley Research Center, 1985.

[MehrmannUnger23] Mehrmann, V.; Unger, B.: Control of port-Hamiltonian differential-algebraic systems and applications. Acta Numerica, Vol. 32, pp. 395–515, 2023.

[MeijerEtAl13] Meijer, R.; Elrofai, H.; Broos, J.; van Hassel, E.: Evaluation of an Active Multi-Body Human Model for Braking and Frontal Crash Events. In Proceedings of the 23rd International Technical Conference on the Enhanced Safety of Vehicles (ESV), pp. 1–12, Seoul, Republic of Korea: NHTSA, 2013.

[MessengerBortz21] Messenger, D.A.; Bortz, D.M.: Weak SINDy for partial differential equations. Journal of Computational Physics, Vol. 443, p. 110525, 2021.

[Meyer21] Meyer, G.P.: An Alternative Probabilistic Interpretation of the Huber Loss. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5261–5269, 2021.

[MeyerOffin17] Meyer, K.; Offin, D.: Introduction to Hamiltonian dynamical systems and the N-Body problem. Springer Cham, 2017.

[MinEtAl22] Min, E.; Chen, R.; Bian, Y.; Xu, T.; Zhao, K.; Huang, W.; Zhao, P.; Huang, J.; Ananiadou, S.; Rong, Y.: Transformer for Graphs: An Overview from Architecture Perspective. arXiv, 2022.

[MontgomeryPeckVining13] Montgomery, D.C.; Peck, E.A.; Vining, G.G.: Introduction to Linear Regression Analysis. Wiley Series in Probability and Statistics. [s.l.]: Wiley, 5$^{\text{th}}$ Edn., 2013.

[MontiEtAl17] Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; Bronstein, M.M.: Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[Moore81] Moore, B.: Principal component analysis in linear systems: Controllability, observability, and model reduction. IEEE Transactions on Automatic Control, Vol. 26, No. 1, pp. 17–32, 1981.

[MorandinNicodemusUnger23] Morandin, R.; Nicodemus, J.; Unger, B.: Port-Hamiltonian Dynamic Mode Decomposition. SIAM Journal on Scientific Computing, Vol. 45, No. 4, pp. A1690–A1710, 2023.

[MoserLohmann20] Moser, T.; Lohmann, B.: A new Riemannian framework for efficient H2-optimal model reduction of port-Hamiltonian systems. In 2020 59th IEEE Conference on Decision and Control (CDC), pp. 5043–5049, 2020.

[NearyTopcu23] Neary, C.; Topcu, U.: Compositional learning of dynamical system models using port-Hamiltonian neural networks. In Learning for Dynamics and Control Conference, pp. 679–691, PMLR, 2023.

[NicodemusEtAl22] Nicodemus, J.; Kneifl, J.; Fehr, J.; Unger, B.: Physics-informed Neural Networks-based Model Predictive Control for Multi-link Manipulators. IFAC-PapersOnLine, Vol. 55, No. 20, pp. 331–336, 2022.

[Nielsen18] Nielsen, M.A.: Neural Networks and Deep Learning, 2018.

[NivenEtAl24] Niven, R.K.; Cordier, L.; Mohammad-Djafari, A.; Abel, M.; Quade, M.: Dynamical System Identification, Model Selection and Model Uncertainty Quantification by Bayesian Inference, 2024.

[OhlbergerRave16] Ohlberger, M.; Rave, S.: Reduced Basis Methods: Success, Limitations and Future Challenges. Proceedings of the Conference Algoritmy, pp. 1–12, 2016.

[OrtegaGarcíaCanseco04] Ortega, R.; García-Canseco, E.: Interconnection and damping assignment passivity-based control: A survey. European Journal of Control, Vol. 10, No. 5, pp. 432–450, 2004.

[OrtegaYin23] Ortega, J.P.; Yin, D.: Learnability of linear port-Hamiltonian systems. arXiv, 2023.

