FACULTY OF ENGINEERING
AND ARCHITECTURE

Hybrid Physics-Based Neural Network Models for Predicting Nonlinear Dynamics in Mechatronic Applications

Wannes De Groote

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Electromechanical Engineering

Supervisors

Prof. Guillaume Crevecoeur, PhD* - Prof. Sofie Van Hoecke, PhD**

* Department of Electromechanical, Systems and Metal Engineering
Faculty of Engineering and Architecture, Ghent University

** Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

October 2021

GHENT
UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Hybrid Physics-Based Neural Network Models for Predicting Nonlinear Dynamics in Mechatronic Applications

Wannes De Groote

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Electromechanical Engineering

Supervisors
Prof. Guillaume Crevecoeur, PhD* - Prof. Sofie Van Hoecke, PhD**

* Department of Electromechanical, Systems and Metal Engineering
Faculty of Engineering and Architecture, Ghent University

** Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

October 2021

GHENT UNIVERSITY

ISBN 978-94-6355-540-1
NUR 984, 929
Wettelijk depot: D/2021/10.500/88

Members of the Examination Board

Chair

Prof. Gert De Cooman, PhD, Ghent University

Other members entitled to vote

Prof. Jeroen De Kooning, PhD, Ghent University
Prof. Dirk Deschrijver, PhD, Ghent University
Prof. Peter Karsmakers, PhD, KU Leuven
Prof. Steven Waslander, PhD, University of Toronto, Canada

Supervisors

Prof. Guillaume Crevecoeur, PhD, Ghent University
Prof. Sofie Van Hoecke, PhD, Ghent University

Voorwoord

1 Augustus 2017, de uitdaging van mijn doctoraat was duidelijk: uitpluizen op welke manier we het hippe machinaal leren konden combineren met de traditionele modeleringsvormen om zo robuuste dynamische modellen te krijgen om het gedrag te voorspellen mechatronische applicaties. Net afgestudeerd en volgeladen met wiskundige materie waarbij de meest complexe controleproblemen zomaar konden worden opgelost, althans in simulatie dan toch, begon ik vol zelfvertrouwen aan mijn doctoraat. Het superheldgevoel zonk al snel weg toen ik kennis maakte met de fysische machines in het labo. *Hoezo, de meetruis op snelheidssensor volgt geen perfecte gaussverdeling? Dus je bedoelt dat de mechanische wrijving verandert elke keer je de setup opstart?* Gelukkig kreeg ik al snel hulp van Hendrik - die toen nog zijn vaste domicilie in het labo had staan - om mij de nodige praktische kennis bij te brengen over data acquisitie en implementaties van machinesturingen. Het werd mij al snel duidelijk dat de imperfecties die optreden in mechatronische applicaties de theoretische ontwikkelde ontwerp en sturingstechnieken vaak onbruikbaar maakten. Dit werd dan ook mijn motivatie om te werken aan hybride modeleringsvormen om zo het gedrag van machines beter te kunnen voorspellen. De grote kloof tussen de realiteit en de theorie, wat initieel een eyeopener was, werd opeens mijn grootste motivatie en het startschot van mijn vier jaar durend onderzoek was gegeven.

Eerst en vooral wil ik dan ook mijn promotoren, Guillaume en Sofie, oprecht bedanken voor de opportuniteiten en de supervisie. Het professionele huwelijk dat jullie aangingen, waarbij mechatronica en data science elkaar officieel het ja-woord gaven, heeft geleid tot veel interessante discussies die essentieel waren voor mijn doctoraatsstudie. Hiernaast wil ik ook een ongelofelijke dank betuigen aan mijn collega Tom S., waarbij geen enkel onderzoeksidee gespaard is gebleven tijdens onze eindeloze brainstorm sparringsessies. Ook collegae Tom L. en Matthias, bedankt om telkens als ik dacht dat ik op het punt stond de Nobelprijs te winnen, mij via een paar simpele wiskundige operaties de trivialiteit van mijn bevindingen aan te tonen. Verder wil ik natuurlijk ook de andere (ex) bureau genoten bedanken voor de leuke sfeer en verreikende koffiepauzes: Annelies, Thijs, Rikkert, Lynn, Mariem, Shima, Andries, Arne,

Jasper, Thomas, Lennert, Victor, Cedric, Matty en Ahmed. Graag wil ik ook Jeroen D.M., onze M&F business development manager, expliciet bedanken om telkens uw oneindige bron van energie te kanaliseren naar het zoeken van industriële partners waar we onze ontwikkelde methodes op konden toepassen. Het kunnen aantonen van de industriële relevantie was voor mij telkens een grote drijfveer om verder te gaan in mijn onderzoek.

Bovendien wil ik ook graag nog eens benadrukken hoe hard ik apprecieer dat ik tijdens mijn doctoraat de kans heb gekregen om te kunnen bouwen aan verschillende mechatronische applicaties waarop mijn onderzoek kon gevalideerd worden. Hierbij dank ik dan ook Marilyn en Ingrid voor de administratieve ondersteuning en het scheppen van orde in de vele facturen. Ook na mijn vertrek zullen de verkeerd geboorde gaten en de relikwieën van de vele adaptaties aan mijn opstellingen overblijven als aandenken aan mijn technische ontwikkeling. Natuurlijk dank ik hier alles aan Tony en Vincent die met plezier hun technische kennis overbrachten en veelal zelf het heft in handen namen om de machines vakkundig af te werken.

Mijn ouders wil ik heel graag nog eens extra bedanken voor het feit dat ze mij de kans hebben gegeven om te studeren. En tot slot, mag ik zeker ook mijn vriendin, Anabel, niet vergeten. Ik besef dat het leven met een PhD student niet altijd even gezellig was. De typische '*ik zal nog rap even vijf minuten mijn simulaties bekijken*' die uitmondden tot een hele avond programmeren voor de tv, waren hoogstwaarschijnlijk niet altijd meest memorabele momenten voor jou. Toch heb ik altijd je onvoorwaardelijke steun gekregen en kreeg ik alle support om mijn persoonlijke doelen na te streven, waarvoor een dikke merci!

Wannes De Groote, 14 oktober 2021

Contents

Summary

Mechatronic applications are paramount in modern society. Just think of robotic manipulators that help machine operators in production or electric vehicles that become more and more efficient. These are good examples of mechatronic applications that combine mechanics, electronics, control and computer science engineering in a synergetic manner. To make the interactions between various subsystems possible, e.g. the battery and the drivetrain in the electric vehicle, we need advanced design, control and sensing strategies. More than ever, the development of these applications is substantiated by accurate system models that can predict the intricate nonlinear dynamics of these mechatronic systems.

The original development of system models was primarily based on physical insights. Fundamental laws such as described by Newton and Kirchhoff were used to make simplified representations of the considered application. These models typically contain physically interpretable parameters that can be directly measured (e.g., mass, length) or identified from measurements (e.g. identification of friction constant based on speed and force measurements). However, in practice, there is often insufficient expert knowledge to capture all interactions at play by physical laws. Therefore, it often occurs that important phenomena are not incorporated in the physics-based model, leading to discrepancies between the model and the physical system. During recent years, the use of machine learning methods, and in particular deep neural networks, have gained increasing interest to learn the system behavior. These models do not require prior knowledge about the system because they can learn patterns directly from the data by fitting the (non-physically interpretable) model parameters to the measurements. Consequently, all system interactions that are present in the measurements can be captured by the data-driven model, leading to highly accurate system models. Nevertheless, these black-box approaches often invoke skepticism when being included in industrial applications due to their non-interpretable nature. Moreover, these models are typically only valid for operating conditions of which data were included during the training process, making them fail when being evaluated for unseen conditions. Hence, both physics-based and data-driven approaches have their own strengths and

weaknesses.

In this PhD research, we searched for modeling methods that exhibit the benefits of both physics-inspired and data-driven approaches. More specifically, we combined physics-inspired models with neural networks, to obtain interpretable, accurate and robust models that can predict the nonlinear behavior of mechatronic applications. This dissertation encompasses four research challenges applied to three different mechatronic applications.

First, we developed a physics-inspired model of a wet friction clutch mechanism, used to predict the shifting time during different operating conditions. This system became a textbook example to illustrate the shortcomings of physics-inspired models. We addressed the model discrepancies by assigning a stochastic nature to the unmodeled system interactions. Consequently, we identified the parameters of this probabilistic distribution, resulting in a stochastic model for the shifting time. The propagation of the model uncertainty towards an output distribution requires many model evaluations, demanding high computational resources. In this research, the required number of model evaluations was reduced by estimating the higher-order statistical moments via a generalized polynomial chaos framework.

The shortcomings of physics-inspired models are addressed in a second research challenge by studying a crank-slider mechanism that was subject to unknown force interactions. We elaborated on methods to complement a partially known physics model with neural networks to obtain an accurate hybrid model. This model was learned by simultaneously optimizing the physics-inspired and neural parameters, to obtain an optimal fit with the motor data. Moreover, after convergence, we could extract the information captured in the neural network to gain new insights about the force interactions, namely friction phenomena and a non-linear spring that interacts with the slider-crank mechanism.

Thirdly, a complementary modeling approach was studied. Starting from a limited amount of physical knowledge, we studied how this information could be used as additional input features to a neural network architecture. We applied this research on a cam-follower mechanism. For some operating conditions, the follower has the tendency to detach from the cam perimeter, leading to harmful bouncing behavior. We used a recurrent neural network to predict the occurrence of follower jumps based on motor data. We could clearly illustrate the benefits of adding physics-inspired expert knowledge, leading to reliable predictions. Because the interpretability is a key factor to make machine learning applicable for industrial applications, we did an elaborate analysis about the actual contribution of the physical features to the model output. To be more precise, we have built local explanation models (SHAP) to study quantitatively how much each model input contributes, enabling for a better understanding of the importance of the physics-inspired features.

Lastly, we studied the robustness of the hybrid modeling formalism on the cam-follower mechanism. The usefulness of system models for mechatronic applications highly depends on their reliability during the prediction of unexpected situations. In this research, we assessed how models learned on limited sets of design parameters (e.g. cam shape) or control parameters (e.g. voltage) behave when evaluated for unseen system configurations. Here as well, we could demonstrate that the hybrid architectures, encompassing both physics-inspired and neural layers, could better predict unseen system configurations than their data-driven counterparts. Consequently, we could exploit the extrapolation capabilities of these hybrid architectures to estimate the system settings for which the unwanted jump phenomena occur, by learning from a limited set of nominal parameter settings.

In conclusion, various implementations of hybrid physics-based neural network models were built along the lines of the above challenges, to predict the nonlinear system behavior of several mechatronic applications. The obtained research results showed the improved accuracy, robustness and explainability of the presented models, which can be of major importance to develop the next generation of mechatronic applications.

Samenvatting

Mechatronische toepassingen, die mechanica, elektronica, regeltechniek en computerwetenschappen, combineren in één, zijn niet meer weg te denken uit onze moderne maatschappij. Denk maar aan robots die volledige fabriekshallen automatiseren of aan elektrische auto's waar de energieoverdracht van de batterij naar de aandrijftrein zo efficiënt mogelijk gebeurt om het rijbereik te optimaliseren. De complexe interacties tussen de vele subsystemen is alleen maar mogelijk dankzij geavanceerde ontwerp-, controle- en sensorstrategieën die achter deze complexe toepassingen verschuild gaan. Meer en meer wordt er hier dan ook beroep gedaan op accurate systeemmodellen die het niet-lineaire dynamisch gedrag van deze mechatronische applicaties kunnen voorspellen.

Het construeren van deze systeemmodellen stoelde traditioneel op fysische kennis, waarbij basiswetten zoals beschreven door Newton en Kirchhoff gebruikt worden om gesimplificeerde voorstellingen te maken van de beschouwde applicatie. Deze modellen bestaan traditioneel uit fysisch interpreteerbare parameters die direct kunnen gemeten worden (bv. massa, lengte), of moeten geïdentificeerd worden door het model te fitten aan de meetdata (bv. wrijvingsconstante). Een probleem dat hierbij telkens terugkeert is dat er vaak onvoldoende kennis is om alle interacties van het systeem te beschrijven door gekende fysische wetten, waardoor discrepanties tussen het model en het werkelijke systeemgedrag optreden. De laatste jaren is er daarom een heuse opmars van het machinaal leren, en dan vooral van diepe neurale netwerken, die geen voorkennis over het systeem eisen en door het fitten van vele - niet fysisch interpreteerbare - modelparameters aan de meetdata zelf verbanden kunnen leren uit de data. Hierdoor worden alle interacties die tot uiting komen in de meetdata opgenomen in het model, wat vaak leidt tot zeer accurate systeemmodellen. Toch zijn er ook grote nadelen verbonden aan deze data-gedreven technieken. Zo is men vaak sceptisch voor de toepassing in industriële toepassingen door de beperkte interpreteerbaarheid van de modellen. Bovendien zijn deze modellen typisch enkel gelding in de gebieden waarvoor data beschikbaar was tijdens hun trainingsfase, waardoor ze vaak falen bij het voorspellen van onvoorziene omstandigheden.

In dit doctoraat hebben we daarom stevig ingezet op het zoeken naar me-

thodes waarbij fysica-geïnspireerde modellen kunnen gecombineerd met neurale netwerken, om zo interpreteerbare, accurate en robuuste modellen te bekomen voor het voorspellen van het niet-lineaire gedrag van mechatronische systemen. Het doctoraat zelf beschrijft vier onderzoeksbijdragen toegepast op drie mechatronische toepassingen. In een eerste onderzoeksbijdrage hebben we het gedrag van een koppelingsmechanisme proberen fysisch te modeleren voor verschillende werkingsgebieden. De complexiteit van het beschouwde systeem maakte van deze casus al snel een schoolvoorbeeld voor de tekortkomingen van fysica gebaseerde modellen. In dit onderzoek hebben we de discrepanties tussen het model en de metingen proberen te verklaren door een onzekerheid toe te wijzen aan de ongemodelleerde interacties. Vervolgens hebben we de parameters proberen te identificeren van deze probabilistische verdeling, om zo een stochastisch model te bekomen dat de schakeltijd schat. Het doorrekenen van deze modelonzekerheid naar een stochastische predictie vraagt veel rekenkracht. In dit onderzoek zijn we er dan ook in geslaagd om het aantal modelevaluaties drastisch te verminderen door het toepassen van een polynomiale chaos expansie waarbij de hogere momenten van de uitgangsdistributie accuraat konden worden geschat.

Het feit dat fysicamodellen vaak niet perfect zijn door ongekende interacties hebben we dieper onderzocht in een tweede onderzoeksbijdrage waarbij we een krukasmechanisme hebben bestudeerd die onderhevig is aan ongekende krachtwerkingen. Er is uitvoerig onderzocht hoe we het deels-gekende fysica model konden aanvullen via neurale netwerk-topologieën om zo een accuraat hybride model te bekomen. Zowel de fysisch interpreteerbare als de neurale parameters werden simultaan geleerd uit de motordata, om zo een optimale fit te bekomen met de metingen. Na convergentie kon ook nieuwe informatie uit de neurale netwerken gehaald worden, waardoor we nieuwe inzichten verworven over de wrijvingsinteracties en niet-lineaire veer die inwerkt op het krukasmechanisme.

In een derde onderzoeksbijdrage hebben we het modeleringsprobleem benaderd vanuit een andere hoek. Startend van een gelimiteerde hoeveelheid fysica-kennis hebben we bekeken hoe we deze kennis toch kunnen gebruiken als extra informatie die kan meegegeven worden als input van een neurale netwerkarchitectuur. Deze studie is toegepast op een nokmechanisme waarbij voor bepaalde condities het contact tussen de nok en de volger verloren gaat. Er is gebruik gemaakt van *recurrent neural networks* die op basis van de motordata kunnen voorspellen of er zich ontkoppelingsfenomenen voordoen in het mechanisme. Hier is ook duidelijk aangetoond dat de extra fysische informatie meegegeven aan het netwerk zorgt voor betrouwbaardere predicties. Daar de interpreteerbaarheid van de modellen een essentiële factor is voor het toepassen van machinaal leren in industriële omgevingen, is er ook diepgaand onderzoek gedaan naar de effectieve bijdrage van de fysica aan de

ontkoppelingspredicties. Concreet hebben we via lokale benaderingsmodellen (SHAP) kunnen onderzoeken hoeveel elke input van het model kwantitatief bijdraagt, om zo een beter beeld te kunnen schetsen van de invloed van de fysica-geïnspireerde inputs van het model.

Tot slot, in een laatste onderzoeksbijdrage, hebben we de robuustheid van de hybride systeemmodellen geanalyseerd op het nokkenasmechanisme. De bruikbaarheid van de systeemmodellen in mechatronische applicaties hangt sterk af van hoe betrouwbaar ze blijven tijdens het voorspellen van onverwachte situaties. In dit onderzoek hebben we daarom bestudeerd hoe systeemmodellen, die getraind zijn op een beperkte set van ontwerpparameters (bv. nokvorm) en controle variabelen (bv. voltage), zich gedragen bij voorspellingen op ongeziene systeemconfiguraties. Ook hier kon duidelijk worden aangetoond dat de modellen waarbij neurale netwerken gecombineerd worden met traditionele fysica beter het gedrag van ongekende configuraties kunnen voorspellen in vergelijking met hun data-gedreven tegenhangers. Bijgevolg konden we de extrapolatie capaciteiten van deze modellen gebruiken om, startend van data geassocieerd aan goed systeemgedrag, te voorspellen voor welke systeemconfiguraties er zich ongewenste ontkoppelingsfenomenen in het nokkenassystem voordoen.

Deze onderzoeksbijdragen hebben ertoe geleid dat we voor verschillende praktische mechatronische applicaties hebben kunnen de niet-lineaire systeem dynamica voorspellen door het opstellen van verschillende hybride fysica-gebaseerde neurale netwerkmodellen. De bekomen onderzoeksresultaten demonstreren verschillende accurate, robuuste en interpreteerbare modeleringsvormen die kunnen gebruikt worden voor het verder optimaliseren van mechatronische applicaties.

List of Abbreviations

BP	BackPropagation
BPTT	BackPropagation Through Time
CAD	Computer Aided Design
CM	Center of Mass
EMD	Earth Mover's Distance
FE	Forward Euler
FFNN	FeedForward Neural Network
gPC	generalized Polynomial Chaos
LOOCV	Leave-One-Out Cross Validation
LSTM	Long-Short-Term-Memory
MAE	Mean Absolute Error
MC	Monte Carlo
ME	Maximum Entropy
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MPC	Model-based Predictive Control
MSE	Mean Squared Error
NARMAX	Nonlinear AutoRegressive Moving Average model with Exogenous input
NARX	Nonlinear AutoRegressive with eXogenous inputs
NN	Neural Network
ODE	Ordinary Differential Equations
NNAP	Neural Network Augmented Physics Model
PDE	Partial Differential Equation
PDF	Probability Density Function
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SHAP	SHapley Additive exPlanations

Chapter 1

Introduction

The always innovative manufacturing industry is driven by the continuous pursuit of obtaining increased efficiency and performance. Multiple engineering disciplines are at play to optimize each perspective of the manufacturing process. Thereby, in production lines, a series of processing steps transform raw materials to consumer goods. Nowadays these production lines have evolved towards complex aggregations of ingenious automated machines that closely interact with humans and other machines.

From a historical view the industrial revolutions are captured by four paradigm shifts [1] as shown in Fig. 1.1. The first industrial revolution started during the second half of the 18^{th} century and was propelled by inventions that converted mechanical energy from water and steam. New developments in powered mechanisms such as weaving looms enabled increased production in textile factories with a smaller expenditure of human energy. The share of such machines in manufacturing further increased during the second industrial revolution. This revolution focused on the division of labor in which operations of factories are restructured to improve labor productivity that ultimately affected entire segments of the economy. In addition, this included the start of the electrification process in factories. The third revolution is characterized by the digitization that gathered momentum in the 20^{th} century. Rapid advances in electronics and computer science led to various solutions that offered alternatives to tasks traditionally performed by humans. Sensing technologies gave possibilities for robotic systems to interact with the environment enabling assembly tasks and other automated technologies in production processes. In addition, the development of the internet allowed large-scale communication and globalization.

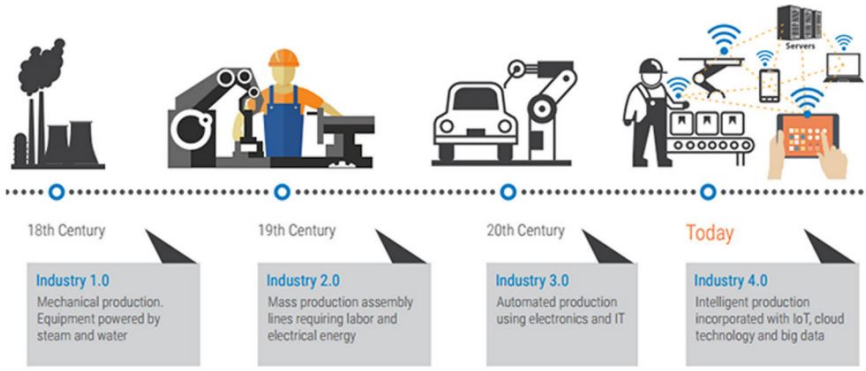


Figure 1.1: Historical perspective of industrial revolutions. Retrieved from [2].

We are currently in the fourth industrial revolution, which is characterized by interconnected manufacturing units that exchange abundant streams of real-time data [2–6]. Technologies such as IoT [7] and cloud computing [8] lead to the essential technological advances that enable intelligent analysis and decision-making to make the transition towards smart manufacturing. The fourth industrial revolution is often associated to the term "Industry 4.0", which became publicly known by an initiative of representatives from business, politics and academia. This initiative was substantiated by the idea of strengthening the competitiveness of the German manufacturing industry in 2011 [9]. The latest advances focus on close interactions between computation, networking and physical processes, commonly referred to as cyber-physical systems (CPS) [10]. The ever-increasing computing capabilities allow to monitor and control the physical processes in an intelligent manner. Feedback loops make sure that the setpoints are tracked, resulting in stable process behavior. Consequently, this leads to large data streams that allow advanced data mining techniques to extract the useful information from the measurements [11]. These big-data techniques have been proven powerful tools to analyze both performance [12, 13] and sustainability [14] of manufacturing processes.

1.1 Mechatronics

The emerging availability of data and fast information processing tools boosted the development of smart electromechanical machines. The field of mechatronics engineering focuses on the integration of various disciplines, such as mechanical, electronics, computer science and control engineering, to obtain smart and high-performing machines. Applications of mechatronic

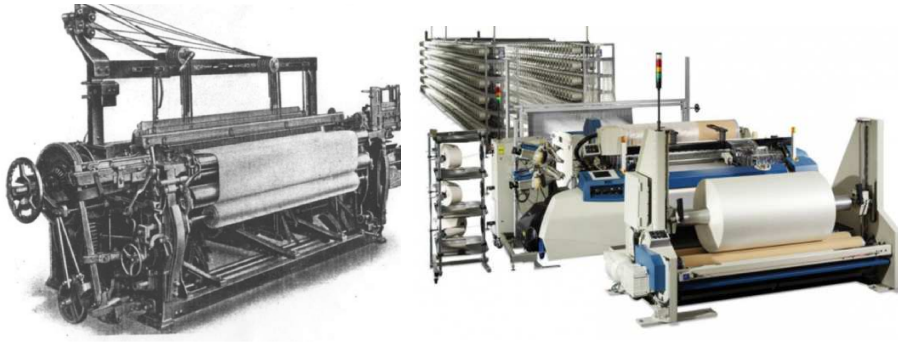
systems can be found on component level (e.g electromagnetic bearings [15] and clutches [16]) or as complete machines that contain many interacting mechatronic subsystems (e.g., electric vehicles [17] and automated assembly lines [18]). Mechatronics engineering emerged in the 1960s due to the advancements in semiconductors, integrated circuits, microprocessors and microcontrollers [19]. Nowadays, the design of mechatronic applications faces new challenges as each application should be integrated in a larger ensemble of interconnected systems [20–22]. The evolution of CNC machines, which are computer controlled manufacturing units that can apply various operations (e.g., milling) on work-pieces, demonstrate this trend. The first implementations (see Fig. 1.2a) required a manual encoding of each motion steps on punched tape, which was then sequentially executed by the machine. Nowadays, CNC machines can interact directly with computer software to translate complete 3D-models into highly precise motions of the servo drives. Furthermore, direct communication with robotic manipulators enables a fully automated manufacturing process, as shown in Fig 1.2b.

(a) **(b)**

Figure 1.2: Mechatronic evolution of CNC machines: starting from punch-tape-controlled devices into almost fully automated production units. (a) MIT's punch tape servo controlled system (1952) [23]. (b) Robot assisted Mazak CNC machine (2018) [24].

Next to integrating the interactions with many other systems, the design of mechatronic applications is challenged by the increasing performance demands of each individual system. The continuous pursuit for increased production performances is illustrated by the evolution of the weaving loom in Fig. 1.3. The weaving loom in Fig. 1.3a illustrates an early automation example for which shuttles, used for the insertion of weft yarn, are returned via a conveyor system that requires 3-4 seconds for the entire operation [25]. By contrast, modern air jet weaving looms (e.g., Fig. 1.3b) can reach up to

1800 picks per minute [25].



(a) **(b)**

Figure 1.3: Mechatronic evolution of a weaving loom: starting from a mechanical device (a) towards a fully interconnected machine (b) that combines various aspects in mechanical, computer science, electrical and control engineering. (a) Northrop shuttle changing loom [25]. (b) Picanol OMNIplus 800 TC airjet weaving machine [26].

The increased productivity of modern mechatronic applications requires mechanical systems that can function at high operational speeds. Consequently, having good insights in the force interactions and the resulting nonlinear dynamics becomes inevitable to assure reliable performances. Therefore, methods that can learn the system behavior from the operational measurement data to provide accurate system predictions can help to design the next generation of high-performing mechatronic applications.

1.2 Digital Representations of Mechatronic Systems

The ability to collect data from machines or entire production entities during operation provides the possibility to build a so-called digital twin [27]. In 2012, the National Aeronautical Space Administration (NASA) introduced the digital twin as a paradigm shift that utilizes high-fidelity simulations to mirror the actual system, enabling to forecast the health, the remaining useful life and the probability of mission success [28]. In practice, the concept of digital twins and traditional computing models are regularly used interchangeably, as many variants of the digital twin definition arose throughout the years [29]. In general, the digital twin concept assumes a real-time digital counterpart of a physical system, for which a modification made to the physical system leads automatically to a change in the digital representation and vice versa.

This requires thus a fully automated information flow. In order to succeed, the digital twin framework needs to be substantiated by an accurate digital model, as illustrated in Fig. 1.4. This model describes a virtual version of a physical system without exhibiting an automatic data exchange. Consequently, the development of a digital model typically requires human interactions to process the data retrieved from the system and translate the model information towards new insights that can be applied to the system.

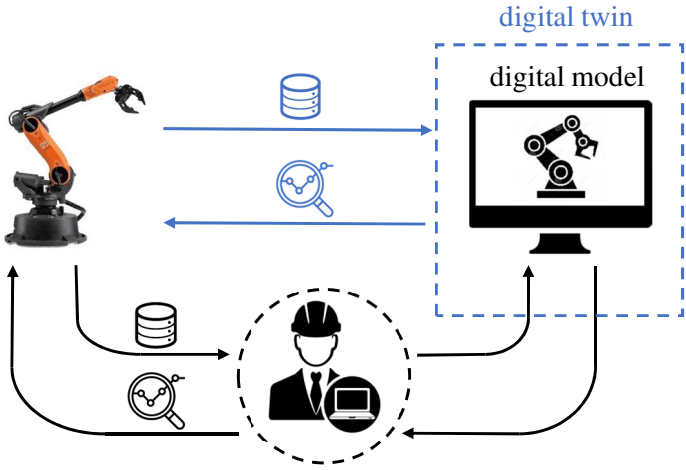


Figure 1.4: Illustration of digital models being used as basis for digital twins.

1.3 Applications of Models for Mechatronic Systems

Digital models of mechatronic applications can help to better understand the complex interactions present in intricate nonlinear systems. Insights from such digital representations have been used for many years to develop advanced techniques to design and improve the performance of real-world mechatronic applications. Below, we provide some validated realizations of dynamic system models used for the design, control and monitoring of mechatronic systems.

1.3.1 Models for Design of Mechatronic Systems

Models are typically used to simulate the nonlinear system behavior for various design choices. This way, multiple scenarios can be emulated offline to study the feasibility of many design possibilities without requiring the actual construction of expensive prototypes. By parameterizing the models and placing them in an optimization loop optimal designs can be determined. Hence,

model-based design approaches can speed-up the development procedures and reduce the cost of expensive prototyping. Numerical methods such as finite elements are typically used to optimize geometrical and material properties of specific machine components such as electric motors [30], gears [31] and machine housings [32]. However, for many mechatronic applications the complexity lies in the interaction between all the components rather than the complexity of the machine parts themselves. Therefore, the overall system is often simplified to a finite dimensional state space representation, defined by a limited amount of lumped parameters (e.g., mass, inertia, spring). This way, simplified models can be used to find optimal design parameter values that ultimately improve the dynamic interactions within mechatronic applications. This has resulted in various advancements in for instance mechanical actuators [33], robotics [34] and electric powertrain systems [35].

1.3.2 Models for Controlling Mechatronic Systems

Next to model-based design, models can help to build control strategies. Traditional methods use feedback approaches in which the control action is each time updated based on the discrepancy between the measured and set-point of the controlled variable [36]. The incorporation of models within the commissioning process can help to optimize these control loops by tuning the control parameters [37] or to determine some key system configurations such as the optimal robot position [38]. The known system dynamics can be used to determine optimal feedback gains that minimize a user-defined cost function (e.g., linear quadratic regulators (LQR)), such as experimentally demonstrated on the control of helicopter maneuvers [39]. However, the increasing complexity ingrained in mechatronic applications and specifically their nonlinear nature, requires the need to incorporate the model dynamics directly in the controller, allowing for online adaptions of the control signals. Feedback-linearization is for instance a concept for which models have been used to transform the nonlinear dynamics of a robotic actuator into an equivalent linear system, which is easier to control [40]. Alternatively, models of robotics manipulators have been used in a dynamic optimization process to determine the optimal feedforward control trajectory to be tracked [41]. Furthermore, explicit model-based predictive control (MPC) approaches calculate the optimal trajectories offline to alleviate the computational burden during the online evaluations, such as applied on electro-hydraulic braking systems [42].

1.3.3 Models for Condition Monitoring of Mechatronic Systems

Condition monitoring is the process in which features and parameters of machines are monitored to detect significant changes that indicate a develop-

ing fault. These features can be measured directly such as accelerations and temperature, or can be derived via mathematical relations. Accurate system models can help to reduce inconsistencies coming from direct observations. Kalman filtering techniques proved valuable to estimate the physical states (e.g., rotational speed) from perturbed measurement data by including digital models [43]. Besides observable states, these techniques also allow to monitor interactions of the system with its environment (e.g., load forces), that are typically difficult to measure [44]. To further substantiate the relevance of using models for monitoring, knowledge on the system dynamics can help to monitor the wear in machine transmissions [45]. Dynamic models have been used in an inverse manner to track the damage of a bearing during operation [46]. Another good example are thermal models that capture temperature flows in machines to detect thermal deficiencies that can harm the system [47].

1.4 Objective and Challenges

The advances in digital twin frameworks that enable automated information flows between intricate mechatronic applications is continuously surging forward. Nevertheless, this trend can only be substantiated if the underlying digital models exhibit the desired accuracy to assure intelligent and reliable autonomous decisions. Furthermore, we showed how the use of accurate dynamic system models seems to be indispensable for the design, control and condition monitoring of the next generation high performing mechatronic applications. Until now, we did not elaborate on the technical details and requirements of these modeling formalisms. In practice, this appears to be a complex topic that has been studied for many years. The models should be able to exhibit very precise predictions to reproduce the continuous speed variations of these highly dynamic systems. Moreover, it appears that system properties (e.g., friction, geometric configurations) induce highly nonlinear dynamics, making the behavior between the changing operating conditions totally different.

Traditionally, physics-inspired methods are used to capture the dynamic behavior based on fundamental physical laws (e.g. Newton, conservation of energy) [48]. The obtained model is typically parametrized by physically-interpretable parameters such as masses, geometrical entities and material properties. Consequently, these models can be easily interpreted and are valid for the entire range for which the used physical laws hold. Nevertheless, it is often very challenging to describe all interactions in mechatronic applications with first-principles, because there are always unknown phenomena that are not included in the model. Consequently, in many cases, we can observe a so-called reality gap, which means that discrepancies between the model

predictions and the actual system arise.

In contrast, data-driven models can learn the system behavior directly from the data without requiring prior system knowledge [49]. The models typically contain many non-interpretable parameters, which are determined by fitting (or training) the model to the system measurements. Consequently, all system interactions that are present in the data will be included in the model, leading to highly accurate prediction models. Nevertheless, the non-interpretable nature of these black-box models typically encounters reluctance from mechatronics engineers. Moreover, these models are typically only valid in regions for which they have seen training data, making them unreliable when being evaluated outside the training data, i.e. outside nominal behavior, for varying conditions or during unexpected circumstances.

A more elaborate discussion about both physics-inspired and data-driven modeling approaches will be given in the next chapter, but we can already say that both formalisms encounter a reality gap. This hinders their direct inclusion in design or control loops of mechatronic applications. Therefore, in this dissertation we focused on the development of hybrid modeling approaches that combine the best of both worlds, as shown in Fig. 1.5. The physically interpretable parameter set (e.g. mass, length), that determines physics-inspired model behavior, is denoted by $\mathbf{p}$. Conversely, the non-interpretable parameter set (e.g. neural network weights), that specifies the data-driven model behavior, is denoted by $\boldsymbol{\alpha}$. This way, we endeavor the development of modeling formalisms that are accurate, reliable and explainable to learn the nonlinear dynamics in mechatronic systems. Moreover, we aim for approaches that also can be experimentally validated on mechatronic applications to assess their actual impact on future mechatronic systems. To further drive innovations in mechatronics engineering and achieve these objectives, we address the following specific challenges:

- **Prediction performances**
  The key objective of a digital model is to accurately predict the system behavior. By combining traditional physical insights with neural network models we aim to enhance prediction performances. To assess the influence of the amount of physics incorporated in the overall model, we study various architectures. We start from incomplete physics models, for which the unmodeled interactions are complemented with neural networks, towards stand-alone black-box models for which we provide physics-inspired input features.

- **Robustness**
  Obtaining hybrid modeling architectures that exhibit accurate prediction performances do not suffice for direct applicability in industrial mechatronic applications. For many mechatronic systems the robustness of

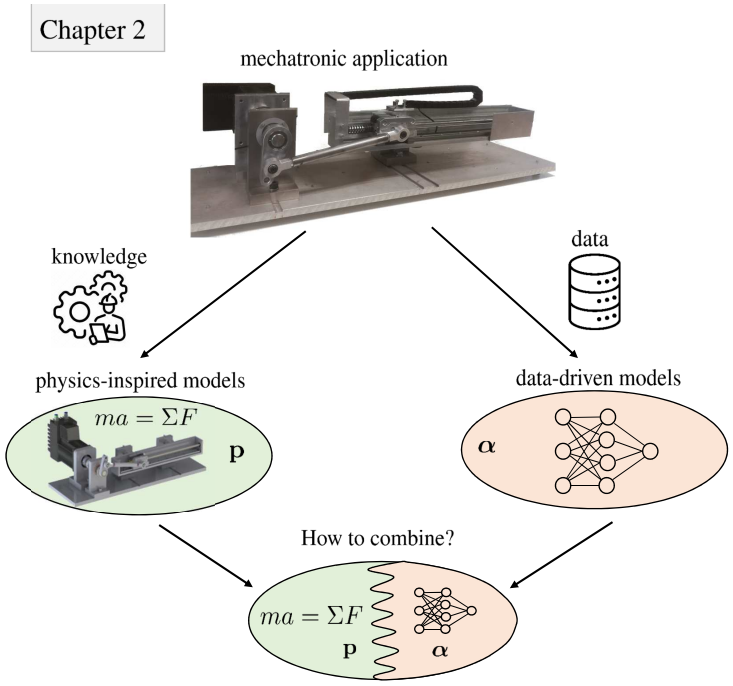


Figure 1.5: Schematic representation of physics-inspired and data-driven models used to capture the dynamics of mechatronic applications. The physics-inspired and data-driven models are parameterized by $\mathbf{p}$ and $\boldsymbol{\alpha}$, respectively.

the software during operation is key to prevent unexpected machine failures. This implies that the models should provide reliable predictions for a wide range of operating conditions, even if there was no data available during the training stage. Therefore, the data-efficiency of the models will be assessed by emulating realistic scenarios for which only scarce datasets exist. This way, we can study the generalization capabilities, and subsequently the robustness of the models when being evaluated beyond regions of seen training data.