[OttoMacchioRowley23] Otto, S.E.; Macchio, G.R.; Rowley, C.W.: Learning nonlinear projections for reduced-order modeling of dynamical systems using constrained autoencoders. Chaos: An Interdisciplinary Journal of Nonlinear Science, Vol. 33, No. 11, 2023.

[ParkPark04] Park, K.C.; Park, Y.H.: Partitioned Component Mode Synthesis via a Flexibility Approach. AIAA Journal, Vol. 42, No. 6, pp. 1236–1245, 2004.

[PazKim19] Paz, M.; Kim, Y.H.: Structural Dynamics: Theory and Computation. Springer International Publishing, 2019.

[Pearson01] Pearson, K.: On Lines and Planes of Closest Fit to Systems of Points in Space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, Vol. 2, No. 11, pp. 559–572, 1901.

[PeherstorferEtAl14] Peherstorfer, B.; Butnaru, D.; Willcox, K.; Bungartz, H.J.: Localized Discrete Empirical Interpolation Method. SIAM Journal on Scientific Computing, Vol. 36, No. 1, pp. A168–A192, 2014.

[PeherstorferWillcox16] Peherstorfer, B.; Willcox, K.: Data-driven operator inference for nonintrusive projection-based model reduction. Computer Methods in Applied Mechanics and Engineering, Vol. 306, pp. 196–215, 2016.

[PerneboSilverman82] Pernebo, L.; Silverman, L.: Model reduction via balanced state space representations. IEEE Transactions on Automatic Control, Vol. 27, No. 2, pp. 382–387, 1982.

[PerssonPeraire09] Persson, P.O.; Peraire, J.: Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics. In 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, 2009.

[PfaffEtAl20] Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; Battaglia, P.W.: Learning Mesh-Based Simulation with Graph Networks. International Conference on Learning Representations (ICLR), 2021, 2020.

[PichiMoyaHesthaven24] Pichi, F.; Moya, B.; Hesthaven, J.S.: A graph convolutional autoencoder approach to model order reduction for parametrized PDEs. Journal of Computational Physics, p. 112762, 2024.

[Plewa05] Plewa, T.; Linde, T.; Weirs, V.G. (Eds.): Adaptive Mesh Refinement - Theory and Applications: Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Sept. 3–5, 2003. Springer Berlin Heidelberg, 2005.

[PolyugavanderSchaft10] Polyuga, R.V.; van der Schaft, A.: Structure preserving model reduction of port-Hamiltonian systems by moment matching at infinity. Automatica, Vol. 46, No. 4, pp. 665–672, 2010.

[QuarteroniManzoniNegri16] Quarteroni, A.; Manzoni, A.; Negri, F.: Reduced Basis Methods for Partial Differential Equations. Springer International Publishing, 2016.

[RadfordEtAl18] Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al.: Improving language understanding by generative pre-training. 2018.

[RahamanEtAl19] Rahaman, N.; Baratin, A.; Arpit, D.; Draxler, F.; Lin, M.; Hamprecht, F.; Bengio, Y.; Courville, A.: On the Spectral Bias of Neural Networks. In Chaudhuri, K.; Salakhutdinov, R. (Eds.): Proceedings of the 36th International Conference on Machine Learning, Vol. 97 of Proceedings of Machine Learning Research, pp. 5301–5310, PMLR, 2019.

[RaissiPerdikarisKarniadakis19] Raissi, M.; Perdikaris, P.; Karniadakis, G.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, Vol. 378, pp. 686–707, 2019.

[RanjanEtAl18] Ranjan, A.; Bolkart, T.; Sanyal, S.; Black, M.J.: Generating 3D faces using convolutional mesh autoencoders. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 704–720, 2018.

[Rayleigh96] Rayleigh, J.W.S.B.: The theory of sound, Vol. 2. Macmillan, 1896.

[Raymond99] Raymond, E.: The cathedral and the bazaar. Knowledge, Technology & Policy, Vol. 12, No. 3, pp. 23–49, 1999.

[ReedSchneiderBurney92] Reed, M.; Schneider, L.W.; Burney, R.E.: Investigation of Airbag-Induced Skin Abrasions. In SAE Technical Paper Series, STAPP, SAE International, 1992.