- **Explainability**
  The non-interpretable nature of the neural networks often invokes skepticism when being applied on industrial applications. By making well-thought connections between physics-inspired and neural layers, we aim to retrieve insights about the unmodeled phenomena. This way, we can make the information captured in the data-driven components explainable, making them more suitable to further improve design and operations of mechatronic applications. Furthermore, the joint identifica-

tion of physics-inspired and black-box parameters results into a high-dimensional optimization problem, potentially leading to non-unique solutions. Therefore, research is devoted to assess the identifiability of the black-box and physical parameters, enabling more reliable insights retrieved from the hybrid model. Lastly, it is hard to quantify the actual contribution of the physical insights to the model output. Therefore, we study the possibility to develop explanation models that objectively determine the contribution of the different (physics-inspired) features.

- **Experimental Validation**
  Many research devoted to novel machine learning approaches is benchmarked on synthetic data, retrieved from simplified physics-based models of dynamic systems (e.g., pendulum). A challenging aspect to make the development of data-driven approaches applicable for industrial mechatronic applications is to make them robust against sensor noise and other disturbances. Therefore, all presented approaches are validated on experimental data retrieved from various mechatronic systems, each characterized with their own nonlinear dynamic behavior. By including multiple applications within this research, we can fairly assess the possibilities to use physics-based hybrid neural network models for predicting the nonlinear dynamics in mechatronic systems.

1.5 Thesis Outline

The structure of this thesis follows the step by step development of hybrid modeling approaches, based on the defined research objectives. Each of the next chapters outlines a distinct aspect of the research in greater detail. Figure 1.6 gives an overall overview of the research performed in each of these chapters.

Chapter 2 provides a brief introduction on the modeling aspects of mechatronic applications. The concept of static and dynamic system models is explained next to the introduction of the current black-box and white-box modeling approaches. This chapter provides an introduction to neural network models, which will by the central thread of this dissertation. Furthermore, we detail the shortcomings associated with traditional white- and black- box models, being the incentive to study hybrid approaches that combine the benefits of both methodologies.

Chapter 3 introduces the computational burden associated with physics-inspired models for complex mechatronic systems. A white-box dynamic system model of a wet-friction clutch is deduced based on first-principles.

Experimental data from a laboratory wet-friction clutch setup are used to identify a parameterized physics model. Hence, the model is used to estimate the duration of the engagement process of this clutch mechanism for varying operating conditions. Although the dynamic models seems to be valuable, discrepancies with experimental measurements remain. The mismatch between the model and measurements can be attributed to uncertainties that could not be captured by the physical laws. We address this problem by introducing additional flexibility to the model architecture by assigning a probability distribution to uncertain physical parameters (e.g., initial piston position). Hence, a generalized Polynomial Chaos (gPC) framework with moment matching is proposed as a computationally efficient manner to propagate the uncertainty through the model. Consequently, the distributions of the uncertain physical parameters are identified by aligning the derived output distribution with the experimental measurements of the lab setup.

Chapter 4 addresses the problem of having an incomplete physics model by complementing the unknown interactions with data-driven components. A hybrid modeling architecture is presented that combines both physics-inspired and neural network layers. The approach is applied on experimental data collected from a slider-crank setup for which the unknown external load interactions (i.e., nonlinear spring and friction) are replaced by a neural network in the hybrid model architecture. The model is optimized by simultaneously identifying the physical and neural parameters, solely by using data of the state measurements and control input. Consequently, we present an identifiability study to determine the set of physical parameters that can be uniquely determined in combination with the black-box parameters in the neural network. Next to analyzing the prediction performances of the converged overall model, we extract the information captured in the data-driven components. This way, models can be distilled that explain the system interactions that were a priori not physically modeled nor measured.

Chapter 5 studies the possibility to use a limited amount of known physics relations as input features of a black-box timeseries model. The approach is experimentally validated on a cam-follower setup. In nominal circumstances the follower tracks perfectly the cam perimeter. However, for some operating conditions, the follower starts to detach from the cam, which leads to hazardous operating conditions. Therefore, we study the ability of dedicated neural network models, i.e. LSTM models, to monitor the follower contact based on motor measurements. Research is devoted to develop physics-inspired input features, based on knowledge about the cam-dynamics, to increase the prediction performances of these black-box models. The effect of the physics-inspired knowledge provided to the model is extensively tested

for sparse datasets. Furthermore, we present the implementation of an additive feature attribution method to quantitatively determine the contribution of the physics-inspired features to the model prediction.

Chapter 6 presents the ability of hybrid models to learn the influence of certain design and control parameters on the system behavior. The hybrid modeling architecture discussed in Chapter 2 is applied on experimental data of the cam-follower setup. Instead of learning the system dynamics for one specific configuration, we learn to predict the behavior for various (given) system modifications (e.g., cam shape, follower mass). This way, we aim to predict the dynamics of the cam-follower mechanism for unseen system parameter settings, beyond the region for which the model has seen training data. This enables the exploration of the parameter space to determine the set of critical parameters for which detachment between cam and follower occurs.

Chapter 7 concludes this dissertation by providing an overview of the research results of this thesis. We also discuss the possibilities and potential challenges for future research regarding this topic.

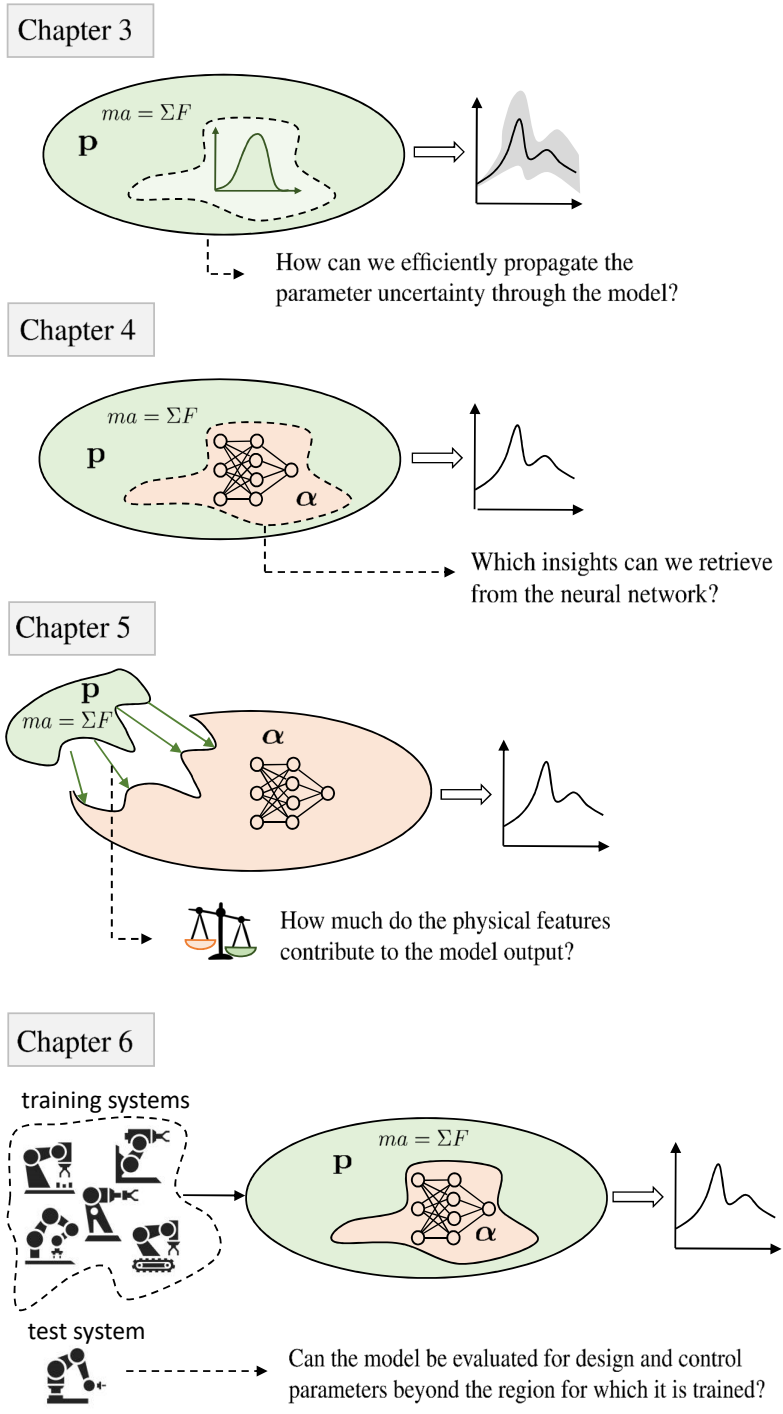


Figure 1.6: Overview of the different chapters in this dissertation. The physics-inspired model structures (green) are defined by the parameters in $\mathbf{p}$ whereas the data-driven components (orange) are characterized by the variables in $\boldsymbol{\alpha}$.

1.6 Research Contributions

This thesis provides an in-depth study to obtain accurate, robust and explainable hybrid physics-based data-driven models that provide the capability to predict the nonlinear dynamics of mechatronic applications. The challenges stipulated in Section 1.4 are being addressed by following research contributions:

Prediction performances

- In a first approach to address the 'reality gap', i.e. mismatches between physics-based model behavior and data, we considered the physical model parameters as stochastic. Hence, the probability distribution function of the key parameters could be propagated through the nonlinear model. This way, a probability distribution of the model output could be obtained. The probability density function of the model parameters is determined by aligning the corresponding model output distribution with the measurements. We proposed a moment matching framework substantiated by generalized polynomial chaos (gPC) expansion to alleviate the computational burden of propagating the parameter uncertainty through the model.

- A hybrid modeling framework is proposed to provide a more profound approach to compensate for incomplete knowledge or insights on the prevailing physical phenomena. The known dynamics are modeled as custom physics-inspired network layers, while the unmodeled system interactions are replaced by neural layers. The model is trained by simultaneously optimizing the physical and neural parameters, enabling for an overall hybrid model that accurately estimates the nonlinear system behavior.

- Modern black-box approaches such as LSTM neural networks can learn important features from time series data. They are often praised to learn intricate patterns directly from raw data. We studied the ability to alleviate the model complexity by providing physics-inspired features to the model, enabling for better generalization and improved predictive performances.

Robustness

- Traditional neural network models become unreliable if insufficient data is available to learn the global dynamics. We show that the required amount of data to learn the system behavior can be reduced if part of the

dynamics are captured by physics-inspired expert knowledge. Consequently, the presented hybrid architectures exhibit better generalization capabilities and, therefore, become a reliable approach to make accurate models on scarce datasets.

- The generalization capabilities of the hybrid modeling architectures are well studied by assessing the ability to predict the system behavior for operating conditions far beyond the regions for which the model is trained. Various extrapolation experiments, implying that the models are evaluated for design and control parameters far beyond the settings included in the training set, show the enhanced robustness due to the physical backbone ingrained in the hybrid models.

Explainability

- The neural layers in the presented hybrid modeling architectures compensate for the interactions that are not included in the physics-inspired network layers. By propagating the model errors through both the neural and physics-inspired layers, we do not need direct data of the unmodeled phenomena. After convergence, we achieved to extract the information captured in the network to retrieve new insights about force interactions.

- The simultaneous optimization of the physics-inspired and neural layers provided new insights in physical parameters that could not be directly measured. Nevertheless, these high-dimensional optimization problems often get stuck in local minima, leading to non-unique solutions. Consequently, we presented the use of a sensitivity study to determine the set of parameters that can be simultaneously optimized with the neural network.

- Despite the beneficial predictive performances obtained by providing physics-inspired features to the LSTM model, it remains challenging to distill the actual contribution of the physics to the model output. Therefore, we implemented an additive feature attribution method that serves as an explanation model on top of the actual prediction model, enabling for an objective quantification of the contribution of the physics-inspired input features to the model output.

Experimental Validation

- The presented prediction models are validated on experimental data of various mechatronic applications. Figure 1.7 shows the setups (i.e., wet friction clutch, slider-crank and cam-follower mechanism) for which the nonlinear behavior has been studied during this research. We could

demonstrate that the methodologies are sufficiently resilient to cope with impurities such as sensor noise and environmental disturbances.

- The hybrid models are programmed in a generic way so that the framework can easily be applied for other use cases. The known dynamics of new applications can be easily given as input to the software so that the required connections in the overall model can be automatically generated. The framework is furthermore implemented in both the Python and Matlab programming languages to be applicable for a large amount of systems for which a partially known physics model is available.

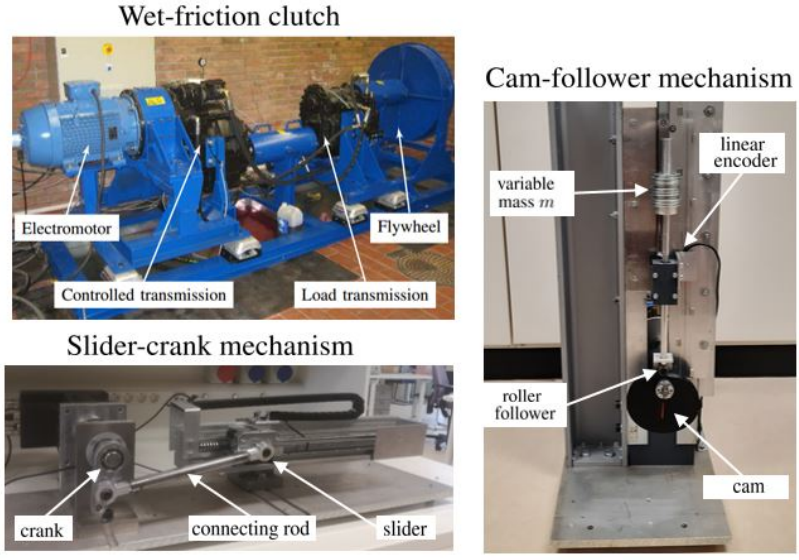


Figure 1.7: Overview of the setups used to validate the presented approaches in this dissertation.

1.7 Scientific Publications

The results obtained during this PhD research have been published in scientific journals and presented at international conferences. The following list provides an overview of these publications.

1.7.1 International SCI Journals

1. **W. De Groote**, T. Lefebvre, G. Tod, N. De Geeter, B. Depraetere, S. Van Poppel and G. Crevecoeur. ”Inverse parametric uncertainty identification using polynomial chaos and high-order moment matching bench-

marked on a wet friction clutch". *Mechatronics*, vol. 65, 2020, DOI: 10.1016/j.mechatronics.2019.102320.

2. **W. De Groote**, E. Kikken, E. Hostens, S. Van Hoecke and G. Crevecoeur, "Neural Network Augmented Physics Models for Systems with Partially Unknown Dynamics: Application to Slider-Crank Mechanism". *IEEE/ASME Transactions on Mechatronics*, accepted 2021, DOI: 10.1109/TMECH.2021.3058536.

3. **W. De Groote**, S. Van Hoecke and G. Crevecoeur, "Prediction of detachment phenomena in cam-follower mechanisms: the benefit of using physics-inspired features in recurrent neural networks". *Mechanical Systems and Signal Processing*, vol. 166, 2022, DOI: 10.1016/j.ymssp.2021.108453.

4. **W. De Groote**, S. Van Hoecke and G. Crevecoeur, "Physics-Based Neural Network Models for Prediction of Cam-Follower Dynamics Beyond Nominal Operations". *IEEE/ASME Transactions on Mechatronics*, accepted, 2021, DOI: 10.1109/TMECH.2021.3101420.

1.7.2 Proceedings of International Conferences

1. G. Tod, **W. De Groote**, T. Lefebvre, N. De Geeter, B. Depraetere, and G. Crevecoeur. "Parametric uncertainty quantification using polynomial chaos expansions applied to a wet friction clutch model". *International Journal of Modeling and Optimization*, vol. 9, no. 4, pp. 185-191, 2019, DOI: 10.7763/IJMO.2019.V9.707.

2. **W. De Groote**, E. Kikken, S. Goyal, S. Van Hoecke, E. Hostens, and G. Crevecoeur. "Hybrid derivative functions for identification of unknown loads and physical parameters with application on slider-crank mechanism". *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Hong Kong, 2019, pp. 1049-1054, DOI: 10.1109/aim.2019.8868376.

References

[1] E. G. Popkova, Y. V. Ragulina, and A. V. Bogoviz. Fundamental differences of transition to industry 4.0 from previous industrial revolutions. In Industry 4.0: Industrial Revolution of the 21st Century, pages 21–29. Springer, 2019.

[2] E. Oztemel and S. Gursev. Literature review of industry 4.0 and related technologies. Journal of Intelligent Manufacturing, 31(1):127–182, 2020.

[3] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman. Intelligent manufacturing in the context of industry 4.0: a review. Engineering, 3(5):616–630, 2017.

[4] V. Alcácer and V. Cruz-Machado. Scanning the industry 4.0: A literature review on technologies for manufacturing systems. Engineering Science and Technology, an International Journal, 2019.

[5] M. Ghobakhloo. The future of manufacturing industry: a strategic roadmap toward industry 4.0. Journal of Manufacturing Technology Management, 2018.

[6] K.-D. Thoben, S. Wiesner, and T. Wuest. Industrie 4.0 and smart manufacturing-a review of research issues and application examples. International journal of automation technology, 11(1):4–16, 2017.

[7] A. Čolaković and M. Hadžialić. Internet of things (iot): A review of enabling technologies, challenges, and open research issues. Computer Networks, 144:17–39, 2018.

[8] J. Surbiryala and C. Rong. Cloud computing: History and overview. In 2019 IEEE Cloud Summit, pages 1–7. IEEE, 2019.

[9] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster. Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group. Forschungsunion, 2013.

[10] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. Cyber-physical systems in manufacturing. Cirp Annals, 65(2):621–641, 2016.

[11] Y. Cheng, K. Chen, H. Sun, Y. Zhang, and F. Tao. Data and knowledge mining with big data towards smart production. Journal of Industrial Information Integration, 9:1–13, 2018.

[12] A. Mangal and N. Kumar. Using big data to enhance the bosch production line performance: A kaggle challenge. In 2016 IEEE International Conference on Big Data (Big Data), pages 2029–2035. IEEE, 2016.

[13] J. Kurpanik, J. Henzel, M. Sikora, Ł. Wróbel, and M. Drewniak. Eye: Big data system supporting preventive and predictive maintenance of

robotic production lines. In International Conference: Beyond Databases, Architectures and Structures, pages 47–60. Springer, 2018.

[14] Y. Zhang, S. Ren, Y. Liu, and S. Si. A big data analytics architecture for cleaner manufacturing and maintenance processes of complex products. Journal of Cleaner Production, 142:626–641, 2017.

[15] N. R. Hemenway and E. L. Severson. Three-pole magnetic bearing design and actuation. IEEE Transactions on Industry Applications, 56(6):6348–6359, 2020.

[16] B. Gao, H. Chen, Q. Liu, and H. Chu. Position control of electric clutch actuator using a triple-step nonlinear method. IEEE Transactions on Industrial Electronics, 61(12):6995–7003, 2014.

[17] G. Du, W. Cao, S. Hu, Z. Lin, and T. Yuan. Design and assessment of an electric vehicle powertrain model based on real-world driving and charging cycles. IEEE Transactions on Vehicular Technology, 68(2):1178–1187, 2018.

[18] F. Dragomir, E. Mincă, O. E. Dragomir, and A. Filipescu. Modelling and control of mechatronics lines served by complex autonomous systems. Sensors, 19(15):3266, 2019.

[19] U. S. Dixit, M. Hazarika, and J. P. Davim. History of mechatronics. In A brief history of mechanical engineering, pages 147–164. Springer, 2017.

[20] D. Bradley, D. Russell, I. Ferguson, J. Isaacs, A. MacLeod, and R. White. The internet of things–the future or the end of mechatronics. Mechatronics, 27:57–74, 2015.

[21] S. K. V. Ragavan and M. Shanmugavel. Engineering cyber-physical systems: Mechatronics wine in new bottles? In 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pages 1–5. IEEE, 2016.

[22] L. Escobar, N. Carvajal, J. Naranjo, A. Ibarra, C. Villacís, M. Zambrano, and F. Galárraga. Design and implementation of complex systems using mechatronics and cyber-physical systems approaches. In 2017 IEEE International Conference on Mechatronics and Automation (ICMA), pages 147–154. IEEE, 2017.

[23] William Pease. An automatic machine tool. Scientific American, 187(3):101–115, 1952.

[24] Robo job. https://www.robojob.eu/en/references/sud-precision. Accessed: 2021-07-15.

[25] K.L. Gandhi. The fundamentals of weaving technology. In Woven textiles, pages 167–270. Elsevier, 2020.

[26] Picanol. https://www.picanol.be/machines-features/machines/omniplus-800-tc-airjet-weaving-machine. Accessed: 2021-07-15.

[27] A. Rasheed, O. San, and T. Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. IEEE Access, 8:21980–22012, 2020.

[28] E. Glaessgen and D. Stargel. The digital twin paradigm for future nasa and us air force vehicles. In 53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA, page 1818, 2012.

[29] A. Fuller, Z. Fan, C. Day, and C. Barlow. Digital twin: Enabling technologies, challenges and open research. IEEE Access, 8:108952–108971, 2020.

[30] Z. Huang and J. Fang. Multiphysics design and optimization of high-speed permanent-magnet electrical machines for air blower applications. IEEE Transactions on Industrial Electronics, 63(5):2766–2774, 2016.

[31] C.-H. Li, H.-S. Chiou, C. Hung, Y.-Y. Chang, and C.-C. Yen. Integration of finite element analysis and optimum design on gear systems. Finite Elements in Analysis and Design, 38(3):179–192, 2002.

[32] B. Xu, S. Ye, and J. Zhang. Numerical and experimental studies on housing optimization for noise reduction of an axial piston pump. Applied Acoustics, 110:43–52, 2016.

[33] A. Kamadan, G. Kiziltas, and V. Patoglu. Co-design strategies for optimal variable stiffness actuation. IEEE/ASME Transactions on Mechatronics, 22(6):2768–2779, 2017.

[34] M. Harant, M. Sreenivasa, M. Millard, N. Šarabon, and K. Mombaur. Parameter optimization for passive spinal exoskeletons based on experimental data and optimal control. In 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), pages 535–540. IEEE, 2017.

[35] S. J. Kim, K.-S. Kim, and D. Kum. Feasibility assessment and design optimization of a clutchless multimode parallel hybrid electric powertrain. IEEE/ASME Transactions on Mechatronics, 21(2):774–786, 2015.

[36] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. Feedback control of dynamic systems, volume 4. Prentice hall Upper Saddle River, NJ, 2002.

[37] R. P. Borase, D. K. Maghade, S.Y. Sondkar, and S.N. Pawar. A review of pid control, tuning methods and applications. International Journal of Dynamics and Control, pages 1–10, 2020.

[38] Y. Bu, W. Liao, W. Tian, J. Zhang, and L. Zhang. Stiffness analysis and optimization in robotic drilling application. Precision Engineering, 49:388–400, 2017.

[39] H. Liu, G. Lu, and Y. Zhong. Robust lqr attitude control of a 3-dof laboratory helicopter for aggressive maneuvers. IEEE Transactions on Industrial Electronics, 60(10):4627–4636, 2012.

[40] Y. Kali, M. Saad, and K. Benjelloun. Optimal super-twisting algorithm with time delay estimation for robot manipulators based on feedback linearization. Robotics and Autonomous Systems, 108:87–99, 2018.

[41] A. Steinhauser and J. Swevers. An efficient iterative learning approach to time-optimal path tracking for industrial robots. IEEE Transactions on Industrial Informatics, 14(11):5200–5207, 2018.

[42] H. Ziyuan, H. Bangcheng, and L. Yun. An explicit nonlinear model predictive abs controller for electro-hydraulic braking systems. IEEE Transactions on Industrial Electronics, 2019.

[43] J. Zhao, M. Netto, and L. Mili. A robust iterated extended kalman filter for power system dynamic state estimation. IEEE Transactions on Power Systems, 32(4):3205–3216, 2016.

[44] B. Forrier, F. Naets, and W. Desmet. Broadband load torque estimation in mechatronic powertrains using nonlinear kalman filtering. IEEE Transactions on Industrial Electronics, 65(3):2378–2387, 2017.

[45] D. Papageorgiou, M. Blanke, H. H. Niemann, and J. H. Richter. Robust backlash estimation for industrial drive-train systems: Theory and validation. IEEE Transactions on Control Systems Technology, 27(5):1847–1861, 2018.

[46] C. J. Li and H. I. Shin. Tracking bearing spall severity through inverse modeling. In ASME 2004 international mechanical engineering

Congress and Exposition, pages 49–54. American Society of Mechanical Engineers Digital Collection, 2004.

[47] P. Nguyen Phuc, H. Vansompel, D. Bozalakov, K. Stockman, and G. Crevecoeur. Inverse thermal identification of a thermally instrumented induction machine using a lumped-parameter thermal model. Energies, 13(1):37, 2020.

[48] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg. System Dynamics: Modeling, Simulation, and Control of Mechatronic Systems. Wiley, 2012.

[49] J. Schoukens and L. Ljung. Nonlinear system identification: A user-oriented road map. IEEE Control Systems Magazine, 39(6):28–99, 2019.

Chapter 2

Modeling Formalism for Mechatronic Systems

The premise of modeling techniques is to digitally represent 'something'. In practice, the detailed description varies depending on the scientific field in which the modeling aspect is studied. Information or insights obtained from the model can be used to further advance in the respective fields. In this study, a selection of important modeling aspects is introduced that are relevant within the field of mechatronics engineering. Figure 2.1 illustrates the different sections discussed in this chapter. First, in Section 2.1 we will introduce the concept of static and dynamic system models. Next, for both the static (Sec. 2.2 and Sec. 2.3) and dynamic (Sec. 2.4 and Sec. 2.5) modeling techniques, we will discuss the relevant physics-inspired and data-driven approaches. Lastly, in Section 2.6, we conclude with hybrid approaches and their necessity, substantiated by the current state of the art. This way, we introduce the relevant concepts mentioned in the subsequent chapters.

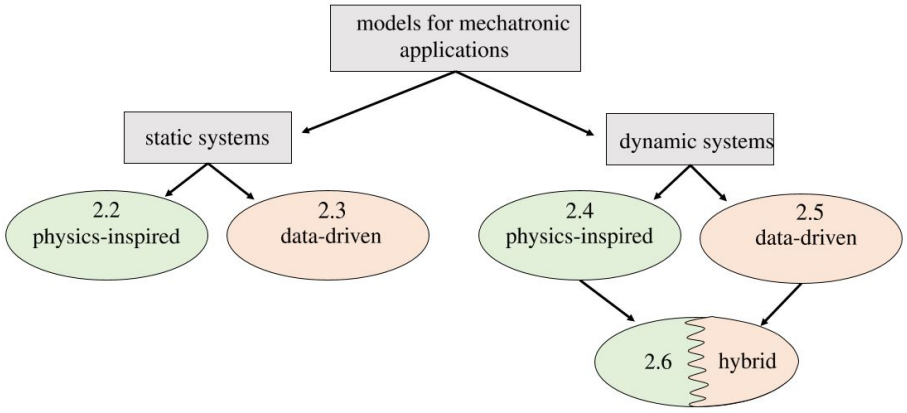


Figure 2.1: Schematic representation of the modeling architectures discussed in this literature overview.

2.1 Static vs. Dynamic Systems Models

The model choice for a specific modeling task highly depends on the nature of the system. First of all, a distinction between static and dynamic system models needs to be made. A static system model is a combination of mathematical operations that transforms an external input $\mathbf{u} \in \mathbb{R}^{n_i}$ to an estimation $\hat{\mathbf{y}}$ of output $\mathbf{y} \in \mathbb{R}^{n_o}$. Figure 2.2 illustrates a schematic representation of a static model. The term static refers to the absence of a time dependent component within the model. Therefore, the output of the model only depends on the values in $\mathbf{u}$ and is independent of the values of $\mathbf{u}$ at prior time instances. The piston position for given angle of a crankshaft is an example of a static relation, because no matter what prior positions were obtained, the current piston position only depends on the present crankshaft angle. A similar dependency can be found in electrical circuits, e.g. the voltage output of a resistor depends on the current through the resistor and the static resistance value.

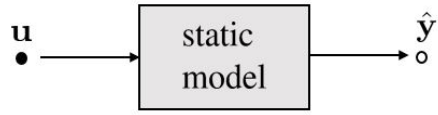


Figure 2.2: Schematic representation of static model.

Dynamic system models describe the behavior of a system over time. In contrast to static system models, the influence of the same external input $\mathbf{u}$ at two different moments will lead to a different output of the system. For instance, a torque input applied to a drivetrain system will have a different influence when applied during steady state versus during start-up. Therefore, these models typically process an entire sequence of control inputs $\mathbf{u}(t)$ into a corresponding output signal $\hat{\mathbf{y}}(t)$, as illustrated in Figure 2.3. In practice, the behavior of mechatronic applications typically has a dynamic nature. These complex nonlinear dynamics need to be learned to develop proper design, control and sensing strategies, and are therefore the main focus of this dissertation. Before diving into advanced modeling strategies a brief introduction on physics-inspired and data-driven static modeling approaches is given. These often serve as conceptual basis for the more complex approaches used for dynamic systems modeling.

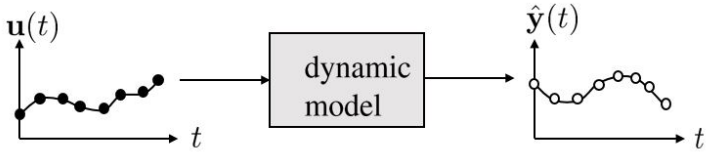


Figure 2.3: Schematic representation of a dynamic system model.

2.2 Static System Models: Physics-Inspired

As mentioned before, a static model $\mathcal{M}$ does not rely on prior values of the model input $\mathbf{u}$ to predict the output $\hat{\mathbf{y}}$. The subclass of physics-inspired static models describes a mapping between inputs and outputs based on fundamental physical laws. For example, take an electronic circuit for which the voltage drop over a resistor is derived via Kirchoff's laws. Another example is the use of Newton's laws to derive a specific force by means of the force equilibrium for an object at rest. In practice, these models often contain various physics-inspired parameters $\mathbf{p}$ such as masses, friction constants and geometrical quantities. Most of these parameters are directly determined by dedicated experiments (e.g., measuring a length, weighting a mass). However, many parameters in $\mathbf{p}$ cannot be measured directly (e.g., friction constant). System identification techniques are concerned with identifying the parameter values based on data. More specifically, they aim to find the optimal value $\mathbf{p}^*$ so that an optimal fit between the measurements $\mathbf{y}$ and model output $\hat{\mathbf{y}} = \mathcal{M}(\mathbf{u};\mathbf{p})$ is obtained. Mathematically, this comes down to defining a cost function that penalizes the prediction error, averaged over all L output measurements $\{\mathbf{y}_1, \ldots, \mathbf{y}_L\}$.

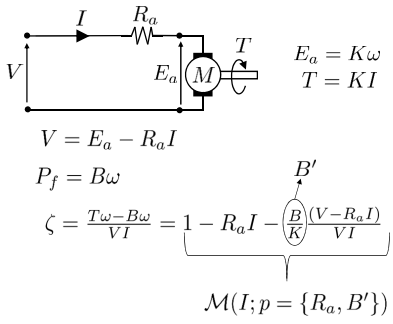
$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \frac{1}{L} \sum_{i=1}^{L} (\mathbf{y}_i - \mathcal{M}(\mathbf{u}_i;\mathbf{p}))^T M_c (\mathbf{y}_i - \mathcal{M}(\mathbf{u}_i;\mathbf{p}))) \tag{2.1}$$

The diagonal matrix M_c assigns a weighting factor to the deviation of each element in the measurement vector $\mathbf{y}$. This is essential if these elements have different orders of magnitude. Typically, the optimal values $\mathbf{p}^*$ can often not be determined analytically and, therefore, rely on numerical optimization solvers that iteratively update their best guess of $\mathbf{p}^*$. These numerical solvers can use gradient information to find local optima (e.g., gradient descent) [1] or metaheuristics (e.g., genetic algorithms and particle swarm) to obtain global solutions [2].

Below, a fictitious, yet intuitive, example of a static physics-inspired model is given. Assume the efficiency ζ of a separately excited DC motor operating in steady state is measured for different currents I and for a constant voltage $V = 200$V. The measured samples are shown by the blue dots in Fig. 2.4b. Figure 2.4a illustrates a simplified physical model relying on two physical parameters: the anchor resistance R_a and the scaled friction coefficient B'. Other physical quantities described in the model are the speed ω, the torque T, the counter electromotive force (EMF) E_a and the motor constant K. The optimized model in Fig. 2.4b definitely does not perfectly match the measurements. However, the general trend of the efficiency curve is captured. Note that although there are no measurements available for $I < 4$A and $I > 15$A ,

the model would still give decent prediction values if it would be evaluated in these unexplored areas.

(a) separately excited DC motor



(b) physics-inspired model

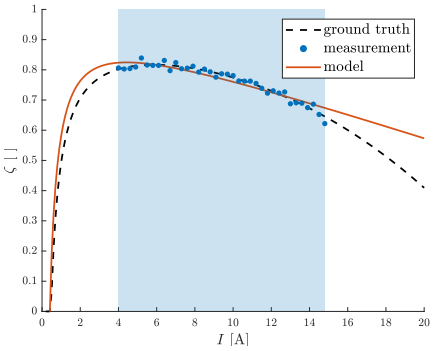


Figure 2.4: Intuitive example of optimizing the physical parameters of physics-based model.

2.3 Static System Models: Data-Driven

Alternatively, if no first principles are available, one can tackle the modeling challenge by considering a black-box approach. Static data-driven models do not make any assumptions on the explicit knowledge of the physical behavior of the system. These models are merely based on the analysis of the data obtained from the system. Traditional statistical techniques define a set of basis functions (e.g., linear functions, polynomials) for which a linear combination is performed to approximate the data. The model is defined via the coefficient corresponding to each basis function [3]. These mathematical relations do not imply any physical knowledge and are solely determined by the shape of the obtained data. However, more advanced techniques in the field of machine learning such as support vector machines, random forest and gradient boosting enable more flexible structures [3]. These methods are able to discover more complex patterns and, therefore, are promising techniques. However, they still require human engineered features as model input to obtain accurate prediction results. Deep neural networks alleviate the burden of the requirement of human designed features by having the ability to find intricate patters in raw data [4,5]. A full analysis of all data-driven techniques would be out of scope for this dissertation, therefore, we dive deeper into the basic concepts of these artificial neural networks, since they will be of major importance in the presented hybrid modeling approaches.

2.3.1 Feedforward Neural Networks (FFNN)

An artificial neural network is an interconnected group of nodes, inspired by a simplification of neurons in the brain. A node $\mathcal{N}j$ can be modeled as a mathematical operation that accepts an input vector $\mathbf{q}_j \in \mathbb{R}^{n_j}$ and performs some (nonlinear) mathematical operations leading to a one-dimensional node output $z_j \in \mathbb{R}$. A schematic representation of a neuron is shown in Fig. 2.5. For notational convenience, the i-th element of $\mathbf{q}_j$ is denoted as $q_j(i)$. Each input $q_j(i)$ is the output of a node of a prior layer within the neural network. The mathematical operations performed in node $\mathcal{N}_j$ are defined by the weights in $\boldsymbol{w}_j \in \mathbb{R}^{n_j}$ and the node bias $b_j \in \mathbb{R}$ that perform an affine transformation on the node input.

$$z_j = \mathcal{K}\left(b_j + \sum_{i=1}^{n_j} w_j(i) q_j(i)\right) \tag{2.2}$$

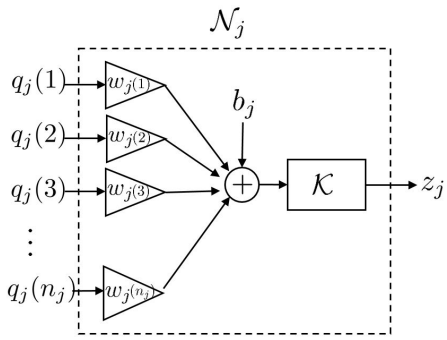


Figure 2.5: Schematic representation of a neuron.

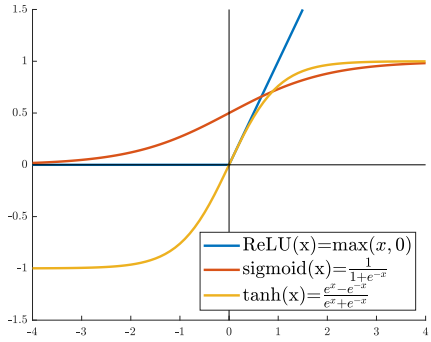


Figure 2.6: Activation functions $\mathcal{K}$.

This transformation is typically followed by a nonlinear operation defined by the activation function $\mathcal{K}$. This function can be any (piecewise) differential function. However, in practice, the used set is rather limited to a group of functions that have shown their benefits in various modeling tasks, such as ReLu, sigmoid and tanh [4, 6]. These activation functions $\mathcal{K}$, shown in Fig. 2.6, are the most frequently used ones as they bring parts of the input space to an (almost) constant output value. Therefore, they are remarkably well-suited to activate or deactivate parts of the input space, which invokes additional nonlinear properties to the overall model.

These neurons or nodes can be combined into various neural networks architectures. We refer to the literature for further details [4, 5]. We will focus on a fully connected feedforward neural network (FFNN) model η that accepts an input $\mathbf{u} \in \mathbb{R}^{n_i}$ and performs a succession of nonlinear operations in each

node resulting in the model output $\hat{\mathbf{y}} \in \mathbb{R}^{n_o}$. This architecture consist of three types of layers as is shown in Fig. 2.7.

- The input layer assigns a node to each element of the input vector $\mathbf{u}$. This layer solely accepts the model input and does not perform any further calculations. Consequently, the inputs are simply passed on to the hidden layer.
- The hidden layer(s) consists of n_h nodes for which each node $N_{j,h}$ accepts all outputs coming from the nodes of the previous layer. This implies that the first hidden layer will receive the outputs coming from the input layer, whereas each subsequent hidden layer will accept all outputs of the prior hidden layer. In practice, each layer can have a different number n_h of hidden neurons which is a typical design parameter depending on the complexity of the problem.
- The output layer assigns a neuron to each element of the target vector $\mathbf{y}$ and performs the last mathematical operations of the neural network. Every node in the output layer accepts all outputs coming from the last hidden layer and outputs the corresponding element of the prediction vector $\hat{\mathbf{y}}$.

The amount of neurons in the input and output layer depend, respectively, on the dimension of the input $\mathbf{u}$ and output $\mathbf{y}$ of the neural network η. The amount of hidden layers and the number of neurons within each of these layers can be freely chosen. In practice, this is a design parameter, often referred to as hyperparameter, which should be optimized depending on the complexity of the system [7].

Once the topology is determined, the neural network model η needs to be trained to match the data. The measured target data is noted by $\mathbf{y}$ and the model output is defined by $\hat{\mathbf{y}} = \eta(\mathbf{u})$. Each node $\mathcal{N}_j$ in the output layer and hidden layers consist of a weight vector $\mathbf{w}_j$ and a bias b_j. The aggregation of all parameters of all nodes of the neural network model η is denoted by $\boldsymbol{\alpha} \in \mathbb{R}^{n_\alpha}$. In analogy with (2.1), we define a cost function E as the averaged quadratic error of all L measurement samples.

$$E = \frac{1}{L} \sum_{i=1}^{L} (\mathbf{y}_i - \eta(\mathbf{u}_i; \boldsymbol{\alpha}))^T (\mathbf{y}_i - \eta(\mathbf{u}_i; \boldsymbol{\alpha}))) \tag{2.3}$$

Compared to (2.1), the matrix M_c is typically not included in this equation due to the assumption of having normalized inputs $\mathbf{u}$ and outputs $\mathbf{y}$. Note that the dimension of $\boldsymbol{\alpha}$ is typically a few magnitudes higher compared to the physical parameter vector $\mathbf{p}$ of the physics model. Therefore, recent techniques such as

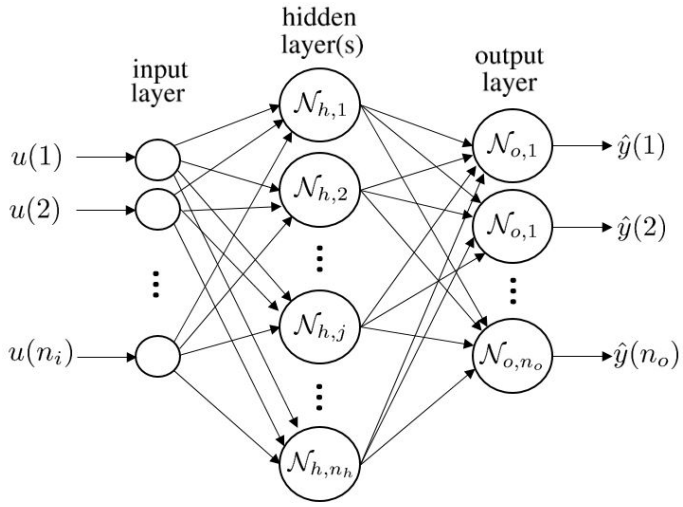


Figure 2.7: Schematic a feedforward fully connected neural network η.

backpropagation (BP) are essential to make optimization algorithms feasible within the currently available computational resources. The BP algorithm employs an analytical solution of the gradient of the error E with respect to the trainable variables in $\boldsymbol{\alpha}$.

$$\nabla_{\boldsymbol{\alpha}} E = \left[\frac{\partial E}{\partial \alpha_1} \quad \cdots \quad \frac{\partial E}{\partial \alpha_j} \quad \cdots \quad \frac{\partial E}{\partial \alpha_{n_\alpha}} \right]^T \tag{2.4}$$

This methodology, based on the chain rule of differentiation, propagates the error through all layers in an efficient manner allowing parallelization of the computing tasks [4]. The gradient is evaluated during each iteration i to update each network parameter α_j by

$$\alpha_j^{i+1} = \alpha_j^i + \gamma_j \left. \frac{\partial E}{\partial \alpha_j} \right|_{\alpha_j = \alpha_j^i} \tag{2.5}$$

In practice, the presented iterative update scheme of $\boldsymbol{\alpha}$ needs some further adaptations to make it feasible and efficient for practical implementation.

- Batch size: The cost function in (2.3) describes the average prediction error of all L available training samples. This implies that the analytical gradient should be evaluated for all samples during each update. In practice, for large datasets, this leads to an excessive computational burden which makes the training process infeasible. Alternatively, the stochastic gradient descent (SGD) incorporates only one sample per update. This drastically reduces the computation time per update but causes a lot of fluctuations and additional variance to the optimization process. Therefore, mini-batch gradient descent offers a trade-off of both approaches

by taking a subset of the full training data during each update. Consequently, we define an epoch as the number of times the algorithm went through the entire dataset. The number of iterations per epoch directly relies on the size of the mini batches, which is an additional degree of freedom (hyperparameter), chosen by the user.

- Learning rate γ: The learning rate γ determines the change of $\boldsymbol{\alpha}$ during each update. Low learning rates only allow a limited change to the parameters in $\boldsymbol{\alpha}$ and, therefore, lead to slow convergence. In contrast, too large learning rates can lead to excessive updates which lead to fluctuations around the local minima or even divergence of the optimization process. In the past, learning rate schedulers were commonly used to prevent these problems. These schedulers initiate the optimization by imposing large learning rates to perform significant updates toward the region of the (local) minimum. Thereafter, the learning rate is gradually decreased to obtain smooth convergence towards the minimum. Although this brings some improvements, the learning rate is determined by a predefined scheme and does not incorporate any feedback of the effectiveness of the performed updates. The algorithm can, therefore, get stuck in a local minimum. In addition, the same learning rate is used for each parameter in $\boldsymbol{\alpha}$, without incorporating the sensitivity of each element to the model output. Consequently, various new methodologies that enable adaptive learning rate schedules that can cope with the aforementioned issues have been developed. Some examples are SGD with momentum, Adagrad and Adam [8].

In practice, the burden of implementing all these intricate algorithms is alleviated by the availability of advanced toolboxes such as Tensorflow [9] and Keras [10]. These libraries employ automatic differentiation so that analytical gradients are automatically derived for the given network topology. The user should only define the network architecture and consequently choose adequate values for hyperparameters such as the number of neurons, number of layers, learning rate and batch size. A common approach is to perform a hyperparameter optimization loop on top of the actual training of the model parameters $\boldsymbol{\alpha}$. The number of hidden layers and consequently the amount of neurons per layer are crucial parameters since they determine the flexibility of the network. A model that contains a limited amount of neurons is restricted by a small set of trainable parameters in $\boldsymbol{\alpha}$. The corresponding model often lacks flexibility to fit the data which leads to an underfit as illustrated in Fig. 2.8a. Defining an excessive amount of neurons within the model increases the predictive capabilities of the model. However, it can also increase the chance of overfitting the data. The error on the training set is driven to a very small value, but

when new data is presented to the network, a large prediction error is obtained. Figure 2.8b illustrates that the network has memorized the training examples, but it has not learned a generalized model that copes well with new situations. Ideally, a perfect balance is obtained between providing sufficient and predictive capabilities while still being accurate for unseen data samples, as shown in Fig. 2.8c. As these black-box models often lack interpretation, the choice of the hyperparameters can be non-intuitive.

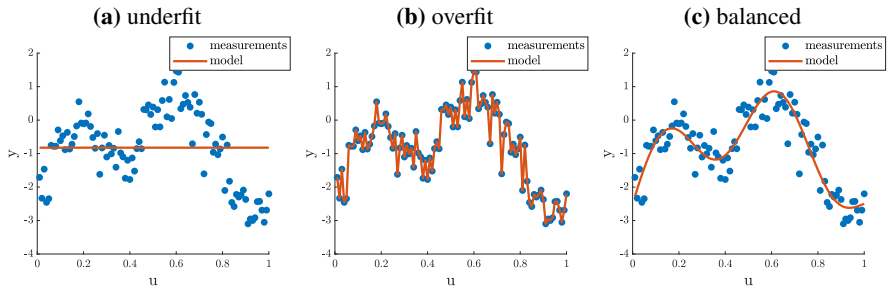


Figure 2.8: Trade-off between allowing using sufficient flexibility of the neural network and avoiding an overfit.

Nevertheless, in practice, it is not always that easy to identify the "right" amount of neurons for the considered regression task. A modern approach is to use an excessive model and add an additional L_2 regularization term on all n_w weights in the cost function. This additional regularization term will enforce smaller values to the weights of the neural network. The parameter λ is an additional hyperparameter that requires tuning.

$$E = \frac{1}{L}\sum_{i=1}^{L}(\mathbf{y}_i - \eta(\mathbf{u}_i;\boldsymbol{\alpha}))^T(\mathbf{y}_i - \eta(\mathbf{u}_i;\boldsymbol{\alpha}))) + \frac{\lambda}{2n_w}\sum_{i=1}^{n_w} w_i^2 \tag{2.6}$$

Note that these network weights w_i scale the different inputs to each node before entering the activation function $\mathcal{K}$, as shown in Fig. 2.5. By having smaller weights, the effect of the activation function decreases. This results into a less complex function that is fitted to the data, effectively reducing overfitting phenomena. Alternatively, however less used, one can apply an L_1 norm by taking the absolute value of the weights instead of squaring them. Unlike the L_2 norm, the weights may be reduced to zero, which can be useful when one tries to compress the model. In addition, dropout regularization techniques, that randomly temporally remove nodes during the training process, can be used. Consequently, this means that each node cannot fully rely on the inputs since there is a random probability that they can be suddenly removed. Therefore, the neural network will be reluctant to give high weights to certain features, because they might disappear. Consequently, the weights are spread across all

features, resulting in smaller weight values. Hence, obtaining a less complex model.

The predictive performance of a neural network is illustrated on the same static dataset as in Fig. 2.4. The network $\eta(I;\boldsymbol{\alpha})$ is trained to model the efficiency ζ of a DC motor for given anchor current I. Compared to the physical model shown in Fig. 2.4b, the neural network model can better predict the true curve within the training region. However, once the model is evaluated beyond the explored region (i.e $I < 4\text{A}$ or $I > 15\text{A}$), the model completely fails to predict the motor efficiency. This illustrates the major drawback of data-driven models. These models will be very accurate within the region for which they are trained, but will behave poorly once evaluated in regions of unseen data. In other words, they are very well in interpolating data but poorly in extrapolating the behavior of the system.

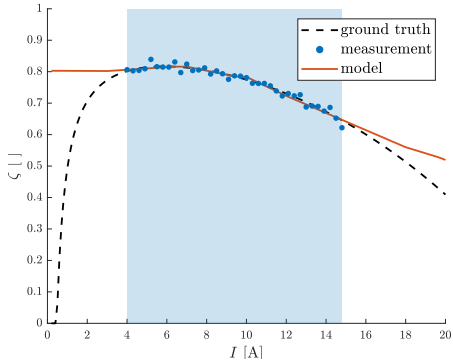


Figure 2.9: Intuitive example of training a neural network model.

2.4 Dynamic System Models: Physics-Inspired

When studying mechatronic applications we typically require models that can cope with the ingrained dynamics of these complex systems. Hence, these models need to possess the capability of describing the nonlinear system dynamics. Traditionally, models of dynamic systems were based on first-principles. Fundamental physical laws such as *Newton's laws* and *conservation of energy* can be used to describe the behavior of a machine component in temporal and spatial domain by constructing partial differential equations (PDE) [11]. In this research, we are more interested in the dynamic interactions of various machine components rather than the component itself. Hence, we omit the spatial domain and focus on how all mechanisms behave in time. Therefore, we make a simplification of the system by assigning lumped physical parameters to moving entities such as masses and inertia. Fig. 2.10 gives an illustration of how a piston mechanism is simplified by considering

weightless links that geometrically determine the linear displacement of a mass.

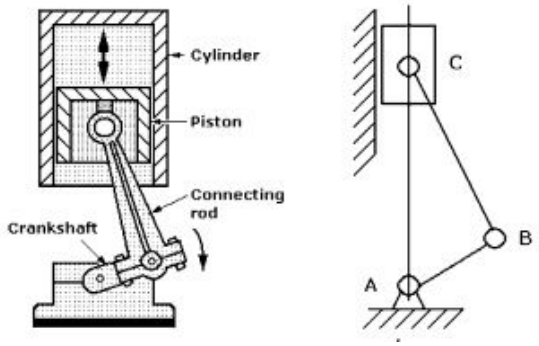


Figure 2.10: Simplification of a piston mechanism. Retrieved from [12].

The dynamic behavior of these lumped models can be captured by distilling the corresponding ordinary differential equations (ODE) [13]. Hence, we define a state $\mathbf{x}$ that is considered as the minimal set of internal time varying properties (e.g., angles, speed) required to explain the behavior of the system in time for given external inputs $\mathbf{u}$ (e.g., control, voltage). Furthermore, we assume a time invariant system, implying that the system dynamics and consequently the mathematical relations in f and g are independent of time.

$$\begin{cases} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t); \mathbf{p}) \\ \mathbf{y}(t) &= g(\mathbf{x}(t), \mathbf{u}(t); \mathbf{p}) \end{cases} \tag{2.7}$$

The relations captured in f can be used to integrate the state behavior over time (i.e., $\hat{\mathbf{x}}$), while the function g serves as a mapping that constructs the estimations of the measurements $\hat{\mathbf{y}}$ from the state simulations $\hat{\mathbf{x}}$, as shown in Fig. 2.11.

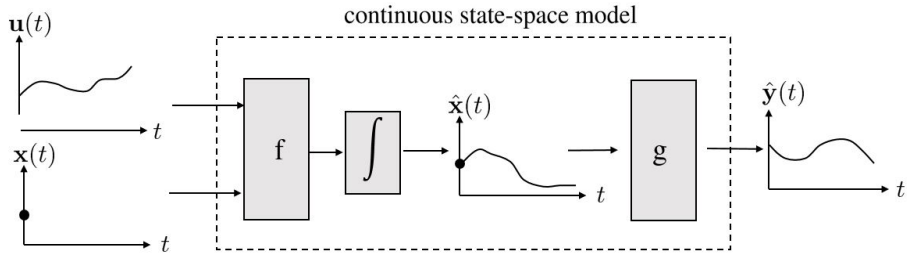


Figure 2.11: Schematic representation of a state-space model.

Ideally, we can use the continuous dynamic model to derive the exact expression for $\mathbf{y}(t)$, starting from an initial state $\mathbf{x}_0$ and given external input signal $\mathbf{u}(t)$. However, for complex nonlinear models it is nearly impossible to solve this ODE exactly, which forces us to rely on numerical approximation techniques. Therefore, we will only consider a sequence of discrete external inputs $\{\mathbf{u}(t_1), \cdots, \mathbf{u}(t_k)\}$ evaluated at a fixed time interval Δt. For notational convenience we denote the index k for values evaluated at time interval t_k. We can consequently approximate the update scheme by applying a forward Euler method (FE).

$$\begin{cases} \hat{\mathbf{x}}_{k+1} & \approx \underbrace{\mathbf{x}_k + f(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p})\Delta t}_{\mathcal{M}(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p})} \\ \hat{\mathbf{y}}_k & = g(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p}) \end{cases} \tag{2.8}$$

More elaborate integration techniques such as Runge-Kutta exist [14], but since we assume sufficiently small sampling time intervals Δt with respect to the dynamics of the system, the use of a standard solver such as FE is justified. The system model $\mathcal{M}$ described in (2.8) approximates a state propagation for a given input $\mathbf{u}_k$ and state $\mathbf{x}_k$. Consequently, a trajectory prediction of the state is obtained by subsequently using the state predictions of prior time steps as input to the model $\mathcal{M}$, as shown in Fig. 2.12.

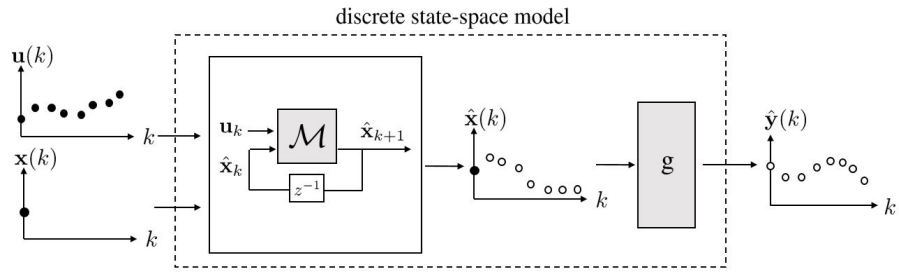


Figure 2.12: Schematic representation of a discrete state-space model used to simulate the system behavior.

.

The first challenge when constructing these physics-inspired models is to translate all observed interactions into first-principles. Second, once the governing equations (i.e., f and g) are determined, the corresponding model parameters should be identified. Many parameters can be directly measured (e.g., mass, length), while other variables are less straightforward to determine (e.g., friction constant, hydraulic properties). Parameter identification techniques can then be used to identify the values of these parameters based on experimental sensor data by aligning the model response with the data [15]. Uncertainties can however still remain as complex machines face nonlinearities such as friction forces that give rise to disturbances that are difficult to model [16].

2.5 Dynamic System Models: Data-Driven

In practice, it is very difficult or even impossible to describe all phenomena within complex machines by physical laws because there are always uncertain system interactions that are unnoticed while constructing physics-inspired dynamic models. Alternatively, black-box, data-driven models can be built that can capture the dynamics directly from the data.

2.5.1 Data-Driven Timeseries Models

The dynamics of a system model can be learned by various time series models that process sequences of data points collected over a time interval. Implementations of nonlinear autoregressive exogenous inputs (NARX) models have shown their value in learning the behavior of dynamic systems [17–20]. Assume we endeavor to capture the dynamics of a one-dimensional measurable variable y, which is controlled by a control input $\mathbf{u}$, the NARX model describes the behavior as

$$\hat{y}_k = F(y_{k-1}, \ldots, y_{k-N_y}, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-N_u}; \boldsymbol{\alpha}) \tag{2.9}$$

The function F is typically a polynomial or a static neural network such as described in Section 2.3.1, that processes a tapped delay line of the input sequences of $\mathbf{u}$ and prior measurements y. Figure 2.13 illustrates the delay operators, indicated by z^{-1}, to construct the input of the function F. The values N_y and N_u are the maximum lags for respectively the system output and input variables. Note that in this research, due to causality reasons, we will always assume that the output y_k is not influenced by the momentaneous control input $\mathbf{u}_k$. Consequently, the NARX model can be used to simulate the system dynamics by subsequently using the model predictions $\{\hat{y}_{k-i}\}_{i=1}^{i=N_y}$, obtained during previous iterations, to replace the actual measurements $\{y_{k-i}\}_{i=1}^{i=N_y}$. Note that within this framework of autoregressive models many variants (e.g., ARX, NARMAX) exist [21]. Furthermore, the modeling capabilities highly rely on the chosen number of lags (i.e, N_y and N_u). From a theoretical point of view, Takens' embedding theorem determines the number of lags required to capture the system dynamics [22]. However, in practice this is typically a hyperparameter that should be empirically tuned according to the system complexity [23].

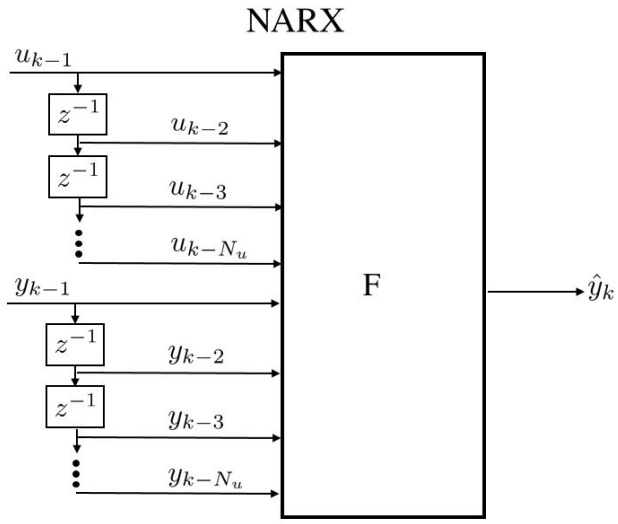


Figure 2.13: Schematic overview of NARX model.

For more complex systems that require many information over long time sequences it becomes cumbersome to reprocess all prior predictions $\{\hat{y}_{k-i}\}_{i=1}^{i=N_y}$ and inputs $\{\mathbf{u}_{k-i}\}_{i=1}^{i=N_u}$ during each prediction step. It is therefore often preferred to compress al prior information until a time instance k within a state $\mathbf{h}_k$. Consequently, this state is updated for each new external input $\mathbf{u}_k$, as illustrated in Fig. 2.14. Mathematically, this can be described by following relations

$$\begin{aligned} \mathbf{h}_{k+1} &= \mathcal{W}_h(\mathbf{u}_k, \mathbf{h}_k; \boldsymbol{\alpha}) \\ \hat{\mathbf{y}}_k &= \mathcal{W}_y(\mathbf{h}_k; \boldsymbol{\alpha}) \end{aligned} \tag{2.10}$$

Note that compared to the state $\mathbf{x}$ of the physics-inspired state-space representation in 2.7, the state $\mathbf{h}$ does not have to represent any physical interpretation. Moreover, an infinite amount of solutions exist since any valid coordinate transformation yields another state-space realization [24]. However, in practice, we aim for the minimal state-space representation, which describes the system by a minimal amount of state variables. If the function $\mathcal{W}_h$ and $\mathcal{W}_y$ are linear, the system identification can be obtained by analyzing the frequency response [25] or using more advanced techniques such as the Ho-Kalman procedure to determine the Markov parameters of the system [26]. However, the intricate dynamics induced in mechatronic applications typically require more flexible modeling architectures that can model nonlinear dynamics [27]. Hence, polynomial terms can be added to linear state space models to induce the required flexibility [28, 29]. Furthermore, Hammerstein-Wiener structures typically combine linear system models with nonlinear mappings to introduce the required non-linearity to the model [30].

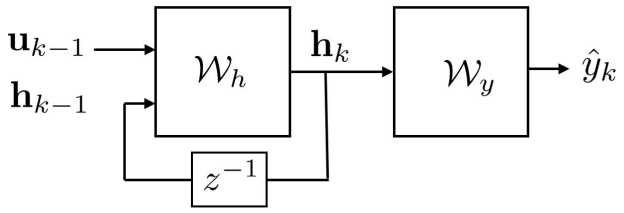


Figure 2.14: Schematic overview of data-driven model for which the information of prior time instance is absorbed in the state $\mathbf{h}$.

2.5.2 Recurrent Neural Networks (RNN)

For highly nonlinear relations, the function $\mathcal{W}_h$ and $\mathcal{W}_y$ in (2.10) can be represented by neural layers, resulting into a so-called recurrent neural network (RNN). The use of these RNN is of particular interest since various research results have shown their capability to learn complex system dynamics starting from experimental data of mechatronic applications [31–33]. In general, a RNN accepts an input sequence for which each input is processed in a sequential manner. The relevant information of prior inputs is stored into the internal state $\mathbf{h}$, which is updated during each time step. Consequently, these RNN models exhibit internal dynamics and, therefore, are highly suited to capture the behavior of dynamic systems, as is extensively demonstrated in the literature [34–39]. A schematic representation of a RNN modeling structure is presented in Fig. 2.15. The RNN model accepts an input sequence of N steps by processing each input element $\mathbf{u}_k$ at a time. The network parameters $\boldsymbol{\alpha}$ are optimized to minimize the prediction error between the measured and predicted trajectories of the training set. The optimal values of $\boldsymbol{\alpha}$ are updated in an iterative manner, based on the update rule described in (2.5). This step exploits the analytical gradient of the loss with respect to the parameters in $\boldsymbol{\alpha}$. This analytical expression of the gradient is derived by a dedicated technique, often referred to as backpropagation through time (BPTT) [40]. This methodology approaches the RNN structure as a FFNN by unfolding the neural network, as illustrated in Fig. 2.15. Consequently, the virtually-static unfolded network contains N inputs and outputs. Furthermore, each copy of the network shares the same parameters, defined in $\boldsymbol{\alpha}$. The BPTT comes down to applying the traditional BP on this unfolded static network to determine the analytical gradient of the loss function.

However, in practice, this approach is often plagued by some numerical inconvenience which make the converge of the parameters $\boldsymbol{\alpha}$ a cumbersome task [41]. Applying the BP algorithm to the unfolded network requires the gradient to be derived by moving layer by layer, starting from the final layer

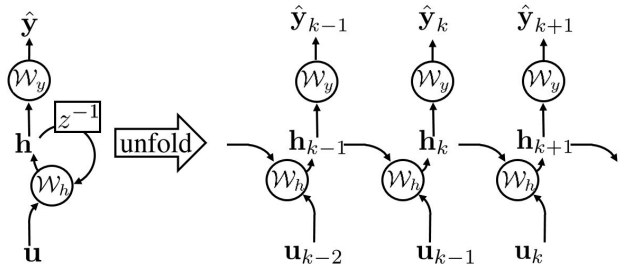


Figure 2.15: Schematic overview of a recurrent neural network.

to the initial layer. As explained in Section 2.3.1, this can be substantiated by the chain rule of differentiation. This implicates that the derivatives of the first layers are multiplied with the derivatives of further layers in the model. Therefore, if small derivatives are obtained, the gradient will become exponentially smaller for the first layers. Consequently, the update step in (2.5) becomes negligible, which makes these networks barely trainable. This phenomenon is typically called a vanishing gradient problem. By contrast, derivative values larger than 1 lead to exponentially growing gradients in the first layers. This is typically referred to as an exploding gradient. The amount of layers in the unfolded RNN equals the amount of time steps. Therefore, using these vanilla RNN on long time sequences make them susceptible for these phenomena. Consequently, more advanced RNN architectures such as LSTM networks, which do not treat each time step equally, have been developed [42]. The exact details about the LSTM network are explained in Chapter 5, but the main idea is that the network contains three gate components that regulate the influence of the new information on the network states. Consequently, these network architectures can learn to distill important events in time series by selectively remembering patterns for a long duration of time, making them more suitable to learn the behavior of nonlinear systems [33, 43].

2.5.3 Physics-Inspired Architectures for Data-Driven Dynamic System Models

In general, the internal state $\mathbf{h}$ does not represent any physically interpretable parameter, making this feature a network property for which the corresponding dimensions are typically considered as a hyperparameter. However, a major challenge arises in determining a proper initialization strategy for the initial state $\mathbf{h}_0$. This choice will directly influence the subsequent predictions according to (2.10). In practice, the initial state is typically chosen fixed. This approach is substantiated by the idea that, for an asymptotically stable system, the influence of the initial state on the output will decrease over time [44]. Al-

ternatively, the initial state $\mathbf{h}_0$ can be considered as an optimization variable that is updated together with the model parameters in $\boldsymbol{\alpha}$ during the learning process [45]. Furthermore, in case the predictions do not start from a steady state, and thus cannot have the same value for $\mathbf{h}_0$, a neural network can be used to initialize the value of $\mathbf{h}_0$ based on preceding measurements [46]. In general, a more structural solution can be obtained if physical insights are available to deduce a proper structure of the black-box model. For instance, the state $\mathbf{h}$ can be chosen equal to the physically interpretable system state $\mathbf{x}$ [47, 48]. Furthermore, a feedforward neural network can used to approximate the ODE of dynamical systems, i.e. f in (2.7) , to return more interpretable modeling architectures [49, 50]. This can be solved via numerical solvers, as mentioned in Section 2.4. Novel neural-ODE techniques approach the dynamics from a continuous perspective, making them suitable for time series with irregular sampling time intervals [51]. Note that the derivative function can also be represented by non-parametric machine learning models such as gaussian processes [52,53]. Next, recent approaches based on symbolic regression [54] and sparse regression [55, 56] were able to return interpretable models by discovering the underlying governing equations. Furthermore, recent developments indicate the possibility to approximate reformulations of classical mechanics such as Lagrangian [57] and Hamiltonian [58] mechanics by neural networks.

2.6 Dynamic System Models: Hybrid

2.6.1 The Need for Hybrid Approaches

In general, physics-inspired and data-driven system identification approaches are traditionally being used next to each other. The physics-inspired models have been widely used due to their interpretable structure and their robust behavior. The increasing growth of computational resources and the availability of high-performing methodologies boosted the introduction of powerful data-driven approaches. These algorithms learn solely from the data and thus, do not require prior system knowledge. Nevertheless, dynamic system models will always be plagued by drift, once evaluated for longer time series. These drift phenomena are caused by subsequently using the estimated information of prior time steps as model input to predict the output on next time instances. Prediction errors, however small, will always be propagated through subsequent time steps so that the deviation between the measured and predicted trajectory increases. This insurmountable shortcoming does not make these models worthless, because as long the general trends are captured, the model can be valuable to predict the overall dynamics of the system. Figure 2.16 and 2.17 demonstrate the general properties associated with dynamic physics and data-driven models, respectively. The gray area in the plots on the left hand

side indicate the explored state-space regions during training. The plots on the right indicate a time series forecast of an unseen trajectory, respectively indicated by the orange and blue line. We compared them with respect to the training region. First, the performance within the training region, is discussed. Thereafter, the properties when exceeding the training region are elaborated.

- **Evaluation within training region**: In general, physics-inspired dynamic models have less flexibility to match the data because they are restricted by the physical laws that define the model structure. It is, however, impossible to include every phenomenon within the physical model so that, although the general dynamics will be captured by the model, the predicted trajectory will deviate from the actual system measurements (see. Fig. 2.16). Data-driven models allow much more flexibility and predictive capabilities in their modeling structures. The model can, therefore, better predict the dynamics in regions for which data is explored. Figure 2.17 illustrates accurate prediction performances of time series for data-driven models as long they are included in regions that are well explored during the training phase.

- **Evaluation outside training region**: The reduced flexibility and interpretable structure ingrained in physics-inspired models make them resilient to evaluations in neighborhoods outside the training region (used for the system identification step). Although the model is not trained in these regions, the model will not violate against physical fundamentals such as conservation of energy. For the sake of completeness, we should mention that for some systems singularities can occur at particular operating regimes, so that the system dynamics cannot be described by the same physical laws. But in general, it appears that for most systems the underlying physics remain consistent so that adequate predictions can still be obtained once the time series prediction propagates towards unexplored areas in the state space region, as shown in Fig. 2.16. In contrast, this is not the case for data-driven models that are trained to fit given data. If the models are evaluated in regions for which they are not trained, the model has unlimited freedom and the dynamic model can become unstable, as shown in Fig. 2.17.

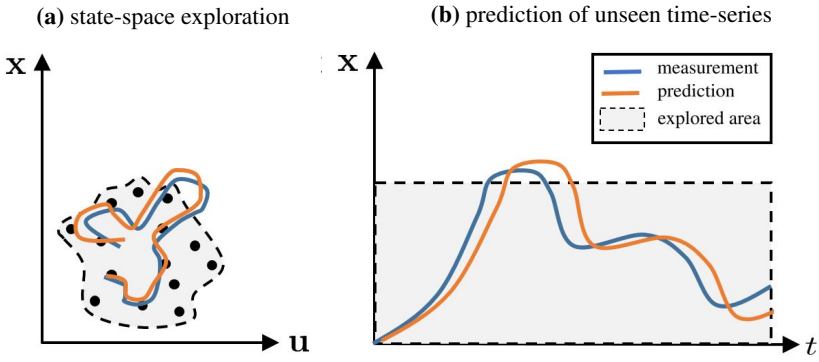


Figure 2.16: Forward propagation of physics-inspired dynamic models. These models exhibit adequate prediction performance for the region for which they are trained and still provide robust dynamics once the trajectory exceeds the training region.

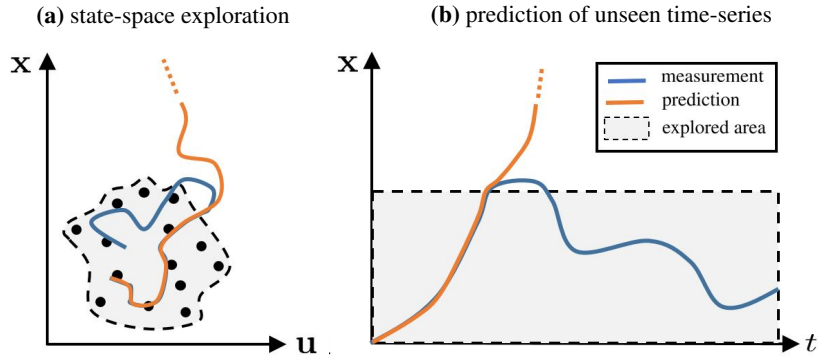


Figure 2.17: Forward propagation of black-box dynamic models. These models exhibit accurate prediction performance for the region for which they are trained, but fail once the trajectory exceeds the training region.

2.6.2 Current Situation

It is clear that both modeling types have their advantages and disadvantages. Therefore, recent research focuses on combining the two techniques to develop new modeling formalisms that combine the strengths of both techniques. A first approach to obtain more reliable data-driven models is to augment the measurement data with synthetic data obtained by emulating situations on physics models [59]. Alternatively, one can directly use physics-based governing equations within the data-driven model. These so-called grey-box models traditionally rely on either complementing a white-box model with a black-

box model, or vice versa. Physical laws can be included in the loss function of black-box neural networks to guide the training process towards physically consistent model predictions [60]. On the contrary, neural networks have been used to compensate for prediction discrepancies of a physics-based model. The neural network can either serve as a nonlinear mapping [33, 61] or as a residual term [62–65] that compensates for the modeling discrepancies. Note that this framework has also been introduced with non-parameteric models such as gaussian processes [66]. Furthermore, neural networks have been used directly in incomplete physics models by replacing the unknown interactions in the physics-inspired model [67, 68]. These researches have shown promising results in combining the advantages of physics-based and data-driven modeling approaches. Based on these findings, we studied various hybrid approaches for predicting the nonlinear dynamics in mechatronic applications, aiming for accurate, robust and explainable modeling architectures that are experimentally validated on various mechatronic setups. Our research results are presented in Chapters 3-6.

References

[1] J. Nocedal and S. Wright. Numerical optimization. Springer Science & Business Media, 2006.

[2] G. Lindfield and J. Penny. Introduction to nature-inspired optimization. Academic Press, 2017.

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani. An introduction to statistical learning, volume 112. Springer, 2013.

[4] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. MIT press, 2016.

[5] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[6] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378, 2018.

[7] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. The Journal of Machine Learning Research, 13(1):281–305, 2012.

[8] S. Ruder. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747, 2016.

[9] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symp. on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016.

[10] F. Chollet et al. Keras. `https://keras.io`, 2015.

[11] J. D. Logan. Applied Partial Differential Equations. Undergraduate Texts in Mathematics. Springer International Publishing, 2014.

[12] N. Sclater and Nicholas P. Chironis. Mechanisms and mechanical devices sourcebook, volume 3. McGraw-Hill New York, 2001.

[13] L. Perko. Differential Equations and Dynamical Systems. Texts in Applied Mathematics. Springer New York, 2013.

[14] J. C. Butcher. Numerical methods for ordinary differential equations. John Wiley & Sons, 2016.

[15] T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. Automatica, 47(1):39–49, 2011.

[16] X. Wang, W. Wang, L. Li, J. Shi, and B. Xie. Adaptive control of dc motor servo system with application to vehicle active steering. IEEE/ASME Transactions on Mechatronics, 2019.

[17] H. Asgari, X. Chen, M. Morini, M. Pinelli, R. Sainudiin, P. R. Spina, and M. Venturini. Narx models for simulation of the start-up operation of a single-shaft gas turbine. Applied Thermal Engineering, 93:368–376, 2016.

[18] M. Basso, L. Giarre, S. Groppi, and G. Zappa. Narx models of an industrial power plant gas turbine. IEEE Transactions on control systems technology, 13(4):599–604, 2005.

[19] I. B. Tijani, R. Akmeliawati, A. Legowo, and A. Budiyono. Nonlinear identification of a small scale unmanned helicopter using optimized narx network with multiobjective differential evolution. Engineering Applications of Artificial Intelligence, 33:99–115, 2014.

[20] M. V. Kumar, S. N. Omkar, R. Ganguli, P. Sampath, and S. Suresh. Identification of helicopter dynamics using recurrent neural networks and flight data. Journal of the American Helicopter Society, 51(2):164–174, 2006.

[21] S. A. Billings. Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. John Wiley & Sons, 2013.

[22] F. Takens. Detecting strange attractors in turbulence. In Dynamical systems and turbulence, Warwick 1980, pages 366–381. Springer, 1981.