[RegazzoniEtAl24] Regazzoni, F.; Pagani, S.; Salvador, M.; Dede', L.; Quarteroni, A.: Learning the intrinsic dynamics of spatio-temporal processes through Latent Dynamics Networks. Nature Communications, Vol. 15, No. 1, 2024.

[ReisStykel08] Reis, T.; Stykel, T.: Balanced truncation model reduction of second-order systems. Mathematical and Computer Modelling of Dynamical Systems, Vol. 14, No. 5, pp. 391–406, 2008.

[RettbergEtAl23] Rettberg, J.; Wittwar, D.; Buchfink, P.; Brauchler, A.; Ziegler, P.; Fehr, J.; Haasdonk, B.: Port-Hamiltonian fluid–structure interaction modelling and structure-preserving model order reduction of a classical guitar. Mathematical and Computer Modelling of Dynamical Systems, Vol. 29, No. 1, pp. 116–148, 2023.

[RettbergEtAl24] Rettberg, J.; Kneifl, J.; Herb, J.; Buchfink, P.; Fehr, J.; Haasdonk, B.: Data-driven identification of latent port-Hamiltonian systems. arXiv, 2024.

[RezendeMohamedWierstra14] Rezende, D.J.; Mohamed, S.; Wierstra, D.: Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In

Xing, E.P.; Jebara, T. (Eds.): Proceedings of the 31st International Conference on Machine Learning, Vol. 32 of Proceedings of Machine Learning Research, pp. 1278–1286, Bejing, China: PMLR, 2014.

[RodriguezEtAl22] Rodriguez, S.N.; Iliopoulos, A.P.; Carlberg, K.T.; Brunton, S.L.; Steuben, J.C.; Michopoulos, J.G.: Projection-tree reduced-order modeling for fast N-body computations. Journal of Computational Physics, Vol. 459, p. 111141, 2022.

[RonnebergerFischerBrox15] Ronneberger, O.; Fischer, P.; Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pp. 234–241, Springer, 2015.

[RuschBronsteinMishra23] Rusch, T.K.; Bronstein, M.M.; Mishra, S.: A Survey on Oversmoothing in Graph Neural Networks. arXiv, 2023.

[SalnikovFalaizeLozienko23] Salnikov, V.; Falaize, A.; Lozienko, D.: Learning port-Hamiltonian systems—algorithms. Computational Mathematics and Mathematical Physics, Vol. 63, No. 1, pp. 126–134, 2023.

[SalvadorDedèManzoni21] Salvador, M.; Dedè, L.; Manzoni, A.: Non intrusive reduced order modeling of parametrized PDEs by kernel POD and neural networks. Computers & Mathematics with Applications, Vol. 104, pp. 1–13, 2021.

[SanchezGonzalezEtAl20] Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; Battaglia, P.: Learning to simulate complex physics with graph networks. In International Conference on Machine Learning, pp. 8459–8468, PMLR, 2020.

[SarkarScherpen23] Sarkar, A.; Scherpen, J.M.: Structure-preserving generalized balanced truncation for nonlinear port-Hamiltonian systems. Systems & Control Letters, Vol. 174, p. 105501, 2023.

[Schaft20] van der Schaft, A.: Port-Hamiltonian modeling for control. Annual Review of Control, Robotics, and Autonomous Systems, Vol. 3, No. 1, pp. 393–416, 2020.

[SchaftJeltsema14] van der Schaft, A.; Jeltsema, D.: Port-Hamiltonian systems theory: An introductory overview. Foundations and Trends in Systems and Control, Vol. 1, No. 2-3, pp. 173–378, 2014.

[SchiehlenEberhard14] Schiehlen, W.; Eberhard, P.: Applied Dynamics. Heidelberg: Springer, 1st Edn., 2014.

[Schmid10] Schmid, P.J.: Dynamic mode decomposition of numerical and experimental data. Journal of Fluid Mechanics, Vol. 656, p. 5–28, 2010.