[23] T. Lin, B. G. Horne, C. L. Giles, and S.-Y. Kung. What to remember: How memory order affects the performance of narx neural networks. In 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227), volume 2, pages 1051–1056. IEEE, 1998.

[24] R. L. Williams and D. A. Lawrence. Linear state-space control systems. John Wiley & Sons, 2007.

[25] R. Pintelon and J. Schoukens. System identification: a frequency domain approach. John Wiley & Sons, 2012.

[26] S. Oymak and N. Ozay. Non-asymptotic identification of lti systems from a single trajectory. In 2019 American control conference (ACC), pages 5655–5661. IEEE, 2019.

[27] J. Schoukens and L. Ljung. Nonlinear system identification: A user-oriented road map. IEEE Control Systems Magazine, 39(6):28–99, 2019.

[28] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon. Identification of nonlinear systems using polynomial nonlinear state space models. Automatica, 46(4):647–656, 2010.

[29] J.-P. Noël, A. F. Esfahani, G. Kerschen, and J. Schoukens. A nonlinear state-space approach to hysteresis identification. Mechanical Systems and Signal Processing, 84:171–184, 2017.

[30] T. B. Wills, A.and Schön, L. Ljung, and B. Ninness. Identification of hammerstein–wiener models. Automatica, 49(1):70–81, 2013.

[31] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans. Nonlinear systems identification using deep dynamic neural networks. arXiv preprint arXiv:1610.01439, 2016.

[32] N. Mohajerin and S. L. Waslander. Modular deep recurrent neural network: Application to quadrotors. In IEEE International Conference on Systems, Man, and Cybernetics (SMC 2014), pages 1374–1379. IEEE, 2014.

[33] N. Mohajerin and S. L. Waslander. Multistep prediction of dynamic systems with recurrent neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2019.

[34] A. G. Parlos, K. T. Chong, and A. F. Atiya. Application of the recurrent multilayer perceptron in modeling complex process dynamics. IEEE Transactions on Neural Networks, 5(2):255–266, 1994.

[35] S. Li. Wind power prediction using recurrent multilayer perceptron neural networks. In 2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No. 03CH37491), volume 4, pages 2325–2330. IEEE, 2003.

[36] D. T. Pham and X. Liu. Identification of linear and nonlinear dynamic systems using recurrent neural networks. Artificial Intelligence in Engineering, 8(1):67–75, 1993.

[37] S. N. Kumpati and P. Kannan. Identification and control of dynamical systems using neural networks. IEEE Transactions on neural networks, 1(1):4–27, 1990.

[38] J.-S. Wang and Y.-P. Chen. A fully automated recurrent neural network for unknown dynamic system identification and control. IEEE Transactions on Circuits and Systems I: Regular Papers, 53(6):1363–1372, 2006.

[39] K. Nouri, R. Dhaouadi, and N. B. Braiek. Identification of a nonlinear dynamic systems using recurrent multilayer neural networks. In IEEE International Conference on Systems, Man and Cybernetics, volume 5, pages 5–pp. IEEE, 2002.

[40] P. J. Werbos. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10):1550–1560, 1990.

[41] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In International Conference on Machine Learning, pages 1310–1318, 2013.

[42] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10):2222–2232, 2016.

[43] J. Gonzalez and W. Yu. Non-linear system modeling using lstm neural networks. IFAC-PapersOnLine, 51(13):485–489, 2018.

[44] H. Jaeger. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach, volume 5. GMD-Forschungszentrum Informationstechnik Bonn, 2002.

[45] V. M. Becerra, J. M. F. Calado, P. M. Silva, and F. Garces. System identification using dynamic neural networks: training and initialization aspects. IFAC Proceedings Volumes, 35(1):235–240, 2002.

[46] N. Mohajerin and S. L. Waslander. State initialization for recurrent neural network modeling of time-series data. In International Joint Conference on Neural Networks (IJCNN 2017), pages 2330–2337. IEEE, 2017.

[47] A. H. Ribeiro, K. Tiels, J. Umenberger, T. B. Schön, and L. A. Aguirre. On the smoothness of nonlinear system identification. Automatica, 121:109158, 2020.

[48] W. Yu. Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms. Information sciences, 158:131–147, 2004.

[49] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. arXiv preprint arXiv:1801.01236, 2018.

[50] X. Li and W. Yu. Dynamic system identification via recurrent multilayer perceptrons. Information sciences, 147(1-4):45–63, 2002.

[51] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In Advances in neural information processing systems, pages 6571–6583, 2018.

[52] S. Eleftheriadis, T. Nicholson, M. P. Deisenroth, and J. Hensman. Identification of gaussian process state space models. In Advances in Neural Information Processing Systems, pages 5309–5319, 2017.

[53] R. Frigola, Y. Chen, and C. E. Rasmussen. Variational gaussian process state-space models. In Advances in neural information processing systems, pages 3680–3688, 2014.

[54] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. Science, 324(5923):81–85, 2009.

[55] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the national academy of sciences, page 201517384, 2016.

[56] S. Khatiry Goharoodi, K. Dekemele, M. Loccufier, L. Dupre, and G. Crevecoeur. Evolutionary-based sparse regression for the experimental identification of duffing oscillator. Mathematical Problems in Engineering, 2020, 2020.

[57] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. International Conference on Learning Representations, 2019., 2019.

[58] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In Advances in Neural Information Processing Systems, pages 15353–15363, 2019.

[59] C. Sobie, C. Freitas, and M. Nicolai. Simulation-driven machine learning: Bearing fault classification. Mechanical Systems and Signal Processing, 99:403–419, 2018.

[60] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In Proceedings of the 2019 SIAM International Conference on Data Mining, pages 558–566. SIAM, 2019.

[61] A. Ajay, M. Bauza, J. Wu, N. Fazeli, J. B. Tenenbaum, A. Rodriguez, and L. P. Kaelbling. Combining physical simulators and object-based networks for control. arXiv preprint arXiv:1904.06580, 2019.

[62] S. C. Stubberud, R. N. Lobbia, and M. Owen. An adaptive extended kalman filter using artificial neural networks. In Proceedings of 1995 34th IEEE Conference on Decision and Control, volume 2, pages 1852–1856. IEEE, 1995.

[63] V. L. Guen, Y. Yin, J. Dona, I. Ayed, E. de Bézenac, N. Thome, and P. Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. arXiv preprint arXiv:2010.04456, 2020.

[64] R. Rico-Martinez, J.S. Anderson, and I.G. Kevrekidis. Continuous-time nonlinear signal processing: a neural network based approach for gray box identification. In Proceedings of IEEE Workshop on Neural Networks for Signal Processing, pages 596–605. IEEE, 1994.

[65] M. L. Thompson and M. A. Kramer. Modeling chemical processes using prior knowledge and neural networks. AIChE Journal, 40(8):1328–1340, 1994.

[66] J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. Autonomous Robots, 27(1):75–90, 2009.

[67] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In IEEE International Conference on Robotics and Automation, pages 3223–3230. IEEE, 2015.

[68] D. C. Psichogios and L. H. Ungar. A hybrid neural network-first principles approach to process modeling. AIChE Journal, 38(10):1499–1511, 1992.

Chapter 3

Inverse Parametric Uncertainty Identification in Physics-Inspired Models

Physics-inspired models are still abundantly used due their explainable nature. However, it is often not possible to capture all interactions within intricate mechatronic applications starting from first principles. In wet friction clutches the so-called shifting time is difficult to predict at varying operating conditions. Non-modeled interactions hinder accurate shifting time estimations and leave a mismatch between the physically-modeled behavioral responses and experimental data, as conceptually shown in Fig. 2.16. To address this mismatch, we make an abstraction of the non-modeled interactions by introducing stochastic system parameters. These stochastic system parameters are then inversely identified starting from collected data. This chapter pinpoints the shortcomings of classical physics-inspired models. This way, we present the basic foundation of the hybrid modeling approaches presented in the next chapters. My contributions can be summarized as follows:

- An inverse uncertainty identification framework is applied on experimental data collected from a wet-friction clutch. The work is specifically concerned with identifying probability distributions of uncertain model parameters by optimally aligning the obtained output distributions with the measurements.

- The propagation of the parameter uncertainty towards an output uncertainty is typically obtained in a brute force manner. This requires many evaluations of a computationally expensive system model, ultimately hindering the identification of the parameter uncertainty distributions. To reduce the required number of model evaluations, I propose a methodology that incorporates the polynomial chaos framework, originating from the stochastic systems research field. Furthermore, I present

how higher-order statistical moments associated to the output distribution can be estimated. The research results show that this framework can reduce the required number of model evaluations by an order of magnitude. This way, I enabled to inversely identify parameter uncertainties within the physics-inspired model from the collected data.

Inverse Parametric Uncertainty Identification using Polynomial Chaos and High-Order Moment Matching Benchmarked on a Wet Friction Clutch

W. De Groote, T. Lefebvre, G. Tod, N. De Geeter, B. Depraetere, S. Van Poppel, and G. Crevecoeur

Published in "Mechatronics", 2020

Abstract *Inconsistent behavior of mechatronic applications is often related to uncertainties ingrained in the system. Stochastic models spawn an output distribution for any given deterministic input. A stochastic model can be obtained by associating a probability distribution to several uncertain model parameters and by propagating the parametric input uncertainty through the deterministic model. Typically a maximum likelihood identification procedure is used to estimate the parametric input uncertainty from observations. Such inverse identification procedures require to iterate the expensive forward input uncertainty propagation. In this chapter we establish an efficient forward uncertainty propagation method by combining Polynomial Chaos and moment matching. The high-order statistical moments of the output distribution are estimated using the generalized Polynomial Chaos framework and by using the maximum entropy strategy a distribution can be associated to them. This strategy is numerically very attractive due to reduced forward sampling and deterministic nature of the propagation strategy. The strategy is integrated in the inverse uncertainty identification of a wet clutch system for which certain model parameters exhibit inconsistent behaviour and are therefore considered as stochastic. The number of required model simulations to achieve the same accuracy as the brute force methodologies is decreased by one order of magnitude. The probability model identified with the high order estimates resulted into a true log-likelihood increase of about 4% since the accuracy of the estimated output probability density function could be improved up to 47%.*

3.1 Introduction

The increasing performance demands in power-trains require well designed transmission systems that enable energy efficient transitions between a large range op operating points. Clutches are widely used to transfer torque from an input shaft to an output shaft by means of friction. Accurate models of these complex systems have been used for fault diagnosis purposes [1] and numerical optimization of both design and control [2]. In this research we study the modelling of a wet clutch which is characterized by having the friction plates

bathed in oil to assure better heat conduction of the friction losses. Models that incorporate both the hydraulic and mechanical properties of these systems have been proven useful in condition monitoring [3] and control [4, 5].

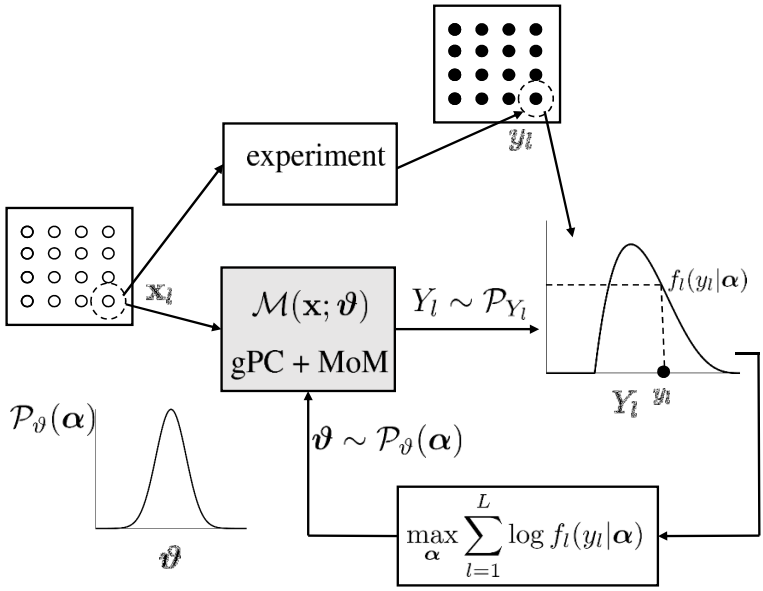


Figure 3.1: Proposed identification method determines optimal $\boldsymbol{\alpha}$, associated to $\mathcal{P}_\vartheta(\boldsymbol{\alpha})$ of the uncertain parameters within the model, via MLE. The parameter distribution $\mathcal{P}_\vartheta(\boldsymbol{\alpha})$ is propagated through the model towards a corresponding output distribution $\mathcal{P}_{Y_l}(\boldsymbol{\alpha})$ for given model input $\mathbf{x}_l$. The novelty lies in the numerical approximation of $\mathcal{M}(\mathbf{x}_l;\boldsymbol{\vartheta})$ by the sequential use of gPC and PDF construction based on MoM of gPC based high-order moments.

Although the value of having accurate and robust models for mechatronic applications, as illustrated with the wet friction clutch, is straightforward, the construction of accurate models remains a cumbersome process. Mostly (mechatronic) models are derived from first principles. Lumped parameter values are then determined empirically by fitting the model output to observations. Since these systems are plagued by their intrinsic complexity and nonlinear behaviour [6], model discrepancies are inevitable [7]. For dynamical phenomena, i.e. variations in the physical behaviour over time due to e.g. wear of plates or degradation and centrifugal effects of the oil in the wet friction clutches, real-time filtering can be applied, even for nonlinear systems and non-Gaussian measurement noise [8, 9], to obtain the superior system model at that time. Alternatively, for static phenomena where one cannot obtain a superior estimate over time, one may associate a stochastic variable

to inconsistent model parameters to express the aleatoric uncertainty. Models with ingrained parametric uncertainty have proven to increase robustness in the field of fault diagnosis [10] and control [11, 12].

In this research we propose to associate a parametrized input probability distribution to several lumped model parameters [13]. We assume that the uncertainty structure (i.e. the parametrized input probability distribution) is fixed, and only its parameters, such as mean and standard deviation (e.g. for a normal distribution), need to be identified based on what parameters best explain a number of experiments. Particularly, this research considers the inverse uncertainty identification of a static model predicting the shifting time of the clutch for various operating points. Figure 3.1 illustrates the iterative process to identify the distribution of uncertain model parameters within a Maximum Likelihood Estimation (MLE) framework [14]. In practice, this requires a forward uncertainty propagation of the uncertain model parameters to the output variables (response) during each iteration. Typically, this is performed by means of prevailing methods such as (quasi) Monte Carlo (MC) [15, 16]. These techniques are however curtailed by the computational inconvenience that comes with the numerous forward simulations required to achieve an acceptable degree of accuracy. More recent work has revived the exploitation of generalized Polynomial Chaos (gPC) expansions that can lead to significant gains in terms of computational feasibility [17–21]. In the gPC framework, the propagation of input uncertainty to output variables is realized by developing the stochastic subspace through a series expansion using orthogonal polynomials. This enables an efficient mathematical context that allows to express the statistical moments as a function of the polynomial coefficients. Once these moments are available, the conditional Probability Density Function (PDF) of the outcome of the experiment can be retrieved through the well-known method of moments (MoM) [22, 23].

To our knowledge, application of the gPC framework to estimate probabilistic moments of a stochastic output model has remained limited to mean and variance estimates. In this work we present the influence of passing high-order statistical moments obtained from the gPC expansion coefficients to the moment matching algorithm. Herewith, we aim for better approximations of the output distributions without requiring additional model evaluations.

First, we elaborate our inverse uncertainty identification method to define different methods and concepts that are included within our theoretical framework. Next we address the wet clutch system and discuss the static shifting time model and performed experiments. Lastly, the developed methods are applied on the clutch application to identify the parametric probability model. The accuracy of the estimated output distributions and sampling efficiency of the proposed uncertainty propagation methodology are compared against the conventional techniques.

3.2 Methodology

3.2.1 Problem Statement

Consider a physics based and nonlinear forward model, $\mathcal{M} : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$, that models the outcome of a physical experiment, y, as $\mathcal{M}(\mathbf{x}; \boldsymbol{\theta})$. Here $\mathbf{x} \in \mathbb{R}^m$ represents a system input that varies for different experiments, and $\boldsymbol{\theta} \in \mathbb{R}^n$ represents a (lumped) physical model parameter that, initially, we assume to be consistent over all experiments. Further consider that we have access to a set of L deterministic measurements $\mathbf{y} = \{y_l\}_{l=1}^L$ for a set of L distinct system inputs $\{\mathbf{x}_l\}_{l=1}^L$. Conventional system identification procedures try to determine the model parameter $\boldsymbol{\theta}^*$ by minimizing the average squared error between the measurements, y_l, and the modelled outputs, $\mathcal{M}(\mathbf{x}_l; \boldsymbol{\theta})$. Here we introduced the shortened notation, $\mathcal{M}_l(\boldsymbol{\theta}) = \mathcal{M}(\mathbf{x}_l; \boldsymbol{\theta})$.

$$\min_{\boldsymbol{\theta}} \frac{1}{L} \sum_{l=1}^{L} \|y_l - \mathcal{M}_l(\boldsymbol{\theta})\|_2^2 \tag{3.1}$$

In many practical conditions, there will remain a mismatch between the measurement, y_l, and the optimized model output, $\mathcal{M}_l(\boldsymbol{\theta}^*)$, even if the measurements can be assumed to be deterministic (i.e. no measurement noise). This observation either implies: that the forward model is only approximate, or, that the assumption that parameter $\boldsymbol{\theta}$ is consistent over all experiments is invalid [13].

In this chapter we investigate the second option and therefore assume that the model parameter set $\boldsymbol{\theta}$ does not correspond to a deterministic value. Alternatively, we propose to address the model parameter set as a stochastic variable $\boldsymbol{\vartheta} \sim \mathcal{P}_\vartheta$ having distribution $\mathcal{P}_\vartheta$ [13]. This modelling strategy renders the forward model stochastic and rather than exact predictions, the model now generates a random output $Y_l = \mathcal{M}_l(\boldsymbol{\vartheta})$. It follows that problem (3.1) is no longer defined. Instead of identifying $\boldsymbol{\theta}$, we are now interested in identifying the distribution $\mathcal{P}_{\boldsymbol{\vartheta}}$[1]. This problem is known as (inverse) uncertainty identification [24].

3.2.2 Inverse Identification Problem

To identify the input distribution $\mathcal{P}_\vartheta$, we suggest the following strategy. For estimation purposes, first we propose to model $\mathcal{P}_\vartheta$ using a parametrized probability distribution family, $\mathcal{P}(\boldsymbol{\alpha})$. Here $\boldsymbol{\alpha}$ represents the distribution parameter to be estimated. The selection of an appropriate distribution family is a task

[1] In this chapter we are concerned exclusively with probabilistic models of uncertainty.

that requires expertise and is not our main subject of concern[2]. Secondly, we propose to determine $\boldsymbol{\alpha}^*$ by maximizing the likelihood of the observations $\mathbf{y}$. Given that we have modeled $\mathcal{P}_\vartheta$ with $\mathcal{P}(\boldsymbol{\alpha})$, we can express the likelihood $\mathcal{L}$ of the observations $\mathbf{y}$ using the conditional density function $f(\mathbf{y}|\boldsymbol{\alpha})$. Hence we can now address the following (inverse) optimization problem, where we can use a logarithmic transform without loss of generality, yielding the traditional maximum log-likelihood estimate (MLE) [24].

$$\max_{\boldsymbol{\alpha}} \log \mathcal{L}(\boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \log f(\mathbf{y}|\boldsymbol{\alpha}) \tag{3.2}$$

Let us now also assume that our experiments are uncorrelated so that we can rewrite optimization problem (3.2) as given below. Here $f_l(y_l|\boldsymbol{\alpha})$ represents the conditional probability of the outcome of the l-th experiment given the probability distribution model $\mathcal{P}(\boldsymbol{\alpha})$. This optimization problem can be solved using any non-gradient based optimization method such as the simplex method [25], interpolation based methods [26–28], evolutionary strategies [29, 30] or traditional genetic algorithms [31, 32].

$$\max_{\boldsymbol{\alpha}} \sum_{l=1}^{L} \log f_l(y_l|\boldsymbol{\alpha}) \tag{3.3}$$

The major challenge faced when solving (3.3), is to propagate the input uncertainty to the output uncertainty $Y_l = \mathcal{M}_l(\boldsymbol{\vartheta})$; equivalently to evaluate the expressions $f_l(y_l|\boldsymbol{\alpha})$, especially since any given numerical optimization method is iterative. In the uncertainty quantification literature this process is referred to as (forward) uncertainty propagation. Well established methods are Monte Carlo methods and perturbation based methods. Monte Carlo methods are very sample intensive and become highly inefficient when used in the context of optimization. Perturbation based methods on the other hand, involve coarse approximations especially when $\mathcal{M}_l$ is nonlinear. In the remainder of this section we develop an original and efficient method to evaluate the expressions f_l.

3.2.3 Forward Uncertainty Propagation Strategy

To accomplish our goal we will combine two well-known concepts in the context of uncertainty propagation. First we will engage the generalized Polynomial Chaos (gPC) expansion framework to characterise the output uncertainty

[2]However, since we have assumed that the model parameters $\boldsymbol{\theta}$ have a physical background, it is possible to base this choice on physical insights. Moreover, the proposed method is independent of the used distribution family. If it is not clear which distribution family to select, our procedure can be practised for several families in order to determine the superior one.

by estimating its statistical moments. Secondly, we will leverage these estimates, practising the method of moments, to provide a maximum entropy distribution density estimate $\hat{f}_l$. Before we detail the proposed methodology extensively, let us briefly introduce the different aspects of it:

Generalized Polynomial Chaos

gPC is a mature framework that has gained a lot of popularity in the uncertainty quantification community [33]. By substituting a polynomial series expansion for the forward nonlinear model an advantageous mathematical setting emerges to propagate uncertainty. This setting is established by choosing the polynomial basis so that it satisfies orthogonality conditions with respect to the probability density function describing the input uncertainty, and allows to express the statistical moments in function of the coefficients associated to the expansion. The underlying assumption is that one can obtain good estimates of the coefficients with less samples than are required to obtain reliable moments estimates with Monte Carlo methods [34]. Conventional application of the gPC framework is limited to the mean and variance [35]. In this work we expand the framework to yield high-order moment estimates so that we can capture nonlinear effects and bimodal output density functions. This is an original contribution since the use of high-order moment estimates is, to the best of our knowledge, unprecedented in the literature.

Method of Moments

Once high-order moment estimates are obtained, we practice the method of moments to construct an estimate of the probability density function $\hat{f}_l(\cdot|\boldsymbol{\alpha})$. This is accomplished by matching the estimated output moments with those of a parametric density function. To what extent a density may be determined uniquely from the knowledge of its moments is still a topic of debate in the mathematical literature, however is not the focus of our contribution[3]. In this work, we use the maximum entropy distribution that can fit a probability density function to any arbitrary number of stochastic features, in this case the statistical moments. For more details concerning the maximum entropy distribution we refer to appendix 3.6.

The gPC framework is introduced rigorously next. Then we will discuss its practice to provide high-order statistical moment estimates and discuss possibilities to determine the expansion coefficients.

[3]This is the classical *moment problem* for which there exist several verification criteria, however, none that are applicable in the context of arbitrary nonlinear forward models, $\mathcal{M}_l$.

3.2.4 Generalized Polynomial Chaos Framework

According to the polynomial approximation theorem, any smooth function, $\mathcal{M}_l : \mathbb{R}^n \to \mathbb{R}$, can be represented as an infinite polynomial series expansion where $\{c_i\}$ represent the expansion coefficients and $\{p_i\}$ the corresponding polynomial basis elements [36]. We define the stochastic variable $\boldsymbol{\varphi} \sim \mathcal{P}_\varphi$ which is distributed according to a non-parametrized standard distribution $\mathcal{P}_\varphi$ (e.g. std. uniform, std. normal ...). Here we implicitly assume that there exist a transformation $\boldsymbol{\vartheta} = \eta(\boldsymbol{\varphi}; \boldsymbol{\alpha})$ based on the distribution parameters $\boldsymbol{\alpha}$ to represent $\mathcal{P}_\vartheta$ starting from a standard distribution $\mathcal{P}_\varphi$. Furthermore, for simplicity purposes we will absorb this transformation in the model by defining $\mathcal{M}'_l(\boldsymbol{\varphi}) = \mathcal{M}_l(\eta(\boldsymbol{\varphi}; \boldsymbol{\alpha}))$.

$$\mathcal{M}_l(\boldsymbol{\vartheta}) = \mathcal{M}'_l(\boldsymbol{\varphi}) = \sum_{i=1}^{\infty} c_i p_i(\boldsymbol{\varphi}) \tag{3.4}$$

For practical purposes our interest is reserved to the d-th order approximation that is obtained by omitting any higher order polynomials from the series expansion. To explore the connection with probabilistic uncertainty propagation, we have to establish the rigorous mathematical framework first.

Let $\mathcal{P}_n^d$ be the n-variate polynomial space of at most degree d and let $\mathbf{p} = \{p_i\}_{i=1}^r$ serve as basis for $\mathcal{P}_n^d$ with $r = \frac{(n+d)!}{n!d!}$. The d-th order approximation is then given by (3.5) for given coefficients $\{c_i\}_{i=1}^r$.

$$\mathcal{M}_l'^{(d)}(\boldsymbol{\varphi}) = \sum_{i=1}^{r} c_i p_i(\boldsymbol{\varphi}) \tag{3.5}$$

Such an n-variate basis $\mathbf{p}$ can be constructed by combining and multiplying elements from n univariate bases $\mathbf{h}^{(k)} = \{h_j^{(k)}\}_{j=0}^d, k \in \{1, \ldots, n\}$ using the following relation, where $\underline{i}$ represents the multi-index $(i_1, \ldots, i_n)$ and $|\underline{i}|$ is defined as $\sum_{k=1}^n i_k$. This is simply dense notation to denote all combinations that arise by picking an element from each basis $\mathbf{h}^{(k)}$ which, when multiplied, result into a polynomial of order d at most.

$$p_{\underline{i}} = \prod_{k=1}^{n} h_{i_k}^{(k)}, \ |\underline{i}| \leq d \tag{3.6}$$

For notational efficiency, we will exploit the bijection between index $\underline{i}$ and $i \in \mathcal{I} = \{1, \ldots, r\}$ and jump in between notations later on.

As mentioned, the gPC framework allows to establish an advantageous computational setting in the context of uncertainty propagation by a distinct choice of the basis, $\mathbf{p}$. This setting is achieved by constructing the basis, $\mathbf{p}$, so that it is orthogonal with respect to the PDF, f_φ, associated to the variable,

$\boldsymbol{\varphi}$. A polynomial basis is said to be orthogonal with respect to an arbitrary distribution f_φ if the following condition holds.

$$\langle p_i, p_j \rangle \equiv \mathrm{E}\left\{p_i p_j\right\} = \int p_i(\boldsymbol{\phi}) p_j(\boldsymbol{\phi}) f_\varphi(\boldsymbol{\phi}) \mathrm{d}\boldsymbol{\phi} \tag{3.7}$$

Now assume that $\boldsymbol{\varphi}$ is composed of n independently distributed random variables, φ_k, with known supports and PDF, f_{φ_k}. Then the joint PDF, f_φ, is given by $\prod_k f_{\varphi_k}$. Now, if we construct the multivariate basis as described above and so that the generating univariate bases, $\mathbf{h}^{(k)}$, satisfy the orthogonality condition with respect to the PDFs, f_{φ_k}, also the multivariate basis, $\mathbf{p}$, will satisfy the orthogonality condition with respect to the joint PDF, f_φ, considering that

$$\begin{aligned}\langle p_i, p_j \rangle &= \prod_{k=1}^{n} \int h_{i_k}^{(k)}(\phi_k) h_{j_k}^{(k)}(\phi_k) g_k(\phi_k) \mathrm{d}\phi_k \\ &= \prod_{k=1}^{n} \left\langle h_{i_k}^{(k)}, h_{j_k}^{(k)} \right\rangle = \delta_{ij}\end{aligned}$$

If these conditions are met, it will turn out that we can calculate the statistical moments directly from the polynomial coefficients. This is exactly the mathematical short cut that the gPC framework provides. A link between several standard univariate distributions and polynomial families is given by the so-called Wiener-Askey polynomial chaos scheme, see Table 3.1. In general the stochastic variable $\boldsymbol{\vartheta}$ is not distributed according to one of the these standard distributions. Therefore we invoke an inverse transformation $\boldsymbol{\varphi} = \eta^{-1}(\boldsymbol{\vartheta}; \boldsymbol{\alpha})$ towards a distribution $\mathcal{P}_\varphi$ which is included in the Wiener-Askey polynomial chaos scheme. Alternatively one can construct a new polynomial basis using an orthogonalization procedure such as the Gram-Schmidt algorithm [37].

Table 3.1: Wiener-Askey polynomial chaos.

distribution, f_φ	polynomials, h_i	support
Gaussian	Hermite	$[-\infty, \infty]$
Gamma	Laguerre	$[0, \infty]$
Beta	Jacobi	$[-1, 1]$
Uniform	Legendre	$[-1, 1]$

How the polynomial coefficients can be determined will be discussed in section 3.2.6. How high-order statistical moments can be estimated with the generalized Polynomial Chaos expansions framework is discussed in the next section.

3.2.5 Moment Estimation with gPC

The m-th statistical moments of a random variable, $Y = \mathcal{M}'_l(\boldsymbol{\varphi})$, where $\boldsymbol{\varphi}$ is distributed according to f_φ, is defined as

$$\mu_m = \mathrm{E}\left\{Y_l^m\right\} = \mathrm{E}\left\{\mathcal{M}'_l(\boldsymbol{\varphi})^m\right\} = \int \mathcal{M}'_l(\boldsymbol{\phi})^m f_\varphi(\boldsymbol{\phi})\mathrm{d}\boldsymbol{\phi} \tag{3.8}$$

The elegance of the gPC framework reveals itself when we substitute the d-th order polynomial approximation of the output model in (3.8). We retrieve an approximate expression for the m-th moment in function of the polynomial expansion coefficients. Here the last term represents a trivial generalization of the inner product $\langle\cdot,\cdot\rangle$.

$$\begin{aligned}\mu_m^{(d)} &= \mathrm{E}\left\{\mathcal{M}_l'^{(d)}(\boldsymbol{\varphi})^m\right\} = \int \left(\sum_{i\in\mathcal{I}} c_i p_i(\boldsymbol{\phi})\right)^m f_\varphi(\boldsymbol{\phi})\mathrm{d}\boldsymbol{\phi} \\ &= \sum_{i_1\in\mathcal{I}} \cdots \sum_{i_m\in\mathcal{I}} c_{i_1}\cdots c_{i_m}\left\langle p_{i_1}\cdots p_{i_m}\right\rangle\end{aligned} \tag{3.9}$$

From this expression one may easily verify that the first two moments, respectively the mean and variance μ and σ^2, can be estimated as shown below.

$$\begin{aligned}\mu &\approx \mu_1^{(d)} = \langle p_1^2\rangle \cdot c_1 \\ \sigma^2 &\approx \mu_2^{(d)} = \sum_{i\in\mathcal{I}} \langle p_i^2\rangle \cdot c_i^2\end{aligned}$$

3.2.6 Determination of Expansions Coefficients

Theoretically, the polynomial coefficients from (3.5) can be obtained by projection of the forward model on the polynomial space exploiting the orthogonality of the expansion basis [20]. When we take the inner product of (3.4) and the elementary polynomials h_i, all terms but one will vanish.

$$c_i = \left\langle \mathcal{M}'_l, p_i\right\rangle = \int \mathcal{M}'_l(\boldsymbol{\phi}) p_i(\boldsymbol{\phi}) f_\varphi(\boldsymbol{\phi})\mathrm{d}\boldsymbol{\phi} \tag{3.10}$$

Since the scope of this work is on general nonlinear forward models, we can not evaluate the associated integral explicitly. We therefore approximate the integrals using Gaussian-quadrature rules (3.11). A univariate Gaussian-quadrature of order q is defined by a set of weights w_j and corresponding collocation points $\boldsymbol{\phi}_j$, $\{(w_j, \boldsymbol{\phi}_j)\}_{j\in\mathcal{Q}}, \mathcal{Q} = \{1,\ldots,q\}$. This set will depend on the (univariate) probability density function function, f_φ, and is as such related to the polynomial basis. A quadrature of order q is exact for polynomials up to degree $2q-1$. The order hence affects the number of coefficients that can

be retrieved correctly and therefore the accuracy of the polynomial approximation. In the multivariate case, a tensor product of univariate quadrature rules is considered. Note that the number of collocation points will scale exponentially with the number of dimensions. For additional details we refer to [19].

$$c_i \approx \sum_{j \in \mathcal{Q}} \mathcal{M}'_l(\boldsymbol{\phi}_j) p_i(\boldsymbol{\phi}_j) w_j \tag{3.11}$$

Similar to MC techniques, collocation based gPC requires a number of evaluations of the forward model, $\mathcal{M}'_l(\boldsymbol{\phi}_j)$. The difference is that the estimation of the statistical moments is realized through a mathematical detour. The focus of approximation in the gPC framework is on modelling the polynomial response function whilst that of the MC approach is on the direct estimation of the output distribution. The accuracy of gPC depends on the capacity of the basis to capture the nonlinearity of the forward model rather than on the capacity of the sampling method to properly represent the input uncertainty by the spatial distribution of the sample points. It is the prevailing consensus that an equivalent level of accuracy can be achieved with only a fraction of the input points of any MC approach.

3.2.7 Proposed Uncertainty Propagation Algorithm

Finally we can combine all the techniques introduced so far to achieve our initial goal, which was to evaluate the expressions $f_l(y_l|\boldsymbol{\alpha})$ to solve problem (3.3). We provide the full recipe here, assisted by Fig. 3.2.

Design Choices

First we decide on a parametric distribution family $\boldsymbol{\vartheta} \sim \mathcal{P}(\boldsymbol{\alpha})$. This requires finding a transformation $\boldsymbol{\vartheta} = \eta(\boldsymbol{\varphi}, \boldsymbol{\alpha})$ for $\boldsymbol{\varphi}$ distributed according to a standard distribution model $\mathcal{P}_\varphi$ from the Wiener-Askey scheme. Next, we choose the degree d and construct the polynomial basis $\mathbf{p}$ corresponding to the distribution $\mathcal{P}_\varphi$ from the Wiener-Askey scheme. The value of d should be sufficient to model the nonlinearity of the forward model and is considered a trial-and-error procedure. The multivariate basis $\mathbf{p}$ can be build using standard polynomials univariate bases $\mathbf{h}$. Once the basis is defined we can decide on the order q of the Gaussian-quadrature which determines the number of collocation points and the maximum polynomial degree that can be estimated. In general we have that $q > d$. Lastly, the number of high-order moments $m > 2$, that we will use to assess the output uncertainty, are chosen. Note that this number determines the possible *shapes* that the output density can adopt, see appendix 3.6 for details. For bimodal output distributions we should choose $m \geq 4$.

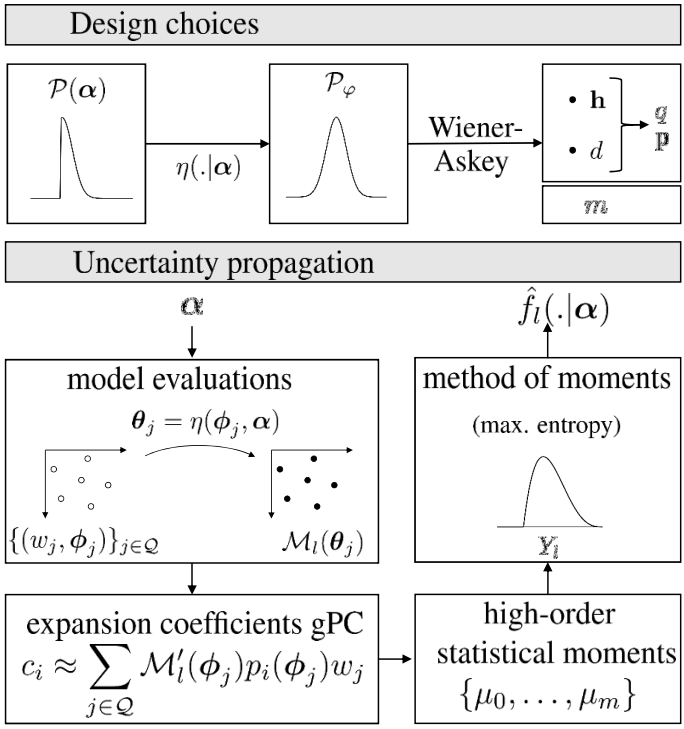


Figure 3.2: Forward uncertainty propagation with PDF reconstruction from high-order moments derived from coefficients of gPC model.

Uncertainty Propagation

The PDF $f_l(.|\boldsymbol{\alpha})$ for a given parametric uncertainty $\boldsymbol{\alpha}$ is approximated starting from a limited number of function evaluations $\mathcal{M}_l(\boldsymbol{\theta}_j)$. In practice, the model is evaluated in the collocation points defined by the chosen quadrature order q. This requires a transformation $\boldsymbol{\theta}_j = \eta(\boldsymbol{\phi}_j; \boldsymbol{\alpha})$ that involves $\boldsymbol{\alpha}$. These function evaluations, together with the corresponding weights w_j of the collocation points are used to approximate the coefficients c_i of the gPC model. Once these coefficient values are available, we can determine the high-order statistical moments μ_m which are then passed on to the maximum entropy distribution to generate the approximation $\hat{f}_l(\cdot|\boldsymbol{\alpha})$ which we can evaluate for y_l to facilitate the desired evaluation of $f_l(y_l|\boldsymbol{\alpha})$.

Also worth mentioning here is the common conception in the literature [38, 39] to obtain the moments, or even directly the PDF, by applying MC techniques on the polynomial expansion (3.5) which is in fact relatively *cheap* to evaluate. In this approach the expansion is used as a response function and the benefit of gPC is only exploited through the supposed superior conver-

gence rate of the expansion for polynomials corresponding the input distribution (3.5). Surprisingly enough, when compared to its peers, gPC is usually applied in this sense [19, 38, 40]. For this contribution we have deliberately omitted this approach, since then it would be as easy to build any approximate model and sample that. Our main objective is to verify whether the gPC framework can be used to provide the high-order moment estimates.

3.2.8 Example of gPC Based Forward Uncertainty Propagation by Higher-Order Moment Matching

In order to illustrate the forward propagation of uncertainty concatenating the techniques described in the sections 3.2.3 to 3.2.6, we apply the methodology on an illustrative univariate nonlinear model, $y(x;\theta)$ having uncertain model parameters $\vartheta \sim \mathcal{P}_\vartheta$.

$$y(x;\theta) = \tan\left(\tfrac{1}{8}\theta x\right) + \exp\left(\tfrac{1}{6}\theta - x\right) + \tanh(\tfrac{1}{2}\theta x)$$

Choice of Parametric Uncertainty $\mathcal{P}_\vartheta$

We assume $\mathcal{P}_\vartheta$ being a Gaussian distribution $\mathcal{N}(\mu,\sigma)$ that is clipped between between $[\theta^-,\theta^+]$. Consequently, the distribution $\mathcal{P}_\vartheta(\boldsymbol{\alpha})$ can be parameterized by $\boldsymbol{\alpha} = (\mu,\sigma,\theta^-,\theta^+)$. Figure 3.3 illustrates the probability density function of $\mathcal{P}_\vartheta(\boldsymbol{\alpha})$ for $\boldsymbol{\alpha} = (0,2,-2,5)$. The forward uncertainty propagation of $y(x;\vartheta)$ for a deterministic input $x = 1$ results into a random variable, Y, plotted in the upper left graph. Additionally, an MC sample set of $N = 50$ is visualized by the black dots with corresponding 10 bin histogram in gray.

Choice of Standard Distribution $\mathcal{P}_\vartheta$

In order to approximate the PDF $f(|\boldsymbol{\alpha})$ of the output variable Y for limited function evaluations we apply gPC. This requires finding a relation $\vartheta = \eta(\varphi;\boldsymbol{\alpha})$ with φ having a distribution $\mathcal{P}_\varphi$ included in the Wiener-Askey diagram. We choose $\varphi \sim \mathcal{N}(0,1)$ resulting in the use of Hermite polynomials h_i according to Table 3.1. The mapping η is facilitated by transforming $\varphi \sim \mathcal{N}(0,1)$ to a stochastic variable $\tilde{\varphi}$ that is distributed according to $\frac{1}{F_\mathcal{N}(\phi^+)-F_\mathcal{N}(\phi^-)} f_\mathcal{N}(\tilde{\phi}), \tilde{\phi} \in [\phi^-,\phi^+]$ where $\phi^- = \frac{1}{\sigma}(\theta^- - \mu)$ and $\phi^+ = \frac{1}{\sigma}(\theta^+ - \mu)$. We denote $f_\mathcal{N}(\cdot)$ and $F_\mathcal{N}(\cdot)$ as the PDF and the cumulative density function (CDF) of the standard normal distribution respectively. First we map φ to a uniform distribution in between the values $F_\mathcal{N}(\phi^-)$ and $F_\mathcal{N}(\phi^+)$. Then we use the inverse transform sampling method to map the uniform variable back onto the standard normal distribution but now limited to the interval $[\phi^-,\phi^+]$. In conclusion the parametric probability transformation $\vartheta =$

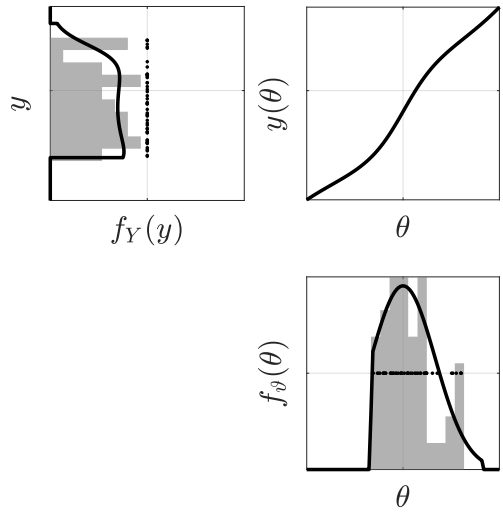


Figure 3.3: Nonlinear transformation of standard normal random variable. The lower right plot shows the density of the original random variable, ϑ. The variable is passed through the function displayed in the upper right. The density of the output random variable, Y, is plotted in the upper left graph.

$\eta(\varphi; \boldsymbol{\alpha})$ is defined through the transformations

$$\begin{aligned} U &= F_{\mathcal{N}}(\phi^-) + F_{\mathcal{N}}(\varphi)\left(F_{\mathcal{N}}(\phi^+) - F_{\mathcal{N}}(\phi^-)\right) \\ \tilde{\varphi} &= F_{\mathcal{N}}^{-1}(U) \\ \vartheta &= \mu + \sigma\tilde{\varphi} \end{aligned} \tag{3.12}$$

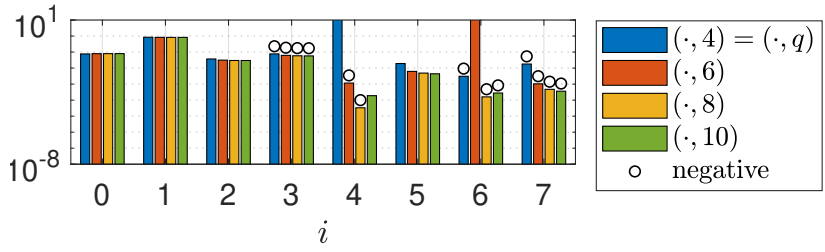
Choice of gPC and Moment Properties

Figure 3.4a depicts the polynomial coefficients for varying quadrature order, q, and using Hermite chaos. One may observe that the higher order coefficients are poorly approximated with low order quadratures and that for this specific model the high order coefficients still exhibit significant magnitude. Recall that a quadrature of order q is exact for polynomials up to degree $2q - 1$. Now, assuming that model $y(x; \theta)$ contains polynomials up to degree d and that we desire to approximate the d-th coefficient, then the integrand in (3.10) will contain a polynomial of degree $2d$. In order to be exact, the quadrature should therefore have order $d + 1$ at least. Remark that the coefficient estimate becomes increasingly precise for q surpassing $d + 1$, as can be seen for e.g. $i = 3$. That is because, $y(x)$ is truly a polynomial of degree $\gg d$ and that for lower quadrature orders the high-order polynomial content deteriorates the lower-order coefficient estimations.

In Fig. 3.4b we compare the stochastic moments numerically obtained from the true PDF[4] with those obtained from gPC approximation (3.5) for varying d and q. Note that we have chosen the quadrature order so that $q \geq d + 1$. As a result, the poorly approximated coefficients from Fig. 3.4a are never considered in the moment computation.

In conclusion, the moments are employed to fit parametric distributions to the output distribution, f_Y. Results are depicted in Fig. 3.5 when using the Gaussian and maximum entropy distribution. The numerical correspondence between two PDFs is quantified by the Earth Mover's Distance (EMD) (Appendix 3.7). Each fit is compared to the actual PDF. The PDF fit obtained with the exact moments serves as an upper bound for the amount of information that is contained within the moments ans is referred to as the reference.

(a) Polynomial coefficients, $c_i[\cdot]$



(b) Stochastic moments discrepancy, $\Delta\mu_i[\%]$

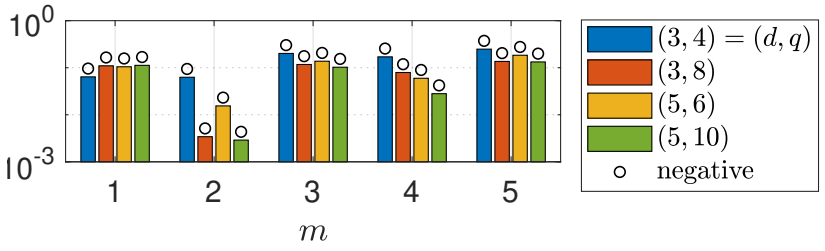


Figure 3.4: Polynomial chaos coefficient estimates for varying quadrature orders are displayed in the top graph. Stochastic moments discrepancy w.r.t. the true moments as obtained from the gPC coefficients and by MC sampling of the gPC approximations are displayed in the bottom graph. Negative values are indicated by a white marker.

[4]For a monotonic function, $y(\theta)$, with inverse, $\theta(y)$, this is equal to, $f_Y(y) = |\theta'(y)| \cdot f_\vartheta(\theta(y)) = |y'(\theta(y))|^{-1} \cdot f_\vartheta(\theta(y))$.

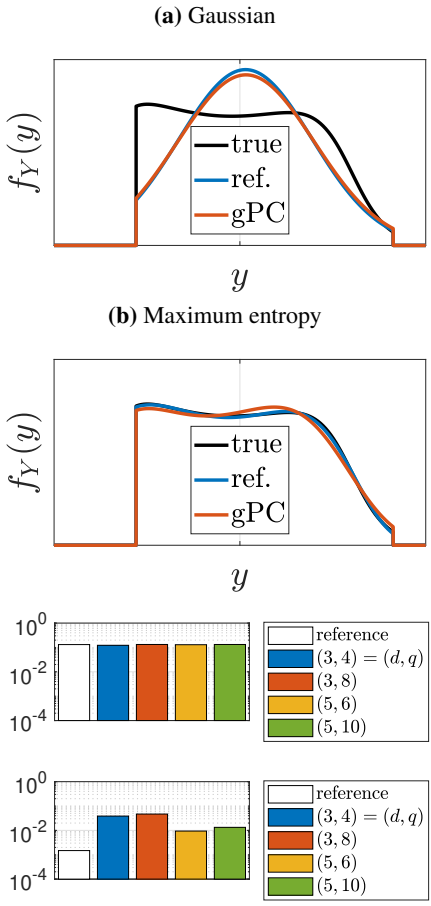


Figure 3.5: Comparison of PDF approximations of a Gaussian and a maximum entropy distribution model fit. The top figures portray the most performant PDF approximation in terms of the EMD. The corresponding EMD values of all approximations considered are compared in the bottom.

3.3 Experimental Validation

3.3.1 System Description

The proposed methodology is applied to identify the parametric uncertainty of a wet clutch system. A clutch is a mechanical device that connects a motoring unit to a load that transforms power into useful work. The objective is to ensure smooth and on demand coupling of the driving shaft to the driven shaft. A wet clutch system realizes this by means of contact friction between two series of friction plates, see Fig. 3.6 for a schematic cross section [4].

By construction these friction plates are fully submerged in oil so that slip induced heat can be transferred adequately. This avoids thermal overheat and increases the clutch's life time. A wet clutch uses the same hydraulic system to activate the engagement process. By pressurizing the hydraulic chamber, a freewheeling piston is pressed against the series of interlocking friction plates realizing a power transfer. The pressure in the hydraulic chamber is controlled via a current signal that activates a servo-valve linking the hydraulic chamber to a pressurized reservoir. The pressure difference over the piston determines its position and therewith the clutch engagement. Since the wet-clutch set-up should be low cost, additional sensors are avoided and a feedforward control strategy using the parameterized current profile shown in Fig. 3.7 is applied. The signal is characterized by three independent control parameters that can be freely chosen to manipulate the shifting process.

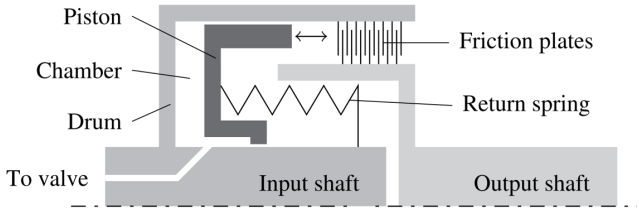


Figure 3.6: Cross section of the wet clutch system.

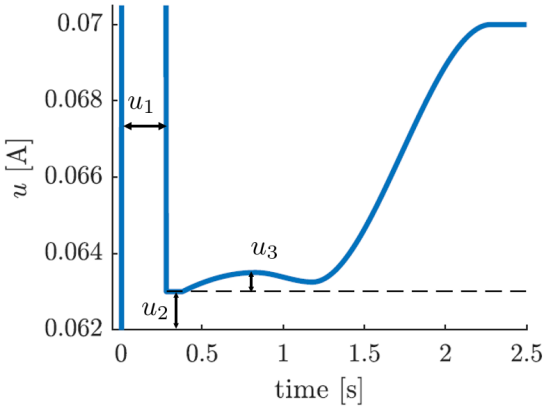


Figure 3.7: Feedforward current signal.

The shifting time, defined as the time interval between initialization of the feedforward control signal and the moment that both shafts obtain the same angular speed, is typically key for the end user. Unfortunately, this is highly dependent on the operating points of the system. Therefore, we derived a static model, including the underlying dynamic model described in Appendix 3.8, to predict the shifting time for given operating points. The model is evaluated on the test set-up depicted in Fig. 3.8. An AC electric motor (30 kW) is controlled to a constant speed ω_m via a high bandwidth motor drive. The motor is connected to a controlled transmission via a torque converter. Within this transmission, the clutch can be controlled to engage with the proper gear. Furthermore, the output shaft of the clutch is connected to a load transmission. The load consists of a flywheel ($2,5\ \mathrm{kg\,m^2}$) and a brake that delivers a counteracting torque T_b. This system was previously used to study smooth gearbox controllers [4].

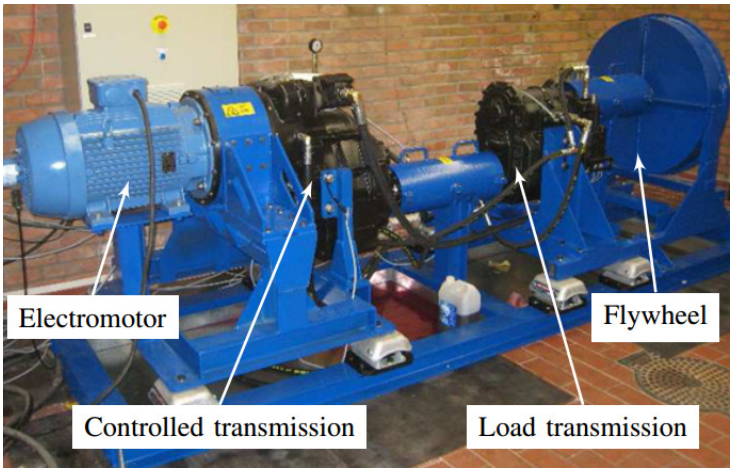


Figure 3.8: Wet clutch test bench.

3.3.2 Design of Experiments

The influence of the operating point $\mathbf{x}$ on the shifting time y is experimentally tested on the wet clutch test set-up. We performed an up-shift from neutral to first gear for various control parameters and conditions, spanning a test scenario space. More specifically, 18 distinct feedforward control signals were tested parameterized by the control parameters u_1,u_2,u_3 as defined in Fig.3.7. Further we verified 3 different motor speeds ω_m (1200, 1350 and 1500 rpm) and 2 different friction load conditions T_b (low, high). For these, a full factorial design of operating points $\{\mathbf{x}_l\}_{l=1}^{108}$ has been tested. Figure 3.9 schematically illustrates the operating point $\mathbf{x}$ being evaluated at the static shifting time model

$\mathcal{M}$. As an illustration of the modeling deficiency, we compare the actual measurements, y_l, with the simulated shifting times, $\hat{y}_l$, in Fig. 3.10. The global trend is captured by the deterministic model, however a distinct discrepancy between measurements and simulation is present.

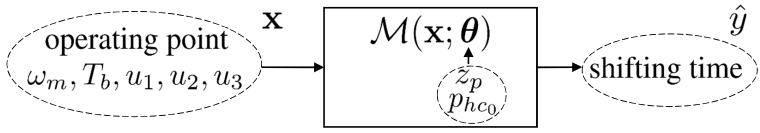


Figure 3.9: Scheme of static shifting time model of the wet clutch for different operating points.

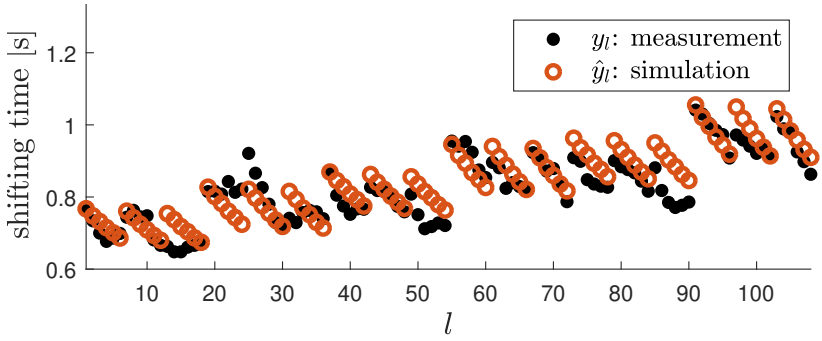


Figure 3.10: Experimental versus deterministic simulation results for all operating points $\{\mathbf{x}_l\}_{l=1}^{108}$.

3.3.3 Parametric Uncertainty Model

The parameters of shifting time model in Appendix 3.8 were identified empirically based on isolated subsystem measurements. However, after each engagement the internal clutch pressure and initial piston position is changed so that it becomes inconvenient to associate a fixed parametric value to the related variables p_{hc_0} and z_p. Since these parameters cannot be measured we address them as stochastic variables. Therefore, we aim at identifying a probabilistic model $\mathcal{P}_\vartheta$ for the random variable $\boldsymbol{\vartheta} = (\vartheta_1; \vartheta_2)$ which includes the scaling factor of the nominal values of p_{hc_0} and z_p respectively. We propose a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, with mean $\boldsymbol{\mu} = (\mu_1; \mu_2)$ and covariance Σ. The distribution $\mathcal{P}_\vartheta$ is clipped to a feasible area $\boldsymbol{\vartheta} \in [\boldsymbol{\theta}^-, \boldsymbol{\theta}^+]$ constricting the stochastic domain of $\boldsymbol{\vartheta}$. Further we assume that both variables are independent so that $\Sigma = \mathrm{diag}(\sigma_1^2, \sigma_2^2)$. Consequently, we can use Hermite polynomials to approximate the clutch model using the same relation $\boldsymbol{\vartheta} = \eta(\boldsymbol{\varphi}; \boldsymbol{\alpha})$ as described in (3.12) for $\boldsymbol{\varphi} \sim \mathcal{N}(0, 1)$.

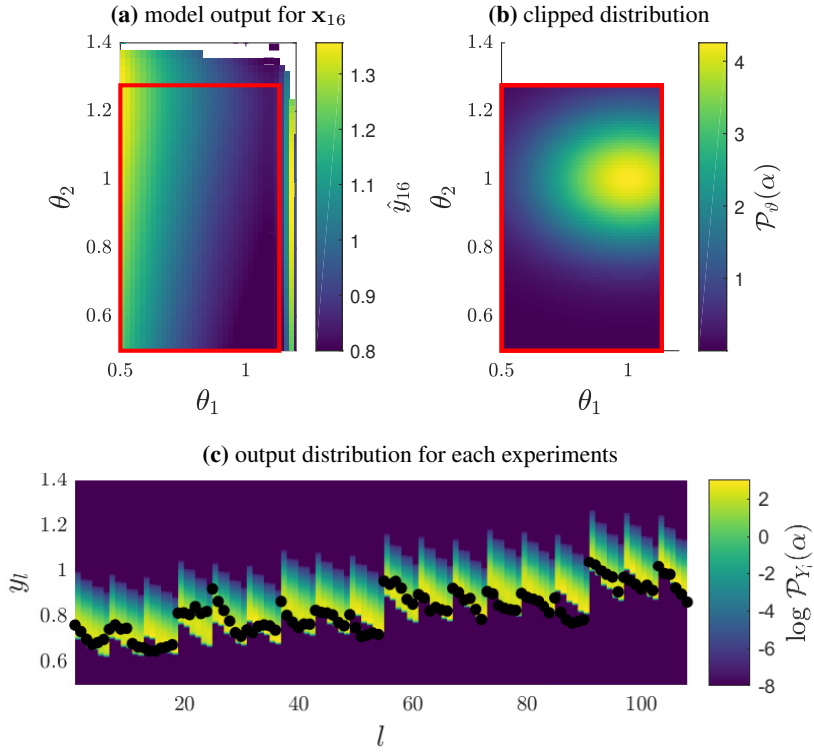


Figure 3.11: *Top left*: Simulated shifting time for $\mathbf{x}_{16}$. The red box indicates the clipping domain. *Top right:* Associated clipped joint distribution. *Bottom*: Representation of corresponding output distributions for all operating points $\{\mathbf{x}_l\}_{l=1}^{108}$. The log-likelihood of the test scenarios for given input distributions is also given.

The clipping boundaries, $\boldsymbol{\theta}^-$ and $\boldsymbol{\theta}^+$, of the stochastic variables are chosen fixed so that the space for which a feasible solution of the shifting time can be obtained for all operating points $\{\mathbf{x}_l\}_{l=1}^{108}$, is maximized. This makes the identification of the parameter distribution $\mathcal{P}_\vartheta(\boldsymbol{\alpha})$ dependent of $\boldsymbol{\alpha} = (\mu_1, \mu_2, \sigma_1, \sigma_2)$. Figures 3.11a and 3.11b illustrate respectively the parametric model output and associated clipped normal input distribution. Figure 3.11c demonstrates the corresponding output distribution for each operating point $\mathbf{x}_l$. Visual comparison with Fig. 3.10 demonstrates that the least-squared approach treats the problem in a non appropriate manner.

3.4 Results and Discussion

In this section we apply the methodologies described in section 3.2 on the wet clutch system as was described in section 3.3. First we demonstrate the forward propagation of uncertainty by concatenating the techniques described in the sections 3.2.3 and 3.2.7, simply to determine the corresponding output distribution by moment matching for a given input distribution. Secondly we use the MLE method to identify the optimal parameter $\boldsymbol{\alpha}^*$ that best describes observed experiments. Each sample involves a simulation of all operating points $\{\mathbf{x}_l\}_{l=1}^{108}$ on a High Performance Computing (HPC) cluster (compute nodes: 2x8-core Intel E5-2670 64 GB RAM). The average computing time per sample on one core takes 2.95 minutes which justifies the need for propagation techniques with enhanced sampling efficiency. The results obtained by max. entropy distributions incorporating high-order moments are compared by those of using Gaussian PDFs. To obtain a benchmark PDF we estimated the output distributions empirically using Monte Carlo (MC) uncertainty propagation of 200 000 simulation samples and fitted a spline curve on the associated histograms. The associated $\log \mathcal{L}$ values are referred to as the true values.

3.4.1 PDF Fitting by the Method of Moments

First we demonstrate the PDF estimation strategy combining high-order gPC moments with the method of moments. Corresponding the variable transformation and associated parametric probability model that were motivated in section 3.3.3, and in concordance with Table 3.1, we make use of Hermite polynomials. We used polynomial order $d = 4$ and corresponding univariate quadrature order $q = 5$, i.e. 25 function simulations were required to estimate the output distribution for a single operating point $\mathbf{x}_l$. Due to numerical restrictions we are limited to the first $m = 4$ moments.

Estimates for the output distribution for operating point $\mathbf{x}_{16}$ are visualized in Fig. 3.12a. We compare the MC reference distribution with Gaussian and max. entropy fits. We also included PDF estimates that are based on the stochastic moments extracted from the MC sample set. Based on these results we can compare how much information could be extracted by the gPC approach with respect to the amount of information that is contained within the first four moments about the true distribution. The minor difference between the gPC PDF estimates and the MC equivalent PDF estimates suggests that gPC is a viable approach with sparse function evaluations.

For a fair comparison besides the visual verification, we calculate the Earth Mover's Distance (EMD) (see Appendix 3.7) for every estimate w.r.t. the estimated true output distribution. Different values are presented in Fig. 3.12b. These results confirm that high-order estimates obtained through the extended gPC framework are capable of extracting statistical information from signifi-

cantly fewer function evaluations when compared to the brute force methodologies such as MC.

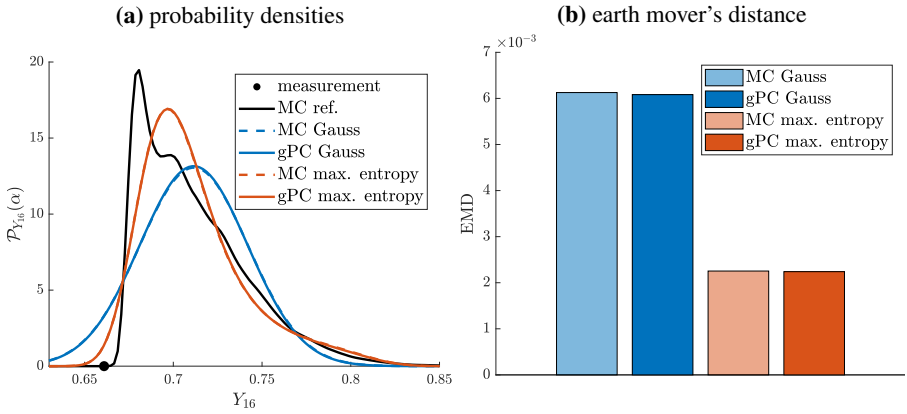


Figure 3.12: PDF estimates and corresponding EMD for given model parameters $\boldsymbol{\alpha} = (1, 1, 0.07, 0.02)$ for operating point $\mathbf{x}_{16}$.

3.4.2 Maximum Log-likelihood Estimation

We demonstrated that the high-order gPC moment matching strategy provides cheap yet accurate output PDF estimates for given input uncertainty. Consequently it can be engaged as an efficient alternative to evaluate the log-Likelihood of a number of experiments at a mere fraction of the computational cost associated to brute force strategies such as MC. Next we apply the method to identify the optimal parameter set $\boldsymbol{\alpha}^*$ associated to the probability model described in section 3.3.3. To solve problem (3.2), we used a genetic algorithm (GA) with population size 50 using the Matlab optimization toolbox. We compared optimization results for both Gaussian and max. entropy PDF abstractions. The performance of the optimal parameter sets are verified using MC estimates of the log-Likelihood corresponding the identified probability models.

Gaussian Output Distribution

Using the Gaussian output distribution and gPC moment estimates, the following optimal parameter set was identified, $\boldsymbol{\alpha}_G^* = (0.91, 0.55, 0.077, 0.014)$. Figure 3.13 compares measurement with the corresponding output distributions. The solution has an estimated log-Likelihood of $\log \hat{\mathcal{L}}^G(\boldsymbol{\alpha}_G^*) = 188.5$ using the moment matching approach. When we evaluate the log-Likelihood using the MC approach, we obtain a smaller value $\log \mathcal{L}(\boldsymbol{\alpha}_G^*) = 176.7$.

Maximum Entropy Output Distribution

Using the max. entropy distribution we obtain the following optimal parameter, $\boldsymbol{\alpha}^*_{ME} = (0.91, 0.55, 0.081, 0.046)$. Note that we retrieve the same mean as with the Gaussian output distributions, $\boldsymbol{\mu}$, but that both covariance values have increased. Figure 3.14 visualizes the corresponding output distributions and measurements. Using the max. entropy moment matching approach, the log-likelihood is estimated on $\log \hat{\mathcal{L}}^{ME}(\boldsymbol{\alpha}^*_{ME}) = 189$. Verification using MC delivers a slightly smaller log-Likelihood $\log \mathcal{L}(\boldsymbol{\alpha}^*_{ME}) = 183.7$. Note that the difference is significantly smaller compared to the log-Likelihood mismatch obtained with the Gaussian PDF estimates.

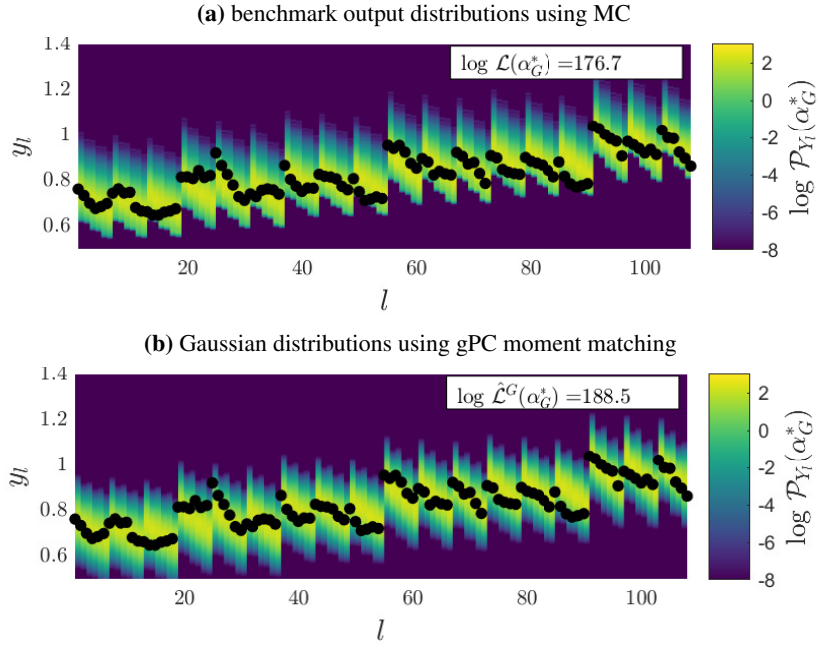


Figure 3.13: Representation of corresponding output distributions of all operating points $\{\mathbf{x}_l\}_{l=1}^{L=108}$ using the optimal parameter set, α^*_G.

3.4.3 Discussion

With respect to the global $\log \mathcal{L}$ estimates, we remark the following. In both the max. entropy and the Gaussian case the $\log \mathcal{L}$ is overestimated using the proposed gPC moment matching method. However, for the max. entropy distribution, the mismatch between true $\log \mathcal{L}$ and estimated $\log \mathcal{L}$ is significantly smaller (approximately 55%). Moreover the true $\log \mathcal{L}$ obtained with $\boldsymbol{\alpha}^*_{ME}$ exceeds that obtained with $\boldsymbol{\alpha}^*_G$ by 4%. Both of these observations confirm that

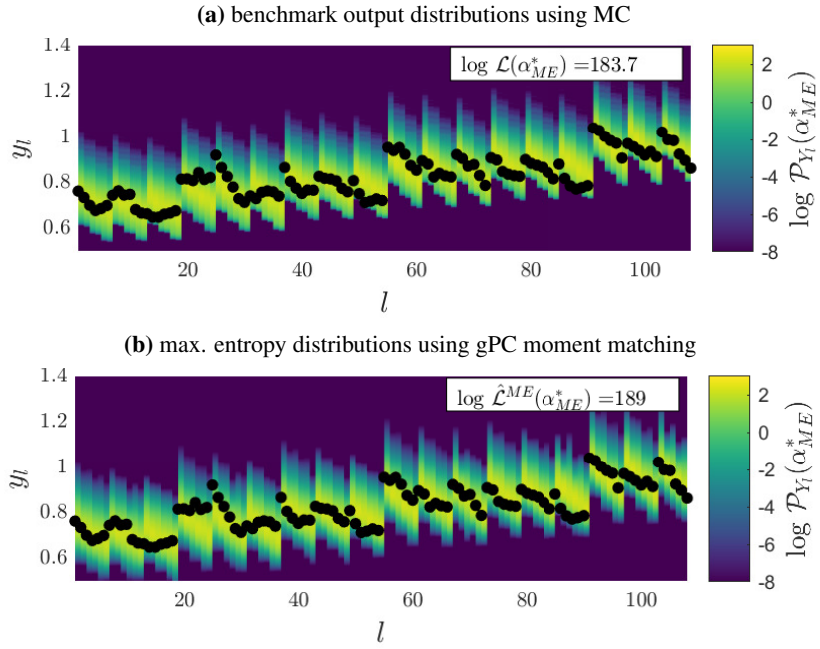


Figure 3.14: Representation of corresponding output distributions of all operating points $\{\mathbf{x}_l\}_{l=1}^{L=108}$ using the optimal parameter set, α^*_{ME}.

the use of high-order stochastic moments benefits the identification of an optimal input probability model and that the gPC framework can be used to obtain accurate estimates of these moments.

Figure 3.15 compares the estimates $\log \hat{\mathcal{L}}_l^G(\boldsymbol{\alpha}^*_G)$ and $\log \hat{\mathcal{L}}_l^{ME}(\boldsymbol{\alpha}^*_{ME})$, for each individual operating point $\mathbf{x}_l$, w.r.t. to their true value. One may note that for the majority of the operating points the $\log \mathcal{L}$ is estimated with great precision for either output distribution. This indicates that for the majority of the experiments, the true output distribution is about Gaussian and no additional information is contained within the high-order moments. However for a small subset of operating points, the Gaussian overestimates the true value significantly due to its incapacity of modeling asymmetric distributions. As a result, measurements that are highly improbable and would strongly affect the global $\log \mathcal{L}$ are penalized less by the Gaussian distribution. This effect was already visualized in Fig. 3.12. Following the argument made above, we compare the EMD metric averaged over all operating points in Fig. 3.16, as it proofs to be valuable to quantify the estimated output distribution's accuracy. We found that the use of the maximum entropy distribution with 4 moments could reduce the EMD by approximately 42% for α^*_G and 47% for α^*_{ME} when compared with the averaged EMD value obtained with Gaussian output dis-

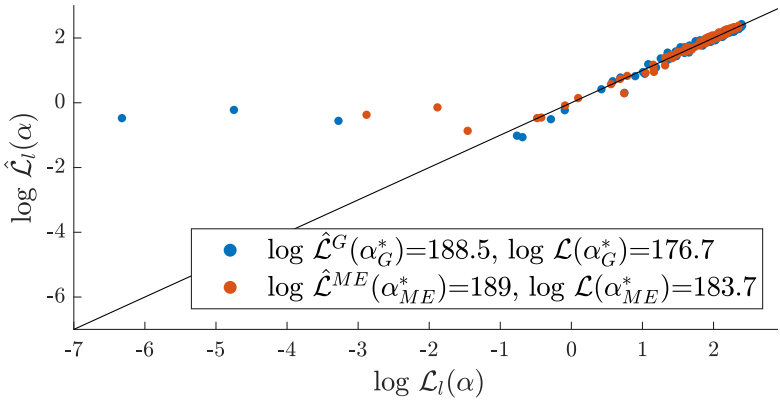


Figure 3.15: Comparison of individual log-Likelihood estimates $\hat{\mathcal{L}}_l(\boldsymbol{\alpha})$ and corresponding reference log-Likelihood $\mathcal{L}_l(\boldsymbol{\alpha})$.

tributions. The minor difference compared to the EMD of the PDF estimates based on the stochastic moments directly extracted from the MC sample set indicates that the gPC can correctly estimate the statistical moments. The improved accuracy can therefore be fully attributed to the higher order moments gPC framework.

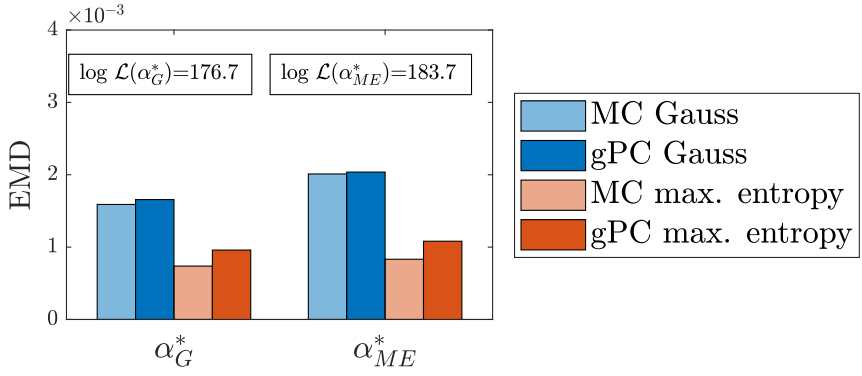


Figure 3.16: EMD of different PDF approximations.

Figure 3.17 illustrates the mean absolute error (MAE) of the log-Likelihood estimates for all operating points $\{\mathbf{x}_l\}_{l=1}^{108}$ compared to the MC reference obtained by 200 000 simulation samples. The accuracy of the brute force propagation techniques increases with respect to the number of samples used for uncertainty propagation. The quasi Monte Carlo (qMC) technique considers an equally distributed grid of S samples. The gPC results require a magnitude less expensive function evaluations to achieve the same accuracy

as the brute force techniques. The interpolated values S_{MQ} and S_{qMQ} are depicted in Table 3.2. A comparison between the gPC techniques indicates that the MAE could be reduced by 15.3% for $\boldsymbol{\alpha}_G^*$ and 24.8% for $\boldsymbol{\alpha}_{ME}^*$ due to the incorporation of higher order estimates in the novel gPC framework.