[Schmid22] Schmid, P.J.: Dynamic Mode Decomposition and Its Variants. Annual Review of Fluid Mechanics, Vol. 54, No. 1, pp. 225–254, 2022.

[Schmidt07] Schmidt, E.: Zur Theorie der linearen und nichtlinearen Integralgleichungen: I. Teil: Entwicklung willkürlicher Funktionen nach Systemen vorgeschriebener. Mathematische Annalen, Vol. 63, No. 4, pp. 433–476, 1907.

[SchmidtLipson09] Schmidt, M.; Lipson, H.: Distilling free-form natural laws from experimental data. Science, Vol. 324, No. 5923, pp. 81–85, 2009.

[SchölkopfSmolaMüller97] Schölkopf, B.; Smola, A.; Müller, K.R.: Kernel principal component analysis. In Gerstner, W.; Germond, A.; Hasler, M.; Nicoud, J.D. (Eds.): Artificial Neural Networks — ICANN'97, pp. 583–588, Berlin, Heidelberg: Springer Berlin Heidelberg, 1997.

[Schulze23a] Schulze, P.: Energy-based model reduction of transport-dominated phenomena. doctoralthesis, Technische Universität Berlin, 2023.

[Schulze23b] Schulze, P.: Structure-preserving model reduction for port-Hamiltonian systems based on separable nonlinear approximation ansatzes. Frontiers in Applied Mathematics and Statistics, Vol. 9, 2023.

[SchulzSpeekenbrinkKrause18] Schulz, E.; Speekenbrink, M.; Krause, A.: A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. Journal of Mathematical Psychology, Vol. 85, pp. 1–16, 2018.

[Schwarzschild16] Schwarzschild, K.: Über das gravitationsfeld eines massenpunktes nach der einsteinschen theorie. Sitzungsberichte der königlich preussischen Akademie der Wissenschaften, pp. 189–196, 1916.

[Schwerdtner21] Schwerdtner, P.: Port-Hamiltonian system identification from noisy frequency response data. arXiv, 2021.

[SchwerdtnerVoigt23] Schwerdtner, P.; Voigt, M.: SOBMOR: Structured optimization-based model order reduction. SIAM Journal on Scientific Computing, Vol. 45, No. 2, pp. A502–A529, 2023.

[ShenEtAl21] Shen, S.; Yang, Y.; Shao, T.; Wang, H.; Jiang, C.; Lan, L.; Zhou, K.: High-order differentiable autoencoder for nonlinear model reduction. ACM Transactions on Graphics, Vol. 40, No. 4, pp. 1–15, 2021.

[ShiibaFehrEberhard12] Shiiba, T.; Fehr, J.; Eberhard, P.: Flexible multibody simulation of automotive systems with non-modal model reduction techniques. Vehicle System Dynamics, Vol. 50, No. 12, pp. 1905–1922, 2012.

[SimpsonEtAl24] Simpson, T.; Vlachas, K.; Garland, A.; Dervilis, N.; Chatzi, E.: VpROM: a novel variational autoencoder-boosted reduced order model for the treatment of parametric dependencies in nonlinear systems. Scientific Reports, Vol. 14, No. 1, 2024.

[Sirovich87] Sirovich, L.: Turbulence and the dynamics of coherent structures. I. Coherent structures. Quarterly of Applied Mathematics, Vol. 45, No. 3, pp. 561–571, 1987.

[SISS20] SISS: Simcenter Madymo Model Manual, version 2020.1. Siemens Industry Software and Services BV (SISS), Rijswijk, Netherlands, 2020.

[Smith24] Smith, R.C.: Uncertainty Quantification Theory, Implementation, and Applications. Society for Industrial and Applied Mathematics, 2024.

[SoleraRicoEtAl24] Solera-Rico, A.; Sanmiguel Vila, C.; Gómez-López, M.; Wang, Y.; Almashjary, A.; Dawson, S.T.; Vinuesa, R.: β-Variational autoencoders and transformers for reduced-order modelling of fluid flows. Nature Communications, Vol. 15, No. 1, p. 1361, 2024.