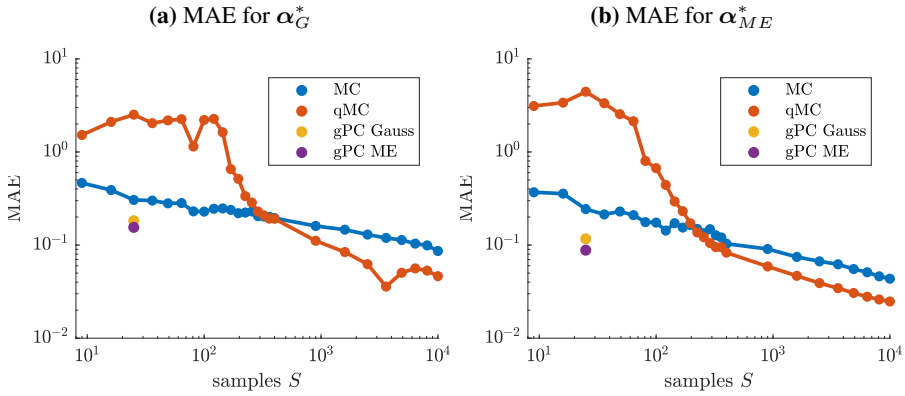


Figure 3.17: Influence of sample density on estimation accuracy of $\log \mathcal{L}_l(\boldsymbol{\alpha}^*)$.

Table 3.2: Summary of log-Likelihood estimation.

	$\log \mathcal{L}$	PDF est.	MAE	S_{gPC}	S_{MC}	S_{qMC}
α_G^*	176.7	Gauss	0.183	25	458	577
		ME	0.155	25	629	1171
α_{ME}^*	183.7	Gauss	0.117	25	265	370
		ME	0.088	25	383	1009

3.5 Conclusion

In this chapter we proposed and discussed an efficient numerical method that is tailored to parametric model uncertainty identification using maximum likelihood estimation. Here the objective is to find a parametric input distribution that best explains a series of observations. A novel method is proposed to determine the output distribution for given input probability model making use of the generalized Polynomial Chaos (gPC) expansion framework. We extended the gPC framework so that high-order stochastic moments can be obtained efficiently. These high-order estimates can then be fed to a Gaussian or maximum entropy distribution. This strategy enables a reduction in the required number of function evaluations to obtain an adequate estimation of the statistical moments of the output distribution. The method is applied to calibrate a static

model for the shifting time for wet clutch engagement based on a series of measurements verified in the lab. It is shown that output distribution estimations via high-order moment matching excels traditional identification methods that are based on mean and variance estimates whilst the computational cost remains the same due to efficient techniques that were developed. The high-order moment matching resulted into a $\log$-likelihood increase of about 4% since the accuracy of the estimated output probability density function could be improved up to 47% compared to Gaussian distributions. This methodology reveals high potential when scaling up the number of uncertain parameters that need to be identified through inverse uncertainty identification. In addition to the inclusion of parametric uncertainty, future work will investigate the incorporation of model structure uncertainty.

3.6 Appendix: Maximum Entropy Distributions

In statistics and information theory, the maximum entropy distribution is defined as the distribution with greatest (greatest or the same) entropy of all the element that belong to a specific class of distributions. This class of candidate distributions is usually specified by an arbitrary number of characterising statistical features or measures, such as the statistical moments. Selecting the distribution with greatest entropy is a heuristic default that yields the least-informative distribution [41].

In a more formal framework, we seek a probability density $\hat{f}$ as a model of the true density function f to which we only have access in the form of a limited number of features, say $\{\mu_k\}$. Such features are determined by taking the expectation of a number of kernel functions $\{g_k\}$. E.g. in case that we use the polynomial kernels $\{x, x^2, x^3, \dots\}$ we get the statistical moments. The maximum entropy distribution is then defined as the solution of the following optimization problem. The maximality criteria represents the entropy of the distribution f, the first equality constraint expresses that f is a density function whilst the latter K constraints represent the premised distribution features.

$$\begin{aligned} \max_f \ \mathrm{H}[f] &= -\int f(x)\log f(x)\mathrm{d}x \\ \text{s.t. } \int f(x)\mathrm{d}x &= 1 \\ \int g_k(x)f(x)\mathrm{d}x &= \mu_k,\ k \in \{1,2,\dots,K\} \end{aligned} \tag{3.13}$$

The corresponding Lagrangian L is given by

$$\begin{aligned} L[f,\boldsymbol{\lambda}] = \mathrm{H}[f] + \lambda_0 \int f(x)\mathrm{d}x - \lambda_0 + \\ \textstyle\sum_{k=1}^{K} \int \lambda_k g_k(x) f(x)\mathrm{d}x - \lambda_k \mu_k \end{aligned} \tag{3.14}$$

Nullifying the functional variation to f yields

$$\hat{f} = \exp\left(-\lambda_0 - \sum_{k=1}^{K} \lambda_k g_k\right)$$

The first equality constrained can be practised to determine λ_0. Accordingly, we can express λ_0 in function of the remaining Lagrangian multipliers

$$\lambda_0 = \log \int \exp\left(-\sum_{k=1}^{K} \lambda_k g_k\right) \mathrm{d}x$$

One may substitute this solution into the original Lagrangian (3.14) so to obtain the dual unconstrained problem which can be solved accordingly.

$$\min_{\boldsymbol{\lambda}} \Gamma(\boldsymbol{\lambda}) = \lambda_0 + \sum_{k=1}^{K} \lambda_k \mu_k$$

Ultimately, we obtain the following expression for

$$\hat{f} = \frac{\exp\left(-\sum_{k=1}^{K} \lambda_k g_k\right)}{\int \exp\left(-\sum_{k=1}^{K} \lambda_k g_k\right) \mathrm{d}x} \tag{3.15}$$

A solution exists for any K. However, in order to capture fourth order asymmetric effects (i.e. bimodal densities), the number of features should ≥ 4 so that the polynomial in the exponent can have two local maxima.

Due to the recurrence of this problem and the specific problem structure, there exist specialized algorithms that are tailored to it. We used an implementation made available on MATLAB file exchange.

3.7 Appendix: Earth Mover's Distance

The Earth Mover's distance (EMD) is a metric for the dissimilarity between two signatures that are each a compact representation of a distribution. In formal mathematics the metric is known as the Wasserstein distance between two probability distributions. The metric was coined in computer science due to an apparent analogy to its definition that can be modeled as the solution to a transportation problem [42].

Given are two signatures $P = \{(u_i, \mathbf{x}_i)\}_{i=1}^{n}$ and $Q = \{(v_j, \mathbf{y}_j)\}_{j=1}^{m}$. An insightful interpretation is that u_i and v_j represent the amount of dirt at the respective positions $\mathbf{x}_i$ and $\mathbf{y}_j$. The EMD between the signatures P and Q is defined as the minimal (normalized) work required to reconfigure P into Q

moving around the dirt. Formally that is

$$\begin{aligned} \mathrm{EMD}(P,Q) = \min_{F=\{f_{ij}\}} \quad & \frac{\sum_{ij} f_{ij} d(\mathbf{x}_i, \mathbf{y}_j)}{\sum_{ij} f_{ij}} \\ \text{subject to} \quad & \textstyle\sum_i f_{ij} \le v_j \\ & \textstyle\sum_j f_{ij} \le u_i \\ & \textstyle\sum_{ij} f_{ij} = \min\left\{\sum_i u_i, \sum_j v_j\right\} \\ & f_{ij} \ge 0 \end{aligned}$$

where f_{ij} represents the dirt transferred from position $\mathbf{x}_i$ to $\mathbf{y}_j$, and, $d(\mathbf{x}_i, \mathbf{y}_j)$ the distance to be covered for that transport. The transportation problem above is a special linear programming problem.

3.8 Appendix: Shifting Time Model

The shifting time is derived from a dynamic simulation of the clutch engagement. Prior research was on the building of a dynamic model of the engagement process based on first principle physics. Model parameter values were then identified via experimental system identification (from isolated subsystem testing....). The exact model identification is not in the scope of this chapter and a general overview is given. We refer to [3, 6, 43, 44] for further reading. Figure 3.18 illustrates the simplified dynamical scheme of the wet clutch system. The motor is controlled towards a constant speed ω_m delivering a torque

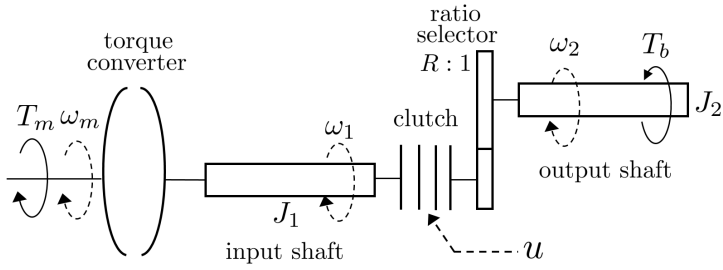


Figure 3.18: Dynamic torque equilibrium scheme.

T_m to the system. The motor is connected with a torque converter that drives the input shaft of the clutch with torque T_1 at rotational velocity ω_1. The system can be modeled using a nonlinear torque ratio function $\frac{T_1}{T_m} = f_t(\frac{\omega_1}{\omega_m})$ and a capacity factor function $\frac{\omega_m}{\sqrt{T_m}} = f_c(\frac{\omega_1}{\omega_m})$ provided by the manufacturer. Consequently, the driving torque T_1 of the input shaft can be written as

$$T_1(\omega_m, \omega_1) = \omega_m^2 \frac{f_t(\frac{\omega_1}{\omega_m})}{f_c(\frac{\omega_1}{\omega_m})^2}.$$

The clutch is connected to a ratio selector with gear ratio $R = \frac{\omega_{in}}{\omega_{out}}$. Thereafter, the output shaft operates at ω_2 as result of the driving torque T_2. Furthermore, the rotational movement of the output shaft and flywheel, characterized by inertia J_2, are counteracted by an additional brake torque T_b.

$$T_b(\omega_2) = T_{b0} + b\omega_o$$

The system is fully engaged when both shafts have an equivalent speed $\omega_1 = R\omega_2$. The dynamics of this engagement process can be described by a succession of two distinct phases.

Asynchronous Phase

The first phase is characterized by a speed difference $\omega_1 > R\omega_2$ of the clutch shafts. During this phase the hydraulic piston chamber starts filling up. Consequently, the pressure p_{hc} in the hydraulic chamber starts to increase as can be seen Figure 3.19a. This behavior is captured by a second order dynamic system for which the parameters are empirically fitted [6, 43]. Mark that the control input u is the feedforward current signal of the valve as illustrated in Fig. 3.7.

$$\ddot{p}_{hc} = a_1\dot{p}_{hc} + a_2 p_{hc} + b_1 u + b_2\dot{u} + a_0 \tag{3.16}$$

The piston movement as result of the increasing pressure is modeled by a first order dynamic system. The state variable $\tilde{z}$ can easily be scaled and truncated within a feasible area to obtain the piston position $z \in [0, z_M]$, for which z_M is the position that assures full contact between the friction plates.

$$\dot{\tilde{z}} = c_1(p_{hc} - p_{hc_0}) + c_2\dot{p}_{hc} + c_3\tilde{z} \tag{3.17}$$

During the filling phase the plates do not make contact. However, the oil between the friction plates will behave as a planar Couette flow. The resulting shear stresses within the fluid will cause an initial torque transmission and acceleration of the load speed ω_2. This phenomena is characterized by the constant γ, capturing the geometry of the plates and the fluid viscosity. A second influence on the torque transmission is the pressure on the plates p. This pressure is a fraction of the total hydraulic overpressure p_{hc} as is illustrated in Fig. 3.19a. There is assumed that the pressure build-up on the plates initiates when the piston reaches a constant threshold z_p.

$$p = \max(0, \frac{z - z_p}{z_M - z_p})p_{hc} \tag{3.18}$$

The torque transfer due to plate pressure is modeled with parameter α, based on the geometry of the plates and fluid characteristics. A naive approach to

model the transferred clutch torque T_c would be to switch between Couette flow and plate friction once z equals z_p in a discrete way. However, the sudden torque shift would cause unrealistic behavior. Therefore, a transient function $\delta_t \in [0, 1]$ is defined (Fig. 3.19b) to have a smooth transition between the aforementioned sources of torque transfer.

$$T_c(\omega_1, \omega_2, p, z) = \delta_t(z) \cdot \alpha p + (1 - \delta_t(z)) \cdot \frac{\gamma}{z_M - z} \cdot (\omega_1 - R\omega_2)$$

The expression for the counteracting clutch torque T_c on the input shaft holds when $\omega_1 > R\omega_2$. The driving torque of the output shaft equals $T_2 = RT_c$ and is illustrated in Fig. 3.20a. Hence, one can formulate the nonlinear system dynamics during the asynchronous phase by considering the torque equilibrium of both shafts.

$$\begin{aligned} J_1\dot{\omega}_1 &= T_1(\omega_m, \omega_1) - T_c(\omega_1, \omega_2, p, z) \\ J_2\dot{\omega}_2 &= RT_c(\omega_1, \omega_2, p, z) - T_b(\omega_2) \end{aligned} \tag{3.19}$$

.

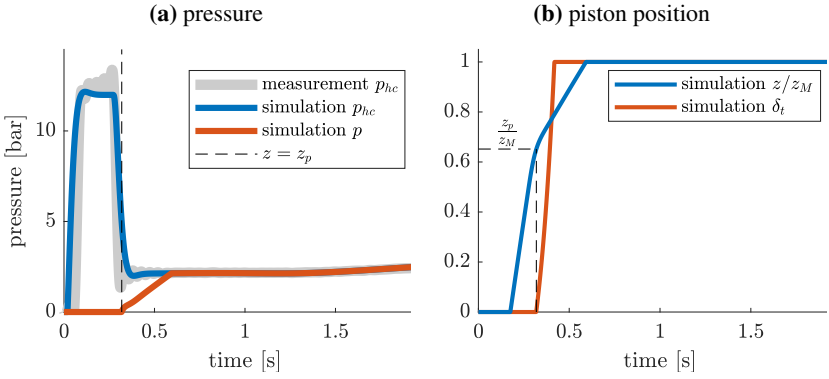


Figure 3.19: Illustration of overpressure and resulting piston position.

Synchronous Phase

The system shifting process arrives in the synchronous phase once an equivalent speed for both shafts is obtained ($\omega_1 = R\omega_2$). Henceforth, the shafts can be considered as fully engaged and the system becomes independent of the control signal u. The driving torque of the output shaft equals $T_2 = RT_1$ since perfect torque transmission is assumed (Fig. 3.20a).

$$(J_1 + \frac{J_2}{R^2})\dot{\omega}_1 = T_1(\omega_m, \omega_1) - \frac{1}{R}T_b\left(\frac{\omega_1}{R}\right) \tag{3.20}$$

Shifting Time

The purpose of this dynamical model is to determine the shifting time, which is key for various applications of the wet clutch. This is determined by measuring the time interval between initialization of the feedforward control signal and the moment that both shafts obtain the same angular velocity (note that the mutual speed does not need to equal the initial input velocity). Fig. 3.20b illustrates the acceleration of the output shaft ω_2 towards a synchronous speed with the input shaft. The figure clearly illustrates the discrepancy between the measured shifting time y_l and corresponding simulation Y_l.

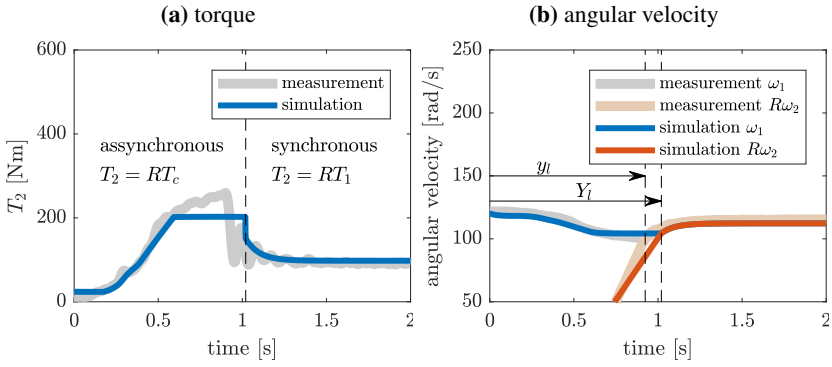


Figure 3.20: Comparison of dynamic simulation and signal measurements of a clutch engagement from neutral to first gear.

References

[1] S. Foulard, S. Rinderknecht, M. Ichchou, and J. Perret-Liaudet. Automotive drivetrain model for transmission damage prediction. Mechatronics, 30:27–54, 2015.

[2] A. Della Gatta, L. Iannelli, M. Pisaturo, A. Senatore, and F. Vasca. A survey on modeling and engagement control for automotive dry clutch. Mechatronics, 55:63–75, 2018.

[3] A. P. Ompusunggu, P. Sas, and H. Van Brussel. Modeling and simulation of the engagement dynamics of a wet friction clutch system subjected to degradation: An application to condition monitoring and prognostics. Mechatronics, 23(6):700–712, 2013.

[4] B. Depraetere, G. Pinte, W. Symens, and J. Swevers. A two-level iterative learning control scheme for the engagement of wet clutches. Mechatronics, 21(3):501–508, 2011.

[5] A. Dutta, Y. Zhong, B. Depraetere, K. Van Vaerenbergh, C. Ionescu, B. Wyns, G. Pinte, A. Nowe, J. Swevers, and R. De Keyser. Model-based and model-free learning strategies for wet clutch control. Mechatronics, 24(8):1008–1020, 2014.

[6] W. D. Widanage, J. Stoev, A. Van Mulders, J. Schoukens, and G. Pinte. Nonlinear system-identification of the filling phase of a wet-clutch system. Control Engineering Practice, 19(12):1506–1516, 2011.

[7] M. Watson, C. Byington, D. Edwards, and S. Amin. Dynamic modeling and wear-based remaining useful life prediction of high power clutch systems. Tribology Transactions, 48(2):208–217, 2005.

[8] V. Stojanovic and N. Nedic. Robust kalman filtering for nonlinear multivariable stochastic systems in the presence of non-gaussian noise. International Journal of Robust and Nonlinear Control, 26(3):445–460, 2016.

[9] V. Stojanovic, N. Nedic, D. Prsic, and L. Dubonjic. Optimal experiment design for identification of arx models with constrained output in non-gaussian noise. Applied Mathematical Modelling, 40(13-14):6676–6689, 2016.

[10] Y. Touati, R. Merzouki, and B. O. Bouamama. Robust diagnosis to measurement uncertainties using bond graph approach: Application to intelligent autonomous vehicle. Mechatronics, 22(8):1148–1160, 2012.

[11] L. Marconi and R. Naldi. Aggressive control of helicopters in presence of parametric and dynamical uncertainties. Mechatronics, 18(7):381–389, 2008.

[12] M. Abdeetedal, H. Rezaee, H. A. Talebi, and F. Abdollahi. Optimal adaptive jacobian internal forces controller for multiple whole-limb manipulators in the presence of kinematic uncertainties. Mechatronics, 53:1–7, 2018.

[13] A. Hajiloo, N. Nariman-Zadeh, and A. Moeini. Pareto optimal robust design of fractional-order pid controllers for systems with probabilistic uncertainties. Mechatronics, 22(6):788–801, 2012.

[14] P. J. Bickel and K. A. Doksum. Mathematical statistics: basic ideas and selected topics, volume I, volume 117. CRC Press, 2015.

[15] M. Stein. Large sample properties of simulations using latin hypercube sampling. Technometrics, 29(2):143–151, 1987.

[16] R. Caflisch. Monte carlo and quasi-monte carlo methods. Acta numerica, 7:1–49, 1998.

[17] D. Xiu and G. Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. SIAM journal on scientific computing, 24(2):619–644, 2002.

[18] D. Xiu and G. Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. Journal of computational physics, 187(1):137–167, 2003.

[19] D. Xiu and J. Hesthaven. High-order collocation methods for differential equations with random inputs. SIAM Journal on Scientific Computing, 27(3):1118–1139, 2005.

[20] D. Xiu. Efficient collocational approach for parametric uncertainty analysis. Commun. Comput. Phys, 2(2):293–309, 2007.

[21] G. Blatman and B. Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. Journal of Computational Physics, 230(6):2345–2367, 2011.

[22] K. O. Bowman and L. R. Shenton. Estimation: Method of moments. Encyclopedia of statistical sciences, 3, 2004.

[23] J. Munkhammar, L. Mattsson, and J. Rydén. Polynomial probability distribution estimation using the method of moments. PloS one, 12(4):e0174573, 2017.

[24] R. Fonseca, M. Friswell, J. Mottershead, and A. Lees. Uncertainty identification by the maximum likelihood method. Journal of Sound and Vibration, 288(3):587 – 599, 2005. Uncertainty in structural dynamics.

[25] J. Nelder and R. Mead. A simplex method for function minimization. The computer journal, 7(4):308–313, 1965.

[26] R. Oeuvray and M. Bierlaire. BOOSTERS: A derivative-free algorithm based on radial basis functions. International Journal of Modelling and Simulation, 29(1):26–36, 2009.

[27] R. Wild, S.and Regis and C. Shoemaker. ORBIT: Optimization by radial basis function interpolation in trust-regions. SIAM Journal on Scientific Computing, 30(6):3197–3219, 2008.

[28] T. Lefebvre, F. De Belie, and G. Crevecoeur. A radial basis function based optimization algorithm with regular simplex set geometry in ellipsoidal trust-regions. arXiv preprint arXiv:1805.11830, 2018.

[29] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. The Journal of Machine Learning Research, 15(1):949–980, 2014.

[30] N. Hansen. The cma evolution strategy: A tutorial. arXiv preprint arXiv:1604.00772, 2016.

[31] C. Chan and G. Liu. Hysteresis identification and compensation using a genetic algorithm with adaptive search space. Mechatronics, 17(7):391–402, 2007.

[32] M. Affenzeller, S. Wagner, S. Winkler, and A. Beham. Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. Numerical Insights. CRC Press, 2009.

[33] A. Kaintura, T. Dhaene, and D. Spina. Review of polynomial chaos-based methods for uncertainty quantification in modern integrated circuits. Electronics, 7(3):30, 2018.

[34] P. Glaser, M. Schick, K. Petridis, and V. Heuveline. Comparison between a polynomial chaos surrogate model and markov chain monte carlo for inverse uncertainty quantification based on an electric drive test bench. In ECCOMAS Congress, volume 2016, 2016.

[35] R. Ghanem and P. Spanos. Stochastic finite elements: a spectral approach. Courier Corporation, 2003.

[36] R. Ghanem and P. Spanos. Stochastic finite element method: Response statistics. In Stochastic Finite Elements: A Spectral Approach, pages 101–119. Springer, 1991.

[37] J. Witteveen and H. Bijl. Modeling arbitrary uncertainties using gram-schmidt polynomial chaos. In 44th AIAA aerospace sciences meeting and exhibit, page 896, 2006.

[38] S. Lee and W. Chen. A comparative study of uncertainty propagation methods for black-box-type problems. Structural and Multidisciplinary Optimization, 37(3):239, 2009.

[39] M. Rajabi, B. Ataie-Ashtiani, and C. Simmons. Polynomial chaos expansions for uncertainty propagation and moment independent sensitivity analysis of seawater intrusion simulations. Journal of Hydrology, 520:101–122, 2015.

[40] H. Najm. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. Annual review of fluid mechanics, 41:35–52, 2009.

[41] L. Mead and N. Papanicolaou. Maximum entropy in the problem of moments. Journal of Mathematical Physics, 25(8):2404–2417, 1984.

[42] Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. International journal of computer vision, 40(2):99–121, 2000.

[43] G. Tod, W. De Groote, T. Lefebvre, N. De Geeter, B. Depraetere, G. Crevecoeur, and S. Van Poppel. Parametric uncertainty quantification using polynomial chaos expansion applied to a wet clutch model. International Journal of Modeling and Optimization, 9(4):185–191, 2019.

[44] S. Iqbal, F. Al-Bender, A. P. Ompusunggu, B. Pluymers, and W. Desmet. Modeling and analysis of wet friction clutch engagement dynamics. Mechanical Systems and Signal Processing, 60:420–436, 2015.

Chapter 4

Neural Network Augmented Physics Models for Mechatronic Systems with Partially Unknown Dynamics

Chapter 3 introduced the insurmountable mismatches between a physics-inspired system model and data within a complex mechatronic application. In nonlinear servo applications such as a slider-crank mechanism we can observe force interactions that are hard to model. Instead of introducing a stochastic nature to the unmodeled interactions, in this chapter, we present a hybrid model that captures the unmodeled phenomena by a neural network model. More specifically, the hybrid model is validated on measurements of a slider-crank mechanism for which the nonlinear spring force and friction interactions are excluded from the physics-model and, thus, replaced by the neural network. My contributions can be summarized as follows:

- A neural network augmented physics (NNAP) model is developed that encompasses both physics-inspired and neural layers. The physical and neural parameters are simultaneously optimized solely by using the control input and state measurements, enabling for a compensated physics model that provides accurate predictions of the system behavior.
- Compared to other research, we focused on the interpretability of the converged NNAP model. This way, we retrieved valuable insights about the friction phenomena and spring force, which were initially not modeled nor measured. Moreover, the joint optimization of physical and neural parameters is substantiated by an extensive sensitivity analysis that elaborates on the identifiability and uniqueness of the obtained solutions.

Neural Network Augmented Physics Models for Systems with Partially Unknown Dynamics: Application to Slider-Crank Mechanism

W. De Groote, E. Kikken, E. Hostens, S. Van Hoecke, and G. Crevecoeur

Accepted in "IEEE/ASME Transactions on Mechatronics", 2021

Abstract *Dynamic models of mechatronic systems are abundantly used in the context of motion control and design of complex servo applications. In practice, these systems are often plagued by unknown interactions, which make the physics-based relations of the system dynamics only partially known. This chapter presents a neural network augmented physics (NNAP) model as a combination of physics-inspired and neural layers. The neural layers are inserted in the model to compensate for the unmodeled interactions, without requiring direct measurements of these unknown phenomena. In contrast to traditional approaches, both the neural network and physical parameters are simultaneously optimized, solely by using state and control input measurements. The methodology is applied on experimental data of a slider-crank setup for which the state dependent load interactions are unknown. The NNAP model proves to be a stable and accurate modeling formalism for dynamic systems that ab initio can only be partially described by physical laws. Moreover, the results show that a recurrent implementation of the NNAP model enables improved robustness and accuracy of the system state predictions, compared to its feed-forward counterpart. Besides capturing the system dynamics, the NNAP model provides a means to gain new insights by extracting the neural network from the converged NNAP model. In this way, we discovered accurate representations of the unknown spring force interaction and friction phenomena acting on the slider mechanism.*

4.1 Introduction

Servo-controlled systems face increasingly demanding performance and efficiency requirements in industrial and manufacturing applications. In, for example, presses, pumps and compressors, linear motion must be realized in direct drive or indirectly. The latter servo systems need to drive a load system via a rotary motor system and mechanical transmissions such as gears, cam-follower, and bar linkages. Position and speed control are critical, and their performance is limited due to unknown dynamics resulting from the external loads and uncertainties in the mechanical system such as inertia, backlash, damping, and friction [1–3]. To better understand these intricate and often

highly nonlinear dynamics, extensive research has been conducted on developing predictive models that capture the overall behavior of servo systems. These models can then be used by a user or manufacturer to further improve the mechatronic system design [4, 5]. Furthermore, predictive models can be inserted in model-based control strategies to improve the mechatronic system's operations, (e.g. [6]).

Capturing the servo drive system dynamics is, however, often cumbersome and challenging because we face against limits of various existing modeling and system identification formalisms. The modeling of a mechatronic system can be based on expert knowledge where partial and/or ordinary differential equations are distilled from the system. These physics-based models include (lumped) physical parameters that relate directly to the actual mechatronic system's behavior. Parameter identification techniques can then be used to identify the values of these parameters based on experimental sensor data [7]. Uncertainties can, however, still remain because servo drive systems face nonlinearities such as friction forces that give rise to disturbances that are difficult to model. Alternatively, black-box models can be built in a data-driven manner. In the context of nonlinear system identification, several methodologies exist such as Gaussian processes [8], Hammerstein-Wiener structures [9], and nonlinear auto-regressive models (e.g. NARMAX) [10]. Supervised learning is typically used to build the model by relating input to output data. Supervised machine learning techniques such as deep learning have become pervasive because they have the ability to recognize patterns in data [11]. Neural networks have, for instance, been used to approximate various nonlinear relations, such as inverse kinematics within complex mechatronic systems [12]. Recurrent neural networks (RNN) have the ability to capture the nonlinear dynamics in time series by means of an internal state and are of particular interest for learning the dynamic behavior of mechatronic applications [13]. Ensembles of interconnected RNNs have demonstrated their usefulness in the modeling of complex dynamical systems [14]. However, as observed in [15], there is no physical interpretation given to this internal state, which makes the convergence rely substantially on hyperparameter tuning and initialization choices.

Having physical interpretations to the modeling can, however, be useful, particularly to assure robust and physics consistent predictions. Physically interpretable neural-fuzzy networks have shown in this respect enhanced explainability by assigning piecewise linear models to the network nodes [16]. The dynamic behavior is typically defined by the so-called derivative function in a partial and/or ordinary differential equations-based model. Based on physical insights, it is possible to deduce an appropriate structure of a black-box model. Feedforward neural networks have been proposed in physics-informed deep learning [17] to approximate derivative functions of dynamical systems that, in contrast to traditional black-box approximations, return more inter-

pretable models. Other recent approaches based on symbolic regression [18] and sparse regression [19] were able to return more interpretable models by discovering the underlying governing equations. Furthermore, recent developments indicate the possibilities to approximate reformulations of classical mechanics, such as Lagrangian [20] and Hamiltonian [21] mechanics by neural networks.

Next to using physical insights to deduce the structure of black-box models, one can directly use physics-based governing equations within the data-driven model. Physical laws can be included in the loss function of black-box neural networks to guide the training process towards physical consistent model predictions [22]. Conversely, neural network mappings can assist in compensating for prediction discrepancies of a complete physics-based model [15, 23]. These neural networks serve as nonlinear mappings that transform the state estimations behavior of the physics-based model. They enable predictions to be improved; however, they do not represent an interpretable physical phenomenon. An incomplete physics model was considered in [24] that was complemented with neural networks. The latter was engaged to accommodate for partially unknown dynamics within the physical system. Neural networks were trained separately and in a supervised manner on the basis of measurement data that directly relate to the unknown phenomena.

This chapter proposes a neural network augmented physics (NNAP) model, encompassing trainable physical layers closely connected to a neural network to compensate for partially unknown dynamics. It is assumed that no direct measurement data on the unknown phenomena are available because they are either difficult or not possible at all to measure. Examples are unknown friction phenomena in servo mechanisms that affect the overall system behavior. The proposed NNAP models consist of an interconnected network of physics layers complemented with black-box layers. Measurement data are propagated through these layers to simultaneously identify the physics and data-driven parameters, as depicted in Fig. 4.1. The effectiveness of the methodology is extensively tested on experimental data collected on a servo slider-crank setup with unknown nonlinear load and unidentified key physical parameters. The objective of the presented methodology is twofold. First, to obtain a model with improved predictive capabilities by performing a joint optimization of physical and neural parameters while only using state and control input data. Secondly, to explain the partially unknown dynamics within the system by analyzing the information captured by the black box layers after training the NNAP model. By using such an approach, new physical insights can be obtained to explain and interpret the unknown phenomena that are inaccessible by direct measurements or are difficult to model *ab initio*.

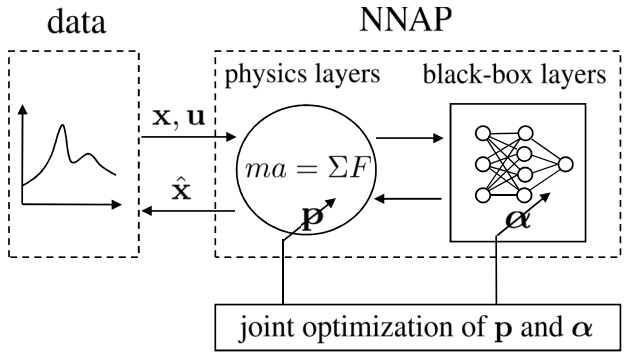


Figure 4.1: NNAP model as an interconnected network of physics and black-box layers, which are defined by trainable physics parameters $\mathbf{p}$ and neural parameters $\boldsymbol{\alpha}$, respectively. All layers are simultaneously optimized solely by using control inputs $\mathbf{u}$ and state measurement $\mathbf{x}$.

4.2 Slider-Crank System

Various industrial applications require reciprocating linear motions. Motion control of these systems can be provided by means of linear electric motors [25]. For high-power applications, a combination of rotary motors in combination with mechanical transmissions [26] such as gears, cam-follower and bar linkage systems are the preferred driveline because they achieve energy efficient and robust operation. These drivelines exhibit highly nonlinear behavior that is often challenging to capture using first-principle physics modeling. They are often plagued by unidentified load disturbances and unknown interactions of the system with the environment that are beyond expert knowledge. These nonlinear phenomena are, henceforth, formalized by relation $\mathcal{P}$ that relates certain inputs of the system (e.g. velocity) to a response of the system (e.g. friction force). Next to these nonlinear unknown phenomena, physical parameters such as the inertia in the servo drive system can be uncertain and are formalized as $\mathbf{p}$.

This chapter applies and aims at validating a methodology to accommodate for unknown $\mathcal{P}$ and $\mathbf{p}$ that can arise in servo-controlled systems. We consider a slider-crank mechanism translating a rotary motion into a linear displacement that is subject to an unidentified load. This application is of direct relevance in various industrial applications such as compressors [27], hydraulic pumps [28], weaving looms [29], and presses [30]. A system model comprehending the dynamics of the slider-crank system is built, and experimental data are collected from a practical setup.

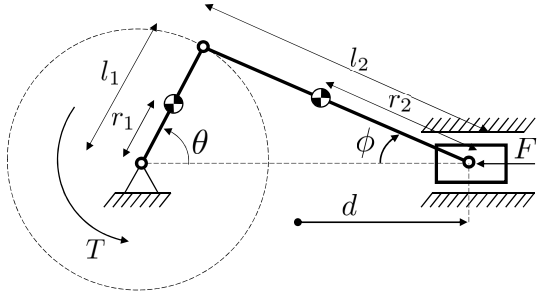


Figure 4.2: Schematic of slider-crank system.

4.2.1 System Model

The slider-crank system considered in this chapter is schematically depicted in Fig. 4.2. The setup consists of a rotary servo motor delivering torque T to a first mechanical link that rotates with angle θ, which, in turn, is connected to a second link. The latter link is connected to a slider (with relative angle ϕ between this link and slider). The dynamics of the considered slider-crank setup are governed by force interactions between the rigid mechanical links. Appendix 4.6 details the expressions that formalize the dynamics of the multi-body system. A time invariant state-space model can be distilled that captures the dynamics of the state $\mathbf{x} = \begin{bmatrix}\theta & \omega\end{bmatrix}^T$. The angular speed is denoted as $\omega \equiv \dot{\theta}$. Note that the slider position d is geometrically coupled to motor angle θ and angle ϕ that, in turn, is related to θ via $\phi = \arcsin\left(\frac{l_1}{l_2}\sin(\theta)\right)$. The non-linear state space relation can be comprehended by a derivative function that relates the state $\mathbf{x}$ and control input $u = T$ to the next state. We, furthermore, parameterize the state space model with physical parameters $\mathbf{p}$, being masses, lengths, inertia, and rotational damping coefficient, which appear in (4.11). Finally, the load force F depicted in Fig. 4.1 is the unknown nonlinear relation $\mathcal{P}$ that is an additional disturbance input of the model. Given the above, we formalize the dynamics in the slider-crank, thus, as $\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, u, F; \mathbf{p}\right)$ with a geometrical relation between the linear displacement and velocity

$$\boldsymbol{\gamma}(\mathbf{x}; \mathbf{p}) = \begin{bmatrix} d \\ v \end{bmatrix} = \begin{bmatrix} l_1 \cos(\theta) + l_2 \cos(\phi) - (l_2 - l_1) \\ -l_1 \sin(\theta)\omega - l_2 \sin(\phi)\dot{\phi} \end{bmatrix} \tag{4.1}$$

that is again dependent on the physical parameters $\mathbf{p}$.

4.2.2 Practical Setup

The practical slider-crank setup is depicted in Fig. 4.3. A 3 kW brushless servo motor with integrated drive is connected to a rigid crank having length $l_1 = 0.05\ \mathrm{m}$. The slider mechanism is connected to the crank via a rigid connecting

rod with approximate length $l_2 = 0.29$ m. The rotary motion of the motor is measured by an incremental encoder (8192 CPR). A compression spring is added to invoke additional nonlinear disturbances in the external force F acting on the slider. The spring force characteristic as a function of displacement d is given in Fig. 4.4. This characteristic was empirically determined for validation purposes later on.

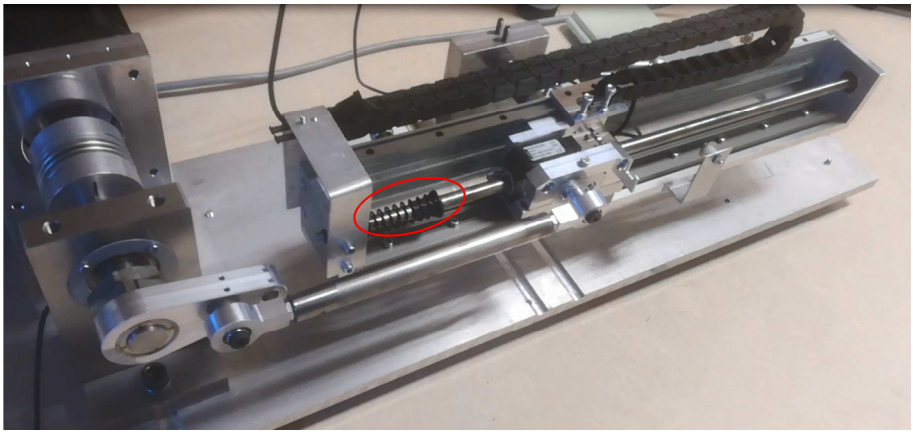


Figure 4.3: Slider-crank setup with spring load indicated by red circle.

Experimental data are collected from the slider-crank setup for different torque signal inputs. These torque profiles, see Fig. 4.5, are chosen *ad-hoc* to have diverse excitations of the system. These torque inputs are each time applied starting from two distinct motor positions, leading to a total of $S = 20$ different trajectories of rotational movements θ and ω. The measurements are sampled at 2000 Hz, resulting in $L = 800$ samples per trajectory.

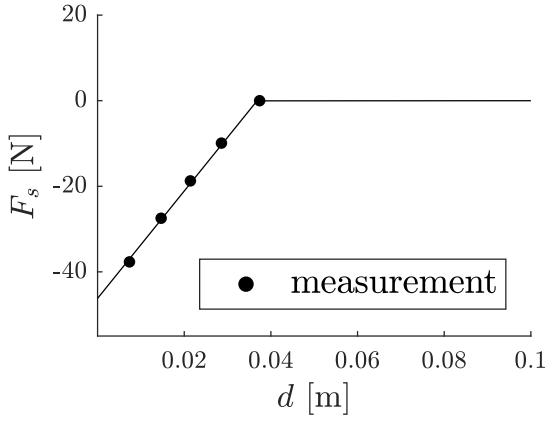


Figure 4.4: Spring characteristic.

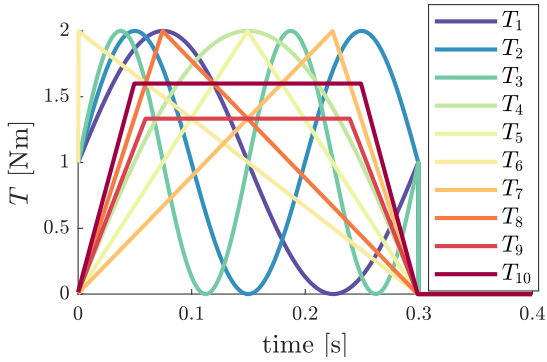


Figure 4.5: Torque signals.

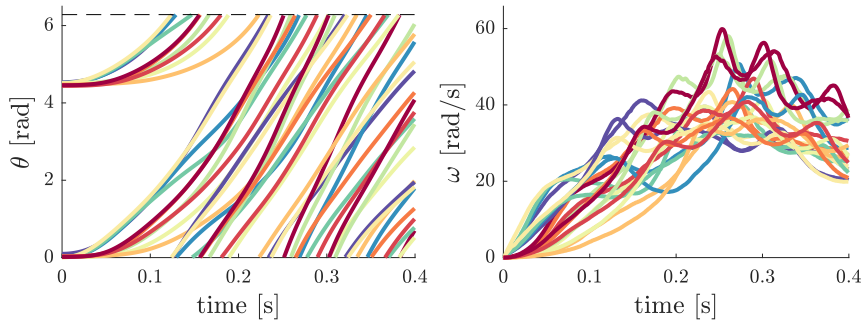


Figure 4.6: Collected data from the slider-crank setup.

4.3 Neural Network Augmented Physics Model

Having accurate predictive models of systems, such as the slider-crank setup presented in Section 4.2, can advance the design and motion control of various manufacturing and industrial systems. To accommodate for partially unknown dynamics (unknown nonlinear disturbance $\mathcal{P}$ and uncertain physical parameters $\mathbf{p}$) arising in servo systems, the physics-based ordinary differential equations (ODE) are complemented with additional data-driven models. The physics-based derivative function $\mathbf{f}$ defines the behavior of the system state $\mathbf{x}$ for given control input $\mathbf{u}$. The known physical laws are encapsulated within $\mathbf{f}$ and characterized by physical parameters $\mathbf{p}$. The lack of knowledge about the unknown phenomena $\mathcal{P}$ is compensated for by introducing an additional input $\mathbf{z}$.

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}, \mathbf{u}, \mathbf{z}; \mathbf{p}\right) \tag{4.2}$$

The NNAP model here includes an artificial neural network $\boldsymbol{\eta}$ with the weights and biases of the neural nodes as learnable parameters $\boldsymbol{\alpha}$ to compensate for these unknown phenomena $\mathcal{P}$. The state measurements $\mathbf{x}$ and control input $\mathbf{u}$ are used to simultaneously update the neural network variables $\boldsymbol{\alpha}$ and physical

parameters $\mathbf{p}$ to match the actual physical dynamical system behavior. Note that here, contrary to [24], the data-driven model is not learned on the basis of (additional) measurement data that relate directly to the unknown phenomena $\mathcal{P}$. A feedforward and a recurrent physics-based neural network model formalism are presented in the two next subsections, respectively. Subsequently, the optimization process is explained, followed by the implementation details on the slider-crank mechanism.

4.3.1 Feedforward NNAP Model

A feedforward model predicts the next state $\mathbf{x}_{k+1}$ for given control input $\mathbf{u}_k$ and current state $\mathbf{x}_k$. By repeatedly feeding the predicted state back as input during the subsequent time instance, the model becomes suitable for multistep predictions. The universal approximation theorem states that feedforward neu-

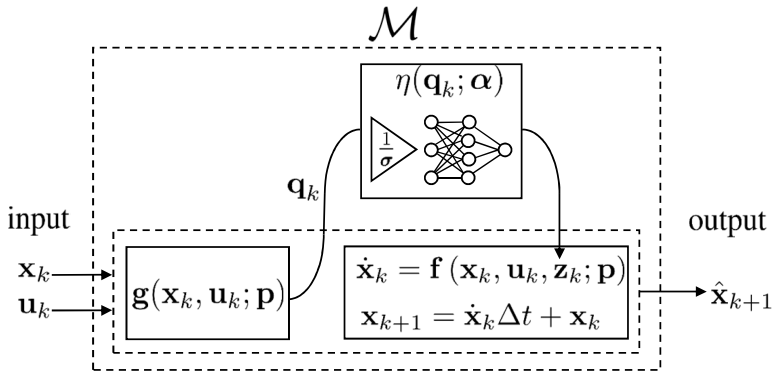


Figure 4.7: Feedforward NNAP modeling formalism.

ral networks with one hidden layer can approximate any continuous function for inputs within a specific range [31]. More specifically, we define $\boldsymbol{\eta}$, a one-hidden-layer rectified linear unit (ReLU) network of n_h hidden units predicting a n_o dimensional output. A ReLU model is chosen due to its demonstrated usefulness in various regression tasks [11,32]. The n_i dimensional input $\mathbf{q} \in \mathbb{R}^{n_i}$ to the neural network is defined by a physics-inspired network layer $\mathbf{g}$.

$$\mathbf{q} = \mathbf{g}(\mathbf{x}, \mathbf{u}; \mathbf{p}) \tag{4.3}$$

The input vector $\mathbf{q}$ is scaled by the standard deviation ($\sigma_j,\ j = 1, \cdots, n_i$) of the corresponding input elements to eliminate magnitude differences and consequently avoid dominance of particular dimensions. The corresponding transformation matrix $\Sigma = \text{diag}\left(\sigma_1^{-1}, \ldots, \sigma_{n_i}^{-1}\right)$ is determined in advance directly from the measurement data. Subsequently, the input undergoes a linear

transformation by the weights $\mathrm{W}_h \in \mathbb{R}^{n_i \times n_h}$ of the hidden layer. The hidden unit activation function Υ will, after adding the bias vector $\mathbf{b}_h \in \mathbb{R}^{n_h}$, include the required nonlinearity in the function $\boldsymbol{\eta}$. Subsequently, the output is obtained via a linear output layer, defined by the weights $\mathrm{W}_o \in \mathbb{R}^{n_h \times n_o}$ and $\mathbf{b}_o \in \mathbb{R}^{n_o}$. We denote the ReLU model as being

$$\begin{aligned} \boldsymbol{\eta}(\mathbf{q};\boldsymbol{\alpha}) &= \mathrm{W}_o^T \Upsilon\left(\mathrm{W}_h^T \Sigma \mathbf{q} + \mathbf{b}_h\right) + \mathbf{b}_o \\ \Upsilon(\cdot) &= \max\left(0, \cdot\right) \end{aligned} \tag{4.4}$$

The architecture of the feedforward NNAP models is given in Fig. 4.7. The output of the ReLU network $\boldsymbol{\eta}$, initialized by random weights and biases included in $\boldsymbol{\alpha}$, replaces the variable $\mathbf{z}$ in the ODE layer (4.2). Consequently, the assumption is that $\mathbf{q}$ contains sufficient input information to estimate $\mathbf{z}$, by approximating $\mathcal{P}$ via a neural network $\boldsymbol{\eta}$. We distill the following derivative function at each time instant k.

$$\dot{\mathbf{x}}_k = \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\eta}(\mathbf{q}_k;\boldsymbol{\alpha}); \mathbf{p}\right) \tag{4.5}$$

The derivative function now has a hybrid nature and is further used to propagate to the next state via Euler's method by choosing Δt equal to the fixed sampling time of the measurement data. The combination of interconnected layers adds up to an overall feedforward network function $\mathcal{M}\left(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p}, \boldsymbol{\alpha}\right)$ that estimates the state $\mathbf{x_{k+1}}$ given the prior state $\mathbf{x_k}$ and input $\mathbf{u_k}$.

$$\hat{\mathbf{x}}_{k+1} = \mathcal{M}\left(\mathbf{x}_k, \mathbf{u}_k; \mathbf{p}, \boldsymbol{\alpha}\right) \tag{4.6}$$

The feedforward NNAP model $\mathcal{M}$ predicts only the state of the subsequent timestep. The model $\mathcal{M}$ encompasses a static relation that can describe dynamical behavior once prior state predictions are fed back as model input.

4.3.2 Recurrent NNAP model

Recurrent neural networks (RNN) can cope with timeseries due to their ingrained dynamic nature [11, 33]. These models are penalized for estimation errors over a larger time horizon, aiming for improved multistep prediction capabilities. Fig. 4.8 illustrates a schematic overview of a basic RNN. The operator $\mathcal{D}$ induces a one-step time delay of the internal state $\mathbf{h}$. If we unfold the RNN in time, we obtain a computational graph that contains shared weights $\boldsymbol{\psi}$ at each time instance j. The total prediction error is an accumulation of the error at each timestep. The gradients of this virtual multilayer network are required to update $\boldsymbol{\psi}$ and can be derived via backpropagation through time (BPTT), which calculates the gradients based on the chain rule of differentiation [33]. A RNN model accepts an input sequence of N steps by processing

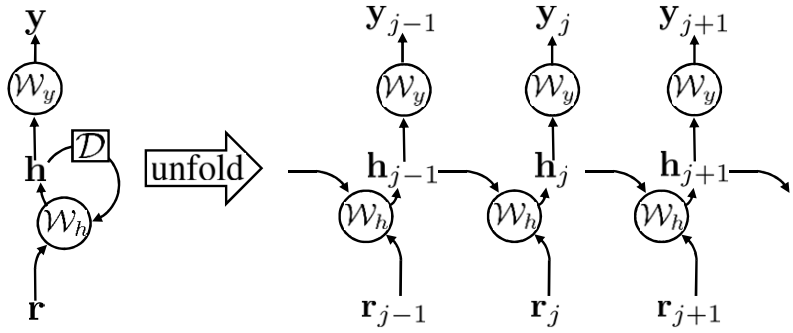


Figure 4.8: Recurrent neural network.

each input element $\mathbf{r}_j$ for $j \in \{1, \ldots, N\}$ at a time. These models contain information about prior inputs by updating an internal state vector $\mathbf{h}_j$ via a (nonlinear) function $\mathcal{W}_h$. Thereafter, an output relation $\mathcal{W}_y$ maps the internal state to the model output $\mathbf{y}$. The functions $\mathcal{W}_h$ and $\mathcal{W}_y$ typically contain network parameters $\boldsymbol{\psi}$ that are optimized via gradient-based algorithms.

$$\begin{aligned} \mathbf{h}_j &= \mathcal{W}_h(\mathbf{r}_j, \mathbf{h}_{j-1}; \boldsymbol{\psi}) \\ \mathbf{y}_j &= \mathcal{W}_y(\mathbf{h}_j; \boldsymbol{\psi}) \end{aligned} \tag{4.7}$$

A recurrent NNAP model $\mathcal{R}$ is constructed to predict a state trajectory $\{\mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+N}\}$ for given input sequence $\{\mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}\}$, starting from state $\mathbf{x}_k$. The dimension of the internal state vector $\mathbf{h}$ is typically a hyperparameter that needs to be tuned depending on the complexity of the problem. Here, we assume that the dimension of the state of $\mathbf{h}$ equals the dimension of the state vector $\mathbf{x}$. The corresponding initial value $\mathbf{h}_0$, typically initialized as zero, is chosen to equal $\mathbf{x_k}$. This allows to use the feedforward NNAP function $\mathcal{M}$, detailed in Fig. 4.7, as an update function $\mathcal{W}_h$ of the internal state $\mathbf{h}$ in (4.7). This forces the hidden state $\mathbf{h}$ to mimic the behavior of the system state $\mathbf{x}$. Consequently, the output function $\mathcal{W}_y$ in (4.7) includes an identity transformation because we can consider $\mathbf{y}_j = \mathbf{h}_j = \mathbf{x}_{k+j}$ for $j \in \{1, \ldots, N\}$. Fig. 4.9 illustrates the architecture of $\mathcal{R}$. The identification of the recurrent NNAP model $\mathcal{R}$ parameters $\boldsymbol{\psi}$ comes down to optimizing the variables $\mathbf{p}$ and $\boldsymbol{\alpha}$ included in the feedforward NNAP model $\mathcal{M}$ in (4.6).

$$\left[\hat{\mathbf{x}}_{k+1}, \ldots, \hat{\mathbf{x}}_{k+N}\right] = \mathcal{R}(\mathbf{x}_k, \mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}; \mathbf{p}, \boldsymbol{\alpha}) \tag{4.8}$$

4.3.3 Optimization

The identification of the NNAP model requires the simultaneous optimization of the neural ($\boldsymbol{\alpha}$) and physical ($\mathbf{p}$) parameters by minimizing the cost function

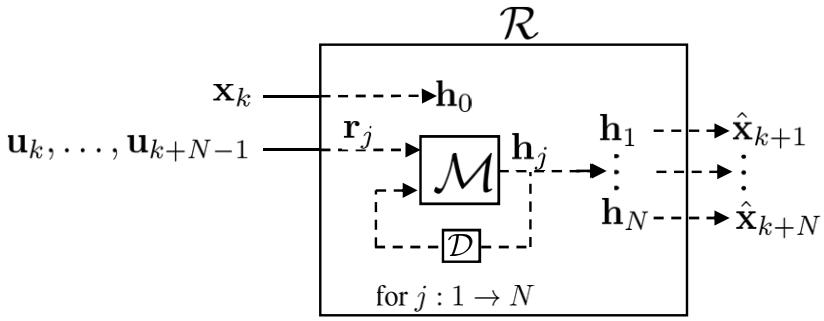


Figure 4.9: Recurrent NNAP modeling formalism.

$\mathcal{L}$.

$$\mathcal{L}(\boldsymbol{\alpha}, \mathbf{p}) = \frac{1}{S}\frac{1}{L}\frac{1}{N}\sum_{s=1}^{S}\sum_{i=0}^{L-N}\sum_{k=i+1}^{i+N}(\mathbf{x}_k^s - \hat{\mathbf{x}}_k^s)^T \mathrm{C}(\mathbf{x}_k^s - \hat{\mathbf{x}}_k^s) \tag{4.9}$$

The function $\mathcal{L}$ penalizes the discrepancies between the measured ($\mathbf{x}$) and predicted ($\hat{\mathbf{x}}$) states, scaled by a diagonal matrix C to accommodate for magnitude differences between the state variables. The loss function $\mathcal{L}$ incorporates S trajectory signals, for which each signal contains L samples of the state $\mathbf{x}$ and control input $\mathbf{u}$. Figure 4.10a illustrates the evaluation of each trajectory

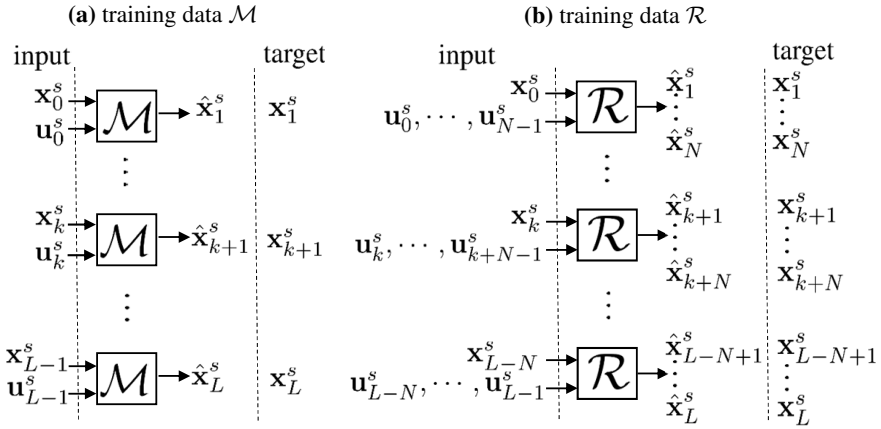


Figure 4.10: Schematic overview of training data specifications for trajectory $s \in \{1, \ldots, S\}$ existing of L training samples.

sample by $\mathcal{M}$, to match the subsequent state value (we have here $N = 1$). The recurrent model $\mathcal{R}$ is constructed on the basis of an initial state value and control sequence of N steps, as depicted in Fig. 4.10b. The derivation of the analytical gradients $\nabla_p \mathcal{L}$ and $\nabla_\alpha \mathcal{L}$ imposes that, apart from the neural network $\boldsymbol{\eta}$, the physics layers also need to be differentiable. The gradient information is used to simultaneously update $\boldsymbol{\alpha}$ and $\mathbf{p}$ based on a gradient descent approach

with step size δ.

$$\begin{aligned} \mathbf{p}_{i+1} &= \mathbf{p}_i - \delta_i \nabla_p \mathcal{L} \\ \boldsymbol{\alpha}_{i+1} &= \boldsymbol{\alpha}_i - \delta_i \nabla_\alpha \mathcal{L} \end{aligned} \tag{4.10}$$

Note that the difference between the feedforward model $\mathcal{M}$ and recurrent architecture $\mathcal{R}$ lies in the different training process as the gradients of the latter are derived by BPTT to allow robust predictions of longer time series. Nevertheless, once the recurrent model $\mathcal{R}$ converges, it can be reworked to a more simple feedforward structure $\mathcal{M}$ because both architectures ingrain the same parameters $\mathbf{p}$ and $\boldsymbol{\alpha}$.

4.3.4 Implementation

The presented NNAP modeling formalism is applied on the slider-crank mechanism detailed in Section 4.2. The general workflow is illustrated in Fig. 4.11. First, the known dynamics of the system are distilled and described by their physical laws. As mentioned before, the dynamics of the slider-crank mechanism are known, except for the state-dependent load interactions $\mathcal{P}$ acting on the slider. The incomplete physics-based relations are combined into an ODE $\mathbf{f}$ that accepts an additional input $z = F$, as explained in Appendix 4.6. Subsequently, the uncertain physical parameters $\mathbf{p}$ are defined. In the slider-crank case, we consider the parameter set $\mathbf{p} = \{J_1, B_m, m_3\}$ as unknown. Other possibilities are discussed in Section 4.4.4.

Next, the unknown load interactions $\mathcal{P}$ are replaced by a randomly initialized ReLU network η. This predictor of the load force F contains one hidden layer of $n_h = 32$ hidden units, determined via hyperparameter tuning. The network input $\mathbf{q}$ needs to provide sufficient information to the network η. Therefore, expert knowledge is used to define $\mathbf{q} = \begin{bmatrix} d & v \end{bmatrix}^T$ by choosing the input function $\mathbf{g}$ equal to the geometrical relations $\boldsymbol{\gamma}$ in (4.1). Other options for $\mathbf{g}$ and their effect are discussed in Section 4.4.1. The relations defined in the functions $\mathbf{g}(., \mathbf{p})$, $\eta(., \boldsymbol{\alpha})$ and $\mathbf{f}(., \mathbf{p})$ are implemented as successive custom network layers, that result into the overall NNAP model $\mathcal{M}(., \mathbf{p}, \boldsymbol{\alpha})$, as shown in Fig. 4.7. Moreover, this feedforward model can be implemented as a recurrent NNAP model $\mathcal{R}(., \mathbf{p}, \boldsymbol{\alpha})$ by adding the feedback loops shown in Fig. 4.9. In practice, these network architectures are implemented by using the Keras API [34] with TensorFlow [35] backend in Python. The physical laws are implemented as custom (differentiable) network layers for which the parameters $\mathbf{p}$ are defined as trainable variables. This library employs automatic differentiation, which provides the analytical expressions for the gradients $\nabla_p \mathcal{L}$ and $\nabla_\alpha \mathcal{L}$. The availability of analytical gradient information enables a joint identification of both $\mathbf{p}$ and $\boldsymbol{\alpha}$ by using a gradient based Adam optimizer that processes the data in mini-batches of 200 samples [36].

4.4 Results and Discussion

The objective of the joint optimization of $\mathbf{p}$ and $\boldsymbol{\alpha}$ is twofold. First, the prediction performances of the optimized NNAP model are assessed by analyzing the root mean squared error (RMSE) of the trajectory prediction of the key variable ω. The presented results are validated via leave-one-out-cross-validation (LOOCV). This implies that the model is trained by $S = 19$ signals and validated by the trajectory excluded from the training set. This is repeated for all 20 signals in order to develop a fair performance benchmark. Unless otherwise stated, we will perform the experiments on the feedforward architecture $\mathcal{M}$ because inherently, both architectures $\mathcal{M}$ and $\mathcal{R}$ include the same parameters to be identified. Secondly, the convergence and identifiability of the physical parameters $\mathbf{p}$ and the neural network η will be analyzed to validate if new physical insights can be gained about the physical parameters and the unknown load interactions captured by η.

4.4.1 Influence of the Neural Network Inputs

The prediction performance of the NNAP model $\mathcal{M}$ highly relies on the information that can be captured by the neural network η during the joint optimization of the parameters in $\boldsymbol{\alpha}$ and $\mathbf{p}$. The ability of η to compensate for the unmodeled physical relations in $\mathbf{f}$ depends on the input $\mathbf{q}$, defined by the function

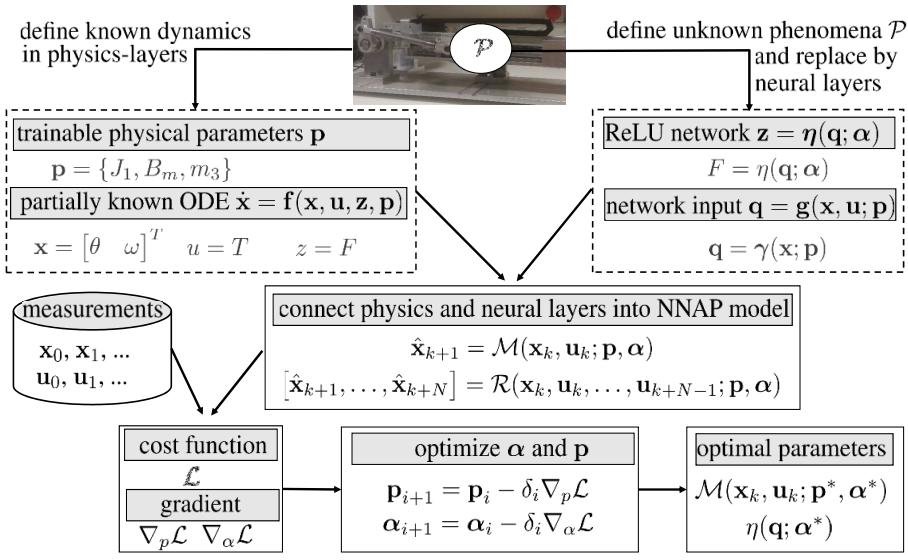


Figure 4.11: Practical implementation of the NNAP model.

$\mathbf{g}$. For this case, expert knowledge is used to define $\mathbf{g}$ as the geometrical relations $\boldsymbol{\gamma}$ in (4.1), to provide $\mathbf{q} = \begin{bmatrix} d & v \end{bmatrix}^T$ as input to η. However, in practice, the dependencies of the unknown phenomena $\mathcal{P}$ are not known. Therefore, the required information to the neural network can be determined by analyzing the obtained prediction accuracy of $\mathcal{M}$ for varying function candidates of $\mathbf{g}$. Variables such as $\{\theta, \omega\}$ and $\{T\}$ can be obtained by choosing $\mathbf{g}$ equal to an identity transformation because they are directly included in the state $\mathbf{x}$ or control input u, respectively. Figure 4.12b illustrates the time evolution of the angular velocity ω on the basis of the discrete model $\mathcal{M}$ for varying inputs $\mathbf{q}$. We perform a multistep prediction as the state $\hat{\mathbf{x}}_{k+1}$ in (4.6) is the subsequent state input $\mathbf{x}_k$ to $\mathcal{M}$ in the next time instant. Errors are propagated each time when $\mathcal{M}$ is evaluated, explaining the diverging trajectories with respect to the reference signal. The lowest average RMSE is obtained for $\mathbf{q} = \begin{bmatrix} d & v \end{bmatrix}^T$. This insight makes sense knowing that the spring load and possible additional friction phenomena, both contained in the overall force F, are not included in the physics equations and, thus, should be approximated by the network η. The set of experiments for which the required input information (i.e. set $\{d, v\}$ or the equivalently $\{\theta, \omega\}$) is not fed to the neural network η clearly lead to inferior trajectory prediction accuracies. This experiment revealed the importance of the input information to the neural network η, at the same time showing that the NNAP model $\mathcal{M}$ converges if indeed sufficient information is fed to η.

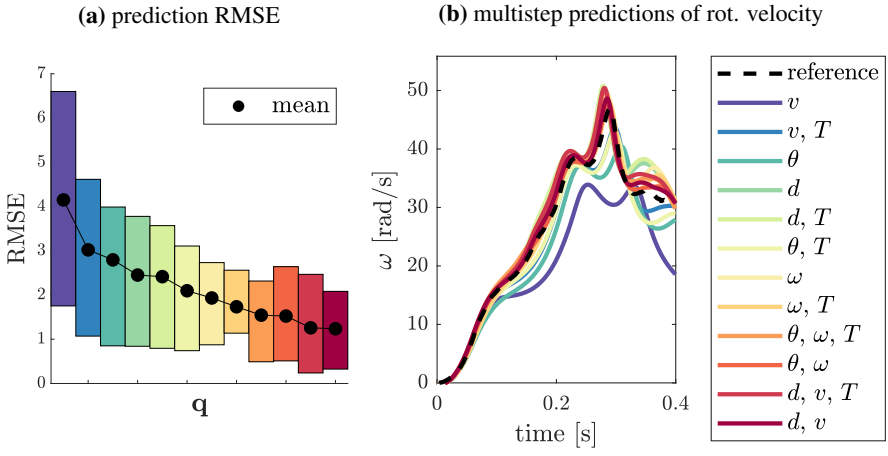


Figure 4.12: Influence of the input information included in $\mathbf{q}$. The left plot represents 90% of the most accurate results obtained on test data via LOOCV. A prediction of test signal T_9 is illustrated on the right.

4.4.2 Region of Operations

Neural networks can capture complex relations. However, as is generally known, in their current form, they fail to extrapolate well outside their training region. Physics-based relations, on the other hand, can cover a wide range of operating points because they are substantiated by general physical laws. By using physics-inspired custom layers, we alleviate the modeling complexity a neural network of the complete system needs to capture. To validate the robustness and stability of the overall NNAP model, we define the set $\mathcal{I}_1$ that includes 10 trajectories with low operating speed. The central plot in Fig. 4.13 shows the predictions of the signals operating at the highest speed, by training the NNAP model only with trajectories included in $\mathcal{I}_1$. The right plot in Fig. 4.13 illustrates the prediction performances obtained by a LOOCV experiment, covering the entire region of operations in the training set. As expected, the average RMSE indicates that the prediction accuracy deteriorates once the test signals exceed the training region. The NNAP model, however, still follows the general dynamic behavior outside the training region. These results indicate the benefit of having a solid physics-inspired basis within the dynamic NNAP model.

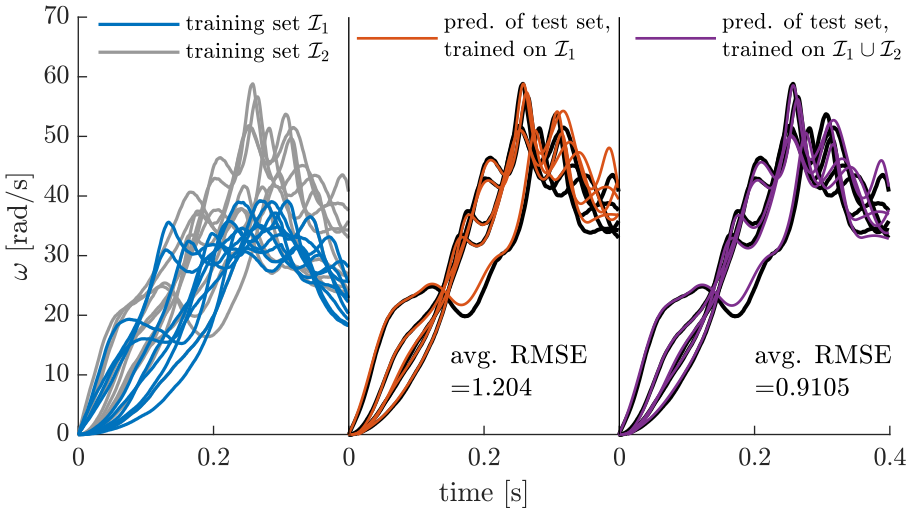


Figure 4.13: Influence of having training data in the neighborhood of test signals. The left plot indicates the subdivision of the training set. The central and right plots show the prediction result on test data of a model trained on halved (i.e. limited region of operation) and full training set respectively.

4.4.3 Recurrent NNAP Model Multistep Prediction

The training process of a feedforward NNAP model $\mathcal{M}$ only penalizes the prediction error of the next timestep. A recurrent NNAP model $\mathcal{R}$, on the other hand, penalizes the average accuracy of a trajectory sample. Because the model learns from the general dynamics this way, multistep predictions can be potentially improved. The influence of the length of the trajectory sequence N of the recurrent model $\mathcal{R}$ of the slider-crank system is depicted in Fig. 4.14a. The aforementioned converged model $\mathcal{M}$ is used to initialize $\mathcal{R}$. Each model $\mathcal{R}$ is trained for 300 epochs in order to check whether these recurrent structures can improve the prediction capabilities. The average accuracy of the feedforward model $\mathcal{M}$ after 300 additional training epochs serves as a reference. The LOOCV reveals that the recurrent model $\mathcal{R}$, in general, does not perform better for unseen trajectories for small values of N. However, the accuracy on test data clearly outperforms the feedforward model $\mathcal{M}$ for training on larger time sequences N. The average RMSE of the test data was reduced by 33.4% for a model with $N = 300$ compared to the feedforward model $\mathcal{M}$. This improved prediction accuracy, however, comes with increased computational costs as recurrent neural networks virtually unfold into more complex multilayer networks in order to apply BPTT (as illustrated in Fig. 4.8). The required computational time (training time per epoch) almost follows a power-law relation w.r.t. the sequences length N used during the training, as can be observed in Fig. 4.14b. This experiment was performed on a processing unit including a 6-core Intel i5-8400 CPU with 16 GB RAM. Fig. 4.15a illustrates an example of a multistep prediction of a test data trajectory of $L = 800$ time instances by a NNAP model. The recurrent neural network $\mathcal{R}$ indicates better prediction performances compared to $\mathcal{M}$. The higher the length N of the time sequences used during training, the more the NNAP models are able to capture the global dynamics of the slider-crank mechanism. A FFNN architecture only incorporates the prediction accuracy of one timestep within the cost function (4.9). Naturally, drift occurs because the prediction error at a specific time instance is propagated through the subsequent timestep. A RNN structure accommodates for these drift phenomena by incorporating the prediction accuracy for an entire trajectory segment within the cost function. Therefore, the RNN model learns to capture the dynamics for a longer time horizon, which leads to more robust prediction capabilities that better track the measured trajectories, as shown in Fig. 4.15b.

4.4.4 Identifiability of the Physical Parameters

Besides having an accurate NNAP model for multistep predictions, we aim to discover new physical insights. Because the physical parameters $\mathbf{p}$ are updated simultaneously with the identification of η, research should be de-

voted to assess the identifiability of these parameters. The physical layers, including the model described in Appendix 4.6, contain 7 parameters $\mathbf{p} = \{m_1, m_2, m_3, l_1, l_2, J_1, B_m\}$. In practice, the ability to uniquely identify a parameter relates both to its sensitivity to the model output and its interference with other parameters during the optimization process. For the presented NNAP model in particular, one should also analyze whether the influence of the neural network η is sufficiently uncorrelated with parameters $\mathbf{p}$ so that the neural network does not completely take over in the optimization process. Therefore, in Appendix 4.7, a sensitivity analysis is applied to select the identifiable parameter sets, indicating that l_1 and m_3 need to be separated. Figure 4.16 illustrates the convergence history of parameter sets[1] $\mathbf{p} = \{m_3, J_1, B_m\}$ and $\mathbf{p} = \{l_1, J_1, B_m\}$ during the gradient-based optimization process of both $\mathbf{p}$ and η. The initial guess is randomly chosen $\pm$ 50% around the nominal value for each optimization run during the training procedure (LOOCV). The remaining physical parameters are held fixed during the optimization process. Nevertheless, we let them vary $\pm$ 10% around their nominal value (see Table 4.1) in each experiment to indicate their insignificance when identifying the NNAP model. The orange line indicates the median, and the optimization boundaries are defined by the black dotted lines. The actual value of the ge-

[1] Because these parameters are implemented as trainable weights within the custom physics layers, the optimization process requires parameter scaling to cope with the differences in order of magnitude between the variables in $\mathbf{p}$.

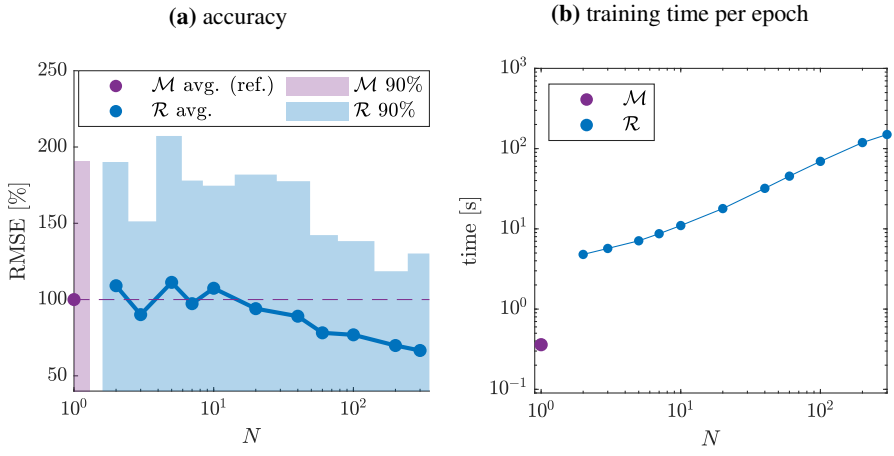


Figure 4.14: Influence of step size N on the accuracy and required training time per epoch. The accuracy of the multistep predictions is tested by LOOCV. The left plot shows the average accuracy and the interval that includes 90% of the most accurate test signals. The right plot illustrates the time required to train one epoch for different number of timesteps N of the recurrent NNAP model $\mathcal{R}$.