[Steinwart08] Steinwart, I.: Support vector machines. EBL-Schweitzer. New York, NYt: Springer, 2008. Description based upon print version of record.

[StraußEtAl24] Strauß, A.; Kneifl, J.; Tkachuk, A.; Fehr, J.; Bischoff, M.: Accelerated Non-linear Stability Analysis Based on Predictions From Data-Based Surrogate Models. International Journal for Numerical Methods in Engineering, Vol. 126, No. 1, 2024.

[Szekeres] Szekeres, G.: On the singularities of a Riemannian manifold. Vol. 7, p. 285.

[Tangirala18] Tangirala, A.K.: Principles of System Identification: Theory and Practice. CRC Press, 2018.

[Toupin65] Toupin, R.A.: Saint-Venant's Principle. Archive for Rational Mechanics and Analysis, Vol. 18, No. 2, pp. 83–96, 1965.

[Trefethen00] Trefethen, L.N.: Spectral methods in MATLAB. No. 10 in Software, environments, tools. Philadelphia, Pa.: Society for Industrial and Applied Mathematics, 2000.

[TrottenbergOosterleeSchuller00] Trottenberg, U.; Oosterlee, C.W.; Schuller, A.: Multigrid. Elsevier, 2000.

[UngerGugercin19] Unger, B.; Gugercin, S.: Kolmogorov n-widths for linear dynamical systems. Advances in Computational Mathematics, Vol. 45, No. 5, pp. 2273–2286, 2019.

[VaswaniEtAl17] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I.: Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, p. 6000–6010, Red Hook, NY, USA: Curran Associates Inc., 2017.

[Volkwein13] Volkwein, S.: Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling. http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Book.pdf, 2013. Accessed 04. August 2022.

[WangEtAl19] Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M.: Dynamic Graph CNN for Learning on Point Clouds. ACM Transactions on Graphics, Vol. 38, No. 5, pp. 1–12, 2019.

[WangHesthavenRay19] Wang, Q.; Hesthaven, J.S.; Ray, D.: Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. Journal of Computational Physics, Vol. 384, pp. 289–307, 2019.

[WangHuanGarikipati19] Wang, Z.; Huan, X.; Garikipati, K.: Variational system identification of the partial differential equations governing the physics of pattern-formation: Inference under varying fidelity and noise. Computer Methods in Applied Mechanics and Engineering, Vol. 356, pp. 44–74, 2019.

[WestGallagher06] West, J.; Gallagher, S.: Challenges of open innovation: the paradox of firm investment in open-source software. R and D Management, Vol. 36, No. 3, pp. 319–331, 2006.

[WilliamsRasmussen06] Williams, C.K.; Rasmussen, C.E.: Gaussian Processes for Machine Learning, Vol. 2. MIT press Cambridge, MA, 2006.

[WilliamsZahnKutz24] Williams, J.P.; Zahn, O.; Kutz, J.N.: Sensing with shallow recurrent decoder networks. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, Vol. 480, No. 2298, 2024.

[WuEtAl21] Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S.: A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, Vol. 32, No. 1, pp. 4–24, 2021.

[Yu20] Yu, R.: A Tutorial on VAEs: From Bayes' Rule to Lossless Compression, 2020.

[YıldızEtAl24] Yıldız, S.; Goyal, P.; Bendokat, T.; Benner, P.: Data-driven identification of quadratic representations for nonlinear Hamiltonian systems using weakly symplectic liftings. Journal of Machine Learning for Modeling and Computing, 2024.

[ZhouEtAl20] Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M.: Graph neural networks: A review of methods and applications. AI Open, Vol. 1, pp. 57–81, 2020.

[ZhuangEtAl21] Zhuang, Q.; Lorenzi, J.M.; Bungartz, H.J.; Hartmann, D.: Model order reduction based on Runge–Kutta neural networks. Data-Centric Engineering, Vol. 2, p. e13, 2021.

[ZienkiewiczTaylorFox14] Zienkiewicz, O.C.; Taylor, R.L.; Fox, D.: The finite element method for solid and structural mechanics. Oxford: Butterworth-Heinemann, 7^{th} Edn., 2014.