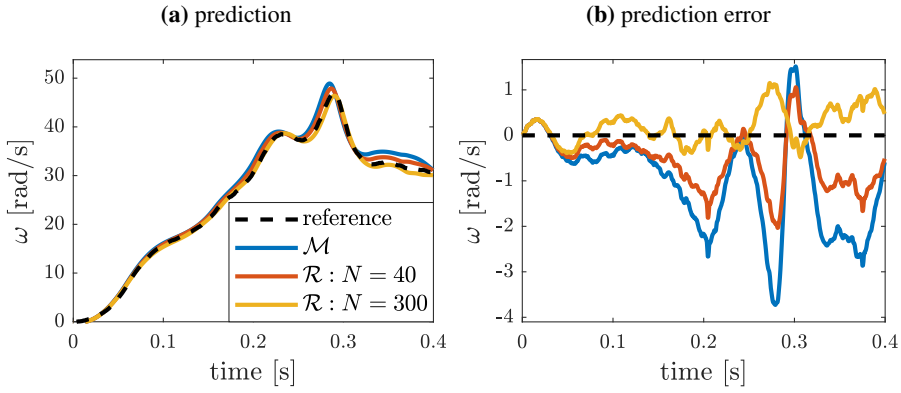


Figure 4.15: Multistep prediction of test signal T_9.

ometrical parameter l_1 can easily be measured and is indicated by the purple line. The presented results indicate convergence of the proposed parameter sets. However, note that convergence cannot be obtained for all parameter combinations, as is further explained in Appendix 4.7.

4.4.5 Retrieving Physical Insights from the Neural Network

The identification of the physical parameters in $\mathbf{p} = \{m_3, J_1, B_m\}$ are indicative for the convergence of η because both parameters $\mathbf{p}$ and $\boldsymbol{\alpha}$ are simultaneously optimized. Therefore, we analyze the information captured by η after convergence of the NNAP model $\mathcal{M}$ to determine if the unknown load interactions $\mathcal{P}$ can be discoverd. At initialization, the neural network has no physical meaning (no resemblance with the actual $\mathcal{P}$) because the neural network is defined by random learnable parameter values $\boldsymbol{\alpha}_0$. The extracted neural network having as inputs d and v and as output $z = F$ is depicted in Fig. 4.17a. In that phase of training, completely wrong estimations of z are used as the input to (4.2), resulting in high initial loss. After convergence to a predictive NNAP model $\mathcal{M}$ (or $\mathcal{R}$), the ReLU network η with optimized parameter values $\boldsymbol{\alpha}^*$ can be extracted, see Fig. 4.17b.

The optimized ReLU network η is evaluated on the training dataset, resulting in Fig. 4.18a. The model η is only precise locally around the area because data-driven models are inherently difficult in extrapolation capabilities. In [37], we show for numerical simulation data that the ReLU network η indeed converges towards the real force relation F at these explored areas. The corresponding side view, depicted in Fig. 4.18b, illustrates the similarity between the empirically validated compression spring force F_s (see also Fig. 4.4) and the identified ReLU network η. From Fig. 4.18b, we can deduce a spring law together with friction phenomena.

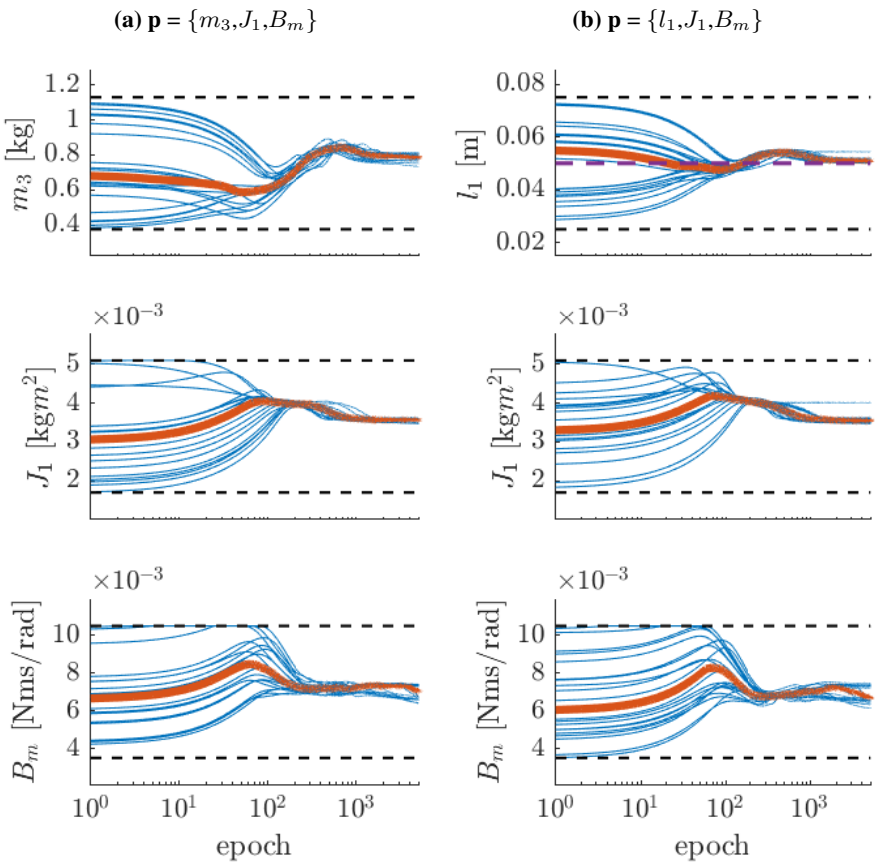


Figure 4.16: Convergence history of physical parameters $\mathbf{p}$ during simultaneous optimization of $\mathbf{p}$ and neural network parameters $\boldsymbol{\alpha}$. The left and right plots indicate the convergence history of an optimization process of parameters $\mathbf{p} = \{m_3, J_1, B_m\}$ and $\mathbf{p} = \{l_1, J_1, B_m\}$, respectively.

The discrepancy between the discovered force relation η and the spring force F_s raises the interest to elaborate further on the interpretability of η. The overall horizontal force acting on the slider is, therefore, considered as a summation $F = F_c + F_{nc}$ of a conservative (F_c) and non-conservative (F_{nc}) force. A conservative force is characterized by the property that the work done by moving the object between two points is independent of the taken path. In practice, we consider a conservative force $F_c(d)$, being only dependent on the position d of the slider. Furthermore, the dissipative force $F_{nc}(d, v)$ relies on both the position d and linear speed v. Therefore, we replaced the ReLU network $z = \eta(d, v)$ with a new relation $z = \eta_c(d) + \eta_{nc}(d, v)$ deploying a summation of two separate feedforward ReLU networks $\eta_c(d)$ and $\eta_{nc}(d, v)$ that are trained in parallel within the NNAP model $\mathcal{M}$. The solution of $\eta_c(d)$

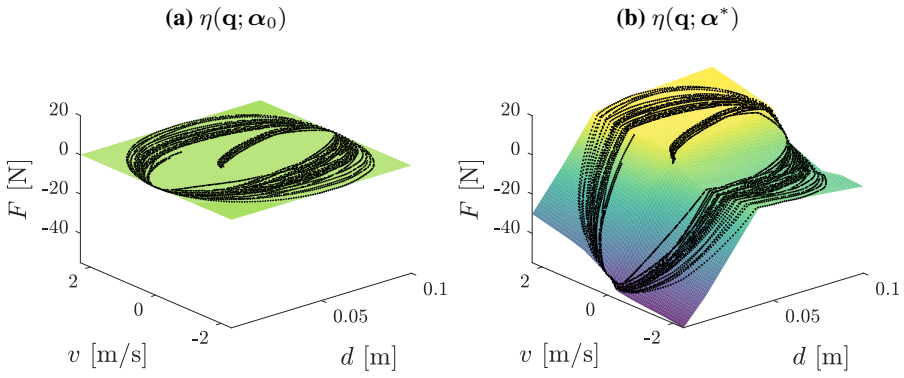


Figure 4.17: Extracted ReLU network η for different model parameters $\boldsymbol{\alpha}$. The black dots indicate the data that were passed through the network during training.

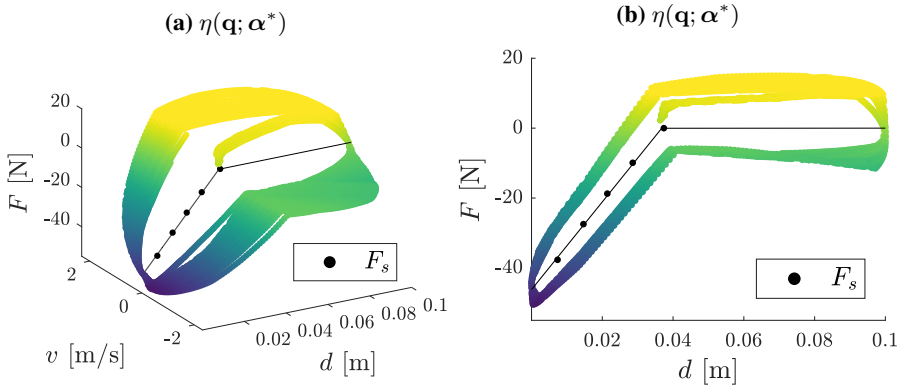


Figure 4.18: Extracted ReLU network η evaluated on the training dataset.

and $\eta_{nc}(d, v)$ is, however, not unique because $\eta_{nc}(d, v)$ includes the inputs of $\eta_c(d)$. In practice, an additional L_2 regularization term $(c \cdot \eta_{nc}^2)$, with regularization parameter c, was added to the loss function. Fig. 4.19 illustrates the identified network components for an experimentally tuned regularization parameter $c = 10^{-6}$. The obtained results are the average predictions of η_c and η_{nc} evaluated by the training data for 10 converged NNAP models $\mathcal{M}$. The ReLU network η_c is able to accurately predict the behavior of the conservative spring force F_s, as can be observed in Fig. 4.19a. The regularization was able to successfully address the spring force to the ReLU network η_c. The remaining unknown dynamics attributed to the η_{nc} are shown in Fig. 4.19c. From the corresponding side view in Fig. 4.19e, we are able to discover the friction

pattern that is dependent on the direction of the slider.

To validate the methodology that was able to retrieve physical insights, the aforementioned experiment was repeated on the experimental setup having no compression spring. Hence, the state-dependent conservative force is removed. From Fig. 4.19b, it can be seen that the discovered conservative force model η_c trends towards zero. The non-conservative force can be extracted from η_{nc} and is depicted in Figs. 4.19d and 4.19f. The non-conservative forces exhibit similar behavior as those shown in 4.19c and 4.19e. These results suggest the ability of NNAP models to provide insights into the on-going conservative and non-conservative forces of a mechatronic system interacting with its environment.

4.5 Conclusion

This chapter presents neural network augmented physics (NNAP) models that can learn the behavior of systems for which the dynamics are only partially known. Data-driven modeling techniques, here neural networks, are inserted in the model to compensate for the unmodeled interactions, without requiring direct measurements of these unknown phenomena. The method is validated on a slider-crank mechanism that exhibited unknown load interactions, induced by a spring and additional friction phenomena. The NNAP model is identified by simultaneously optimizing the neural network and the physical parameters, solely by using state and control input measurements. Therefore, automatic differentiation is used to calculate the gradient information through both the custom physics-inspired and neural layers. The obtained results of the NNAP model are twofold. First, the modeling formalism enabled robust and accurate multistep predictions of a system that was a priori only partially known. Furthermore, both a feedforward and recurrent implementation of the modeling formalism are benchmarked, for which the latter was able to reduce the prediction error by up to 33.4% on test data. Secondly, the methodology provided a means to discover new insights of the unknown system properties. Results have shown that the NNAP methodology applied to a slider-crank mechanism can uniquely identify uncorrelated sets of physical parameters, while optimizing the neural network. In turn, the neural network, which was extracted from the NNAP model after convergence, accurately captured the state dependent load interactions. The presented methodology, however, heavily relies on the availability of accurate physical models describing the known dynamics. Furthermore, a good understanding of the influence of the unknown phenomena within the overall model is essential to assure physically interpretable results. It should be noted that for more complex cases, the flexibility of the neural network can take over the identification process, resulting in non-unique so-

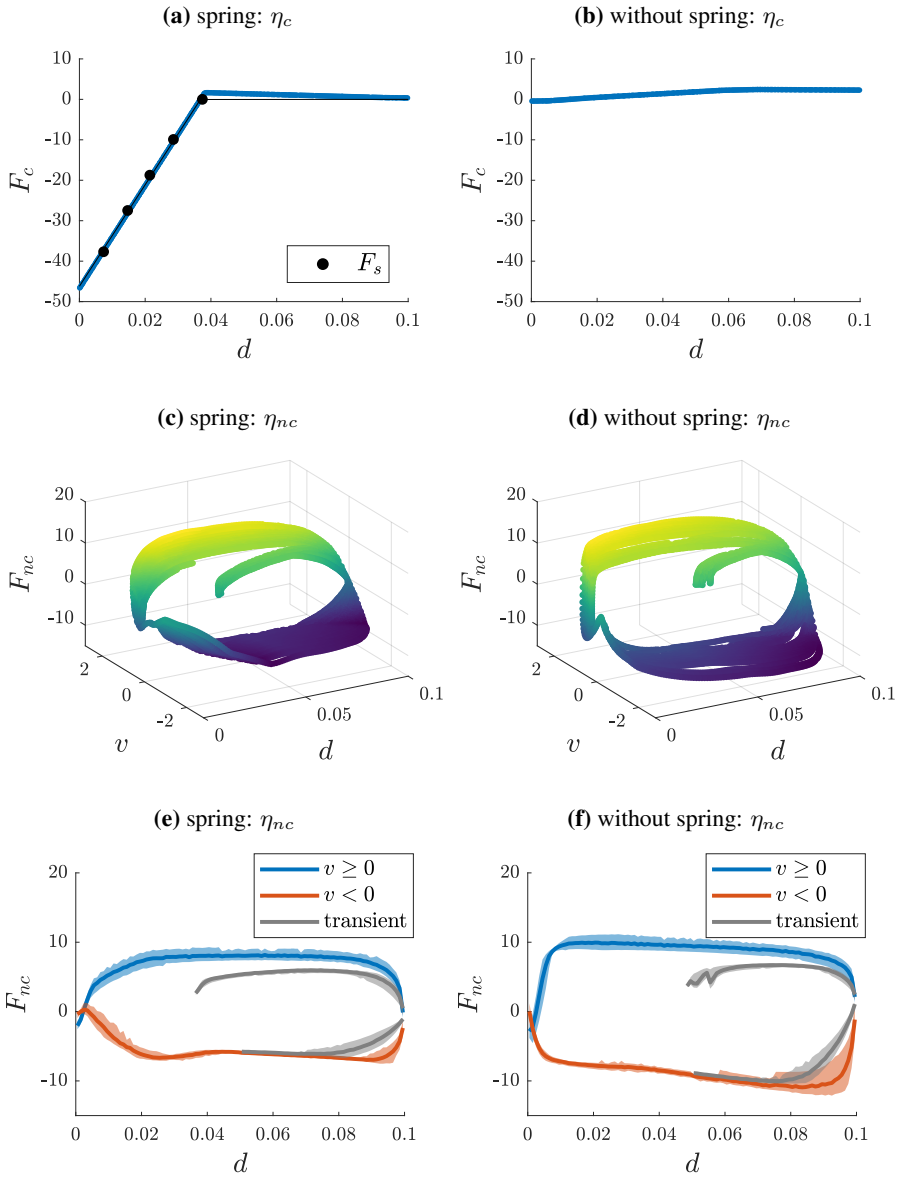


Figure 4.19: Extracted ReLU networks η_c and η_{nc} evaluated on the training dataset, capturing respectively conservative and nonconservative forces.

lutions. Further research is, therefore, required for discovering unknown phenomena using more elaborate regularization strategies.

4.6 Appendix: Physics-Based Model

The physics-based dynamical model of the slider-crank mechanism is derived by considering the simplified multibody system in Fig. 4.20. The variable J_1 includes the rotational inertia of both motor and crank with respect to the rotation axis. The rotational inertia J_2 of the connection rod is approximated by $J_2 = \frac{1}{3} l_2 m_2^2$ in its center of mass (CM). Subsequently, this implies that we take the $r_2 = \frac{l_2}{2}$ in Fig. 4.2. By analogy, we approximate the CM of the crank by $r_1 = \frac{l_1}{2}$. Furthermore, we define $r' = l - r$ and let the subscripts r_x and r_y indicate the projection of r on, respectively, the x-axis and y-axis.

$$
\begin{aligned}
m_1 \ddot{x}_1 &= F_{ax} + F_{bx} \\
m_1 \ddot{y}_1 &= -m_1 g + F_{ay} + F_{by} \\
J_1 \dot{\omega} &= T - B_m \omega - r_{1x} m_1 g + l_{1x} F_{by} - l_{1y} F_{bx} \\
m_2 \ddot{x}_2 &= -F_{bx} + F_{cx} \\
m_2 \ddot{y}_2 &= -F_{by} + F_{cy} - m_2 g \\
J_2 \ddot{\phi} &= -r'_{2x} F_{by} - r'_{2y} F_{bx} - r_{2x} F_{cy} - r_{2y} F_{cx} \\
0 &= -F_{cy} - m_3 g + F_N \\
m_3 \dot{v} &= -F_{cx} - F
\end{aligned}
\tag{4.11}
$$

The aforementioned equations are combined with the geometrical relations, illustrated in Fig. 4.2, towards a physics-inspired model $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u, F)$ with state $\mathbf{x} = \begin{bmatrix} \theta & \omega \end{bmatrix}^T$ and control input $u = T$. The highly nonlinear expression of $\mathbf{f}$ is derived via symbolic solvers and is due to its complexity not mentioned in this chapter.

Table 4.1: Identified parameters of slider-crank setup.

m_1	mass of crank	0.23	[kg]
m_2	mass of connection rod	0.35	[kg]
m_3	mass of slider	0.77	[kg]
l_1	length of crank	0.05	[m]
l_2	length of connection rod	0.29	[m]
J_1	inertia motor + crank	0.0034	[kg m^2]
B_m	friction coefficient of motor	0.007	[$\frac{\text{N m}}{\text{rad/s}}$]

4.7 Appendix: Sensitivity Analysis

The identified physical parameters $\mathbf{p}$ of the slider-crank mechanism are depicted in Table 4.1. The parameters that could not be determined by optimizing the NNAP model are measured experimentally or determined via CAD

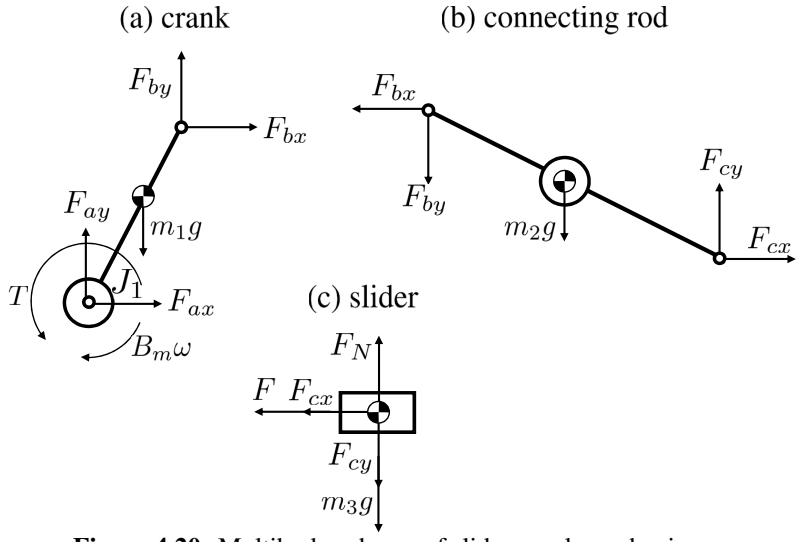


Figure 4.20: Multibody scheme of slider-crank mechanism.

drawings. Because some parameter combinations may lead to the same model output, non-unique solutions may, however, appear. The highly nonlinear nature of the proposed slider-crank model makes a structural identifiability analysis non trivial. However, based on [38], we implemented approximation techniques indicative for local identifiability around the optimal solution. We elaborate on the derivative function f evaluated for a sample k as $\dot{\omega}_k = f(\theta_k, \omega_k, T_k, \hat{F}_k; \mathbf{p})$. In practice, all measurement trajectories are aggregated into one dataset of K samples. The converged neural network η is deployed on each measurement sample $k \in \{1, \ldots, K\}$ to generate the corresponding estimations $\hat{F}_k$ of the unknown force interaction. Next, we derive the sensitivity matrix S in which we quantify the impact on the model output for modifications on F and physical parameters $\mathbf{p} \in \mathbb{R}^r$.

$$\mathrm{S} = \left[\mathbf{s}_1 \cdots \mathbf{s}_{r+1}\right] = \begin{bmatrix} \frac{\partial f}{\partial F}\Big|_{k=1} & \frac{\partial f}{\partial p_1}\Big|_{k=1} & \cdots & \frac{\partial f}{\partial p_r}\Big|_{k=1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial F}\Big|_{k=K} & \frac{\partial f}{\partial p_1}\Big|_{k=K} & \cdots & \frac{\partial f}{\partial p_r}\Big|_{k=K} \end{bmatrix}$$

Again, we consider normalized parameters in order to develop a fair scaling to cope with the different sizes of magnitude. The L_2 norm of each sensitivity vector $\mathbf{s}_j$ measures the influence of the corresponding feature j to the model output. Figure 4.21 illustrates the high sensitivity of the parameters l_1, J_1 and m_3, justifying the choice to include them in the optimization set $\mathbf{p}$. In addition, the parameter B_m is added because this variable is unknown and cannot be measured. The high sensitivity of F, which is representative for the output of the neural network η, is related to the convergence of η. Although a set

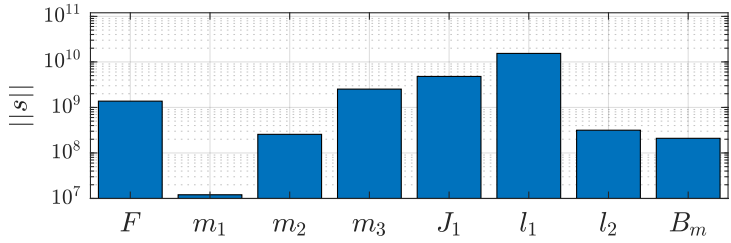


Figure 4.21: Local sensitivity analysis of unknown force interaction F and physical parameters $\mathbf{p}$.

of variables can be highly sensitive, if parameter combinations have the same influence on the model output, they cannot be uniquely determined. Therefore, we construct the normalized sensitivity matrix S_N that contains the normalized column vectors $\mathbf{s}_j^N = \mathbf{s}_j/||\mathbf{s}_j||$ for $j \in \{1, .., r+1\}$. The corresponding correlation matrix $\mathrm{Q} = \mathrm{S}_N^T \mathrm{S}_N$ exhibits diagonal elements equal to one and results in the graph shown in Fig. 4.22. This result demonstrates that the physical parameters are barely correlated with the output of the neural network represented by F. However, we can observe, for instance, a high correlation of parameters l_1 with m_3 (and m_2) on the model output. When considering parameter set $\mathbf{p} = \{l_1, m_3, J_1, B_m\}$, we can observe from Fig. 4.23 that they indeed do not converge well in the optimization process.

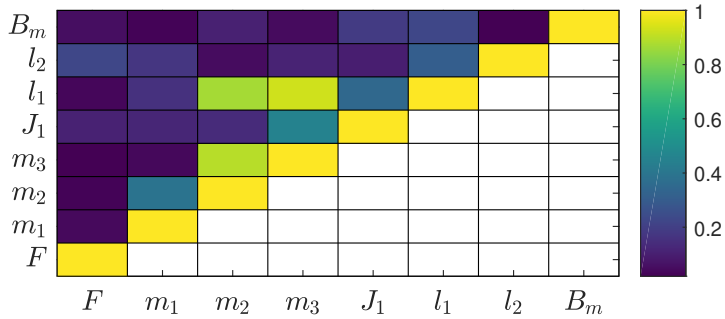


Figure 4.22: Correlation matrix Q of the sensitivity vectors $\mathbf{s}_j^N$.

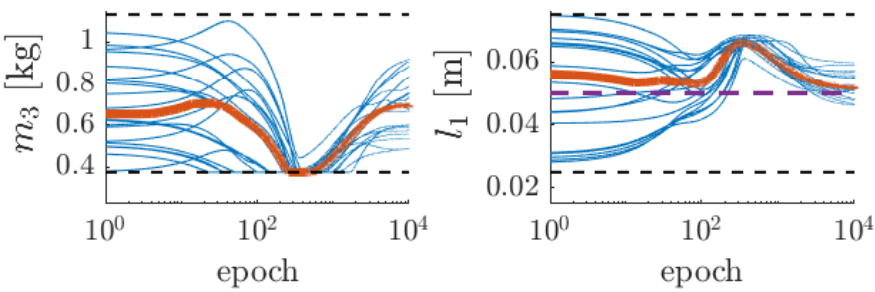


Figure 4.23: Convergence history of l_1 and m_3 when including both in the parameter set $\mathbf{p} = \{l_1, m_3, J_1, B_m\}$, simultaneously optimized with the neural network parameters $\boldsymbol{\alpha}$.

References

[1] S. Kim. Moment of inertia and friction torque coefficient identification in a servo drive system. IEEE Transactions on Industrial Electronics, 66(1):60–70, 2019.

[2] D. Papageorgiou, M. Blanke, H. H. Niemann, and J. H. Richter. Robust backlash estimation for industrial drive-train systems–theory and validation. IEEE Transactions on Control Systems Technoly, 2018.

[3] X. Wang, W. Wang, L. Li, J. Shi, and B. Xie. Adaptive control of dc motor servo system with application to vehicle active steering. IEEE/ASME Transactions on Mechatronics, 2019.

[4] A. Kamadan, G. Kiziltas, and V. Patoglu. Co-design strategies for optimal variable stiffness actuation. IEEE/ASME Transactions on Mechatronics, 22(6):2768–2779, 2017.

[5] S. J. Kim, K.-S. Kim, and D. Kum. Feasibility assessment and design optimization of a clutchless multimode parallel hybrid electric powertrain. IEEE/ASME Transactions on Mechatronics, 21(2):774–786, 2015.

[6] H. Ziyuan, H. Bangcheng, and L. Yun. An explicit nonlinear model predictive abs controller for electro-hydraulic braking systems. IEEE Transactions on Industrial Electronics, 2019.

[7] T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. Automatica, 47(1):39–49, 2011.

[8] J. Kocijan. Modelling and control of dynamic systems using Gaussian process models. Springer, 2016.

[9] T. B. Wills, A.and Schön, L. Ljung, and B. Ninness. Identification of hammerstein–wiener models. Automatica, 49(1):70–81, 2013.

[10] S. A. Billings. Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. John Wiley & Sons, 2013.

[11] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521(7553):436, 2015.

[12] O. Lakhal, A. Melingui, and R. Merzouki. Hybrid approach for modeling and solving of kinematics of a compact bionic handling assistant manipulator. IEEE/ASME Transactions on Mechatronics, 21(3):1326–1335, 2015.

[13] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans. Nonlinear systems identification using deep dynamic neural networks. arXiv preprint arXiv:1610.01439, 2016.

[14] N. Mohajerin and S. L. Waslander. Modular deep recurrent neural network: Application to quadrotors. In IEEE International Conference on Systems, Man, and Cybernetics, pages 1374–1379. IEEE, 2014.

[15] N. Mohajerin and S. L. Waslander. Multistep prediction of dynamic systems with recurrent neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2019.

[16] Z. Liu, H. Fang, and J. Xu. Identification of piecewise linear dynamical systems using physically-interpretable neural-fuzzy networks: Methods and applications to origami structures. Neural Networks, 116:74–87, 2019.

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. arXiv preprint arXiv:1801.01236, 2018.

[18] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. Science, 324(5923):81–85, 2009.

[19] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. of the Nat. Academy of Sciences, page 201517384, 2016.

[20] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. International Conference on Learning Representations, 2019., 2019.

[21] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In Advances in Neural Information Processing Systems, pages 15353–15363, 2019.

[22] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In Proceedings of the 2019 SIAM International Conference on Data Mining, pages 558–566. SIAM, 2019.

[23] A. Ajay, M. Bauza, J. Wu, N. Fazeli, J. B. Tenenbaum, A. Rodriguez, and L. P. Kaelbling. Combining physical simulators and object-based networks for control. arXiv preprint arXiv:1904.06580, 2019.

[24] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In IEEE International Conference on Robotics and Automation, pages 3223–3230. IEEE, 2015.

[25] I. Boldea, L. N. Tutelea, W. Xu, and M. Pucci. Linear electric machines, drives, and maglevs: an overview. IEEE Transactions on Industrial Electronics, 65(9):7504–7515, 2018.

[26] C. S. Sharma and K. Purohit. Theory of mechanisms and machines. PHI Learning, 2006.

[27] Z. Gao, R. S. Colby, L. Turner, and B. Leprettre. Filter design for estimating parameters of induction motors with time-varying loads. IEEE Transactions on Industrial Electronics, 58(5):1518–1529, 2011.

[28] M. Li, R. Foss, K. Stelson, J. Van de Ven, and E. J. Barth. Design, dynamic modelling and experiment validation of a novel alternating flow variable displacement hydraulic pump. IEEE/ASME Transactions on Mechatronics, 2019.

[29] G. Eren, R.and Ozkan and M. Karahan. Comparison of heald frame motion generated by rotary dobby and crank & cam shedding motions. Fibres and textiles in eastern Europe, 13(4):78, 2005.

[30] E. Zheng and X. Zhou. Modeling and simulation of flexible slider-crank mechanism with clearance for a closed high speed press system. Mechanism and Machine theory, 74:10–30, 2014.

[31] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989.

[32] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. arXiv preprint arXiv:1710.05941, 2017.

[33] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In International Conference on Machine Learning, pages 1310–1318, 2013.

[34] F. Chollet et al. Keras. `https://keras.io`, 2015.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symp. on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016.

[36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[37] W. De Groote, E. Kikken, S. Goyal, S. Van Hoecke, E. Hostens, and G. Crevecoeur. Hybrid derivative functions for identification of unknown loads and physical parameters with application on slider-crank mechanism. In 2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). IEEE, 2019.

[38] I. Ioslovich, M. Moran, and P. Gutman. Identification of a nonlinear dynamic biological model using the dominant parameter selection method. Journal of the Franklin Institute, 347(6):1001–1014, 2010.

Chapter 5

Using Physics-Inspired Features in Recurrent Neural Networks to Predict Nonlinear System Behavior

In Chapter 4 we studied the use of neural networks to complement for the unmodeled interactions in a partially-known physics model. In practice, there is often no substantial physics-model available, but only some insights about particular subsystems. In this chapter, these insights are provided as physics-inspired input features to a recurrent neural network, enabling for enhanced prediction performances. The presented approach is validated on experimental data of a cam-follower mechanism to predict the possible occurrence of detachment phenomena. My contributions can be summarized as follows:

- An extensive study is performed to assess the influence of providing physics-inspired features to an LSTM model. The effect of the amount of physics included in the features and of the scarcity of the dataset on the prediction performances is elaborated upon.

- Next to analyzing the predictive capabilities, I studied the possibility to quantify the contribution of the physics-inspired features to the model output. Therefore, I implemented an additive feature attribution method (i.e., SHAP), enabling for an explanation model that objectively can determine the influence of the input features.

Prediction of Follower Jumps in Cam-Follower Mechanisms: the Benefit of Using Physics-Inspired Features in Recurrent Neural Networks

W. De Groote, S. Van Hoecke, and G. Crevecoeur

Published in "Mechanical Systems and Signal Processing", 2022

Abstract *The high functional performance exhibited by modern applications is very often established by an aggregation of various intricate mechanical mechanisms, providing the required motion dynamics to the overall system. Above all, the mechanism's behavior should be reliable for a wide range of operating conditions to assure at all times appropriate functioning of the entire application. In particular, cam-follower mechanisms, which translate a rotational movement into a linear displacement, are plagued by the high dynamics induced by the reciprocating motions. For specific operating conditions, the follower tends to detach from the cam perimeter, resulting in harmful bouncing behavior. This chapter presents the use of recurrent neural networks to estimate the follower jump trajectory, based on cam rotation measurements, for a wide range of operating conditions and system modifications. Although these data-driven models are typically known to learn intricate patterns directly from raw data, enhanced prediction performances are observed when providing physics-inspired features to the model. The effect is especially more pronounced when learning from a small amount of data or from datasets for which the data are not uniformly distributed along the parameter space. In addition, this chapter presents the use of an additive feature attribution method to quantify the contribution of features in multivariate timeseries on the prediction output of recurrent neural network models. Hence, we show that, by means of the Shapley additive explanation (SHAP) values, the model prioritizes the incorporation of physics-inspired features, explaining the improved generalization capabilities of the prediction model. In general, these presented results indicate the potential to incorporate physics-inspired expert knowledge into various other prediction models, enabling advanced methodologies to monitor inconvenient phenomena in mechanical systems.*

5.1 Introduction

Modern machines are typically complex aggregations of various interacting mechanical subsystems, which transform an amount of energy into an intended action. In practice, mechanical systems often require to endure transient load interactions for a wide range of operating conditions, making them

prone to failure. These mechanical mechanisms, therefore, require intricate design choices to assure optimal functional performance of the overall system. Hence, having reliable systems that can be accurately monitored is becoming increasingly important. This research places emphasis on a cam-follower mechanism which translates a rotational displacement into a reciprocating motion [1]. These mechanisms are typically implemented in combustion engines to regulate the intake and exhaust valves of the cylinders [2,3]. Their ability to accurately determine a motion path has been incorporated in the design of actuators [4], robotics [5] and presses [6]. Furthermore, they have applications in high-precision pumps used in avionics [7] and biomedical applications [8,9].

In general, the correct functioning of a cam-follower mechanism imposes a perfect and continuous tracking of the cam perimeter by the follower. However, for increased rotational speed, the follower can detach from the cam, resulting in harmful bouncing behavior, as experimentally observed and described in [10,11]. These follower jumps lead to high periodical impacts that can damage the system [12,13]. For some dedicated machines, such as cutting tools, these impacts can be desired [14]. However, most systems aim to avoid this harmful behavior and require continuous contact between cam and follower. Therefore, for many years, research has been devoted to study the effect of the operational and design parameters on the occurrence of jump phenomena [15–20]. Consequently, design procedures are developed that provide the required follower displacements while avoiding excessive dynamics [21]. Next to optimizing the system design, smooth motion behavior has also been obtained by controlling the cam speed [22,23].

Despite these ingenious advancements, machines can always encounter unexpected excessive loads or being unintentionally forced towards operating conditions for which they were not designed. Therefore, the development of advanced data-driven techniques that can monitor the machine condition is of increasing importance [24]. These so-called machine learning models often rely on engineered input features, typically obtained from classical statistical metrics or by using dedicated signal processing techniques, such as a transformation from the time to a spectral domain [25]. These methodologies have proven to be useful for monitoring purposes in the field of various applications, such as wind turbines [26] and bearings [25,27].

Deep learning methodologies are a particular subclass of machine learning that has shown the ability to learn intricate patterns directly from raw data, hence limiting the need for feature engineering [28–30]. More in particular, long short term memory (LSTM) networks have shown, due to their ability to distill important events in long timeseries, their utility in the field of timeseries classification and machine prognostics [31]. Furthermore, such networks can be implemented as a seq2seq model, which serves as a nonlinear mapping between an input and output sequence. These architectures have shown ac-

curate results in the prediction of various inconvenient phenomena, such as high-impact loads [32] and brake squeal [33].

Although the LSTM formalism is designed to learn patterns directly from the data, recent advances explore the possibility of combining LSTM neural networks directly with expert knowledge. LSTM models have shown to serve as mappings to compensate for modeling discrepancies of physical models [34, 35]. Moreover, improved generalization is obtained by imposing physical consistency (i.e., conservation of energy) in the loss function [36]. By adding physics-inspired constraints, the network is alleviated from overfitting and consequently learns more generalized models from scarce datasets [37].

The incorporation of physics has revealed the potential for improved prediction capabilities. However, it remains difficult to quantify how much the physics-inspired prior knowledge actually contributes to the model. This question can, however, be approached from a more general perspective by analyzing the influence of each input feature to the model output. Traditional machine learning models can often be explained by intuitive techniques such as permutation methods, which are based on calculating the effect on the prediction error after permuting one or multiple features [38]. However, as shown in [39], these methods often fail for deep learning models due to saturation phenomena caused by the nonlinear activation functions. By contrast, backpropagation methodologies provide a means to accommodate for this highly nonlinear nature of deep learning models by propagating the importance signal from an output neuron backwards through the layers to the input. Various recent advances, such as Interior Gradients [40], Layer-Wise Relevance Propagation [41] and DeepLIFT [39] have shown promising results by propagating information through the layers of deep neural networks. Lundberg & Lee [42] discovered that many known techniques, such as LIME [43], Layer-Wise Relevance Propagation [41] and DeepLIFT [39], although initially not intended, boil down to one group of explanation models (i.e., additive feature attribution methods). However, although describing the same prediction model, they all assign a different contribution to the features. Consequently, Lundberg & Lee introduced the idea of SHapley Additive exPlanation (SHAP) values [42], which combine the aforementioned group of techniques with traditional Shapley values [44], resulting in a unified approach to interpret model predictions. Within the field of timeseries modeling, SHAP values have shown to be useful for interpreting classifications of univariate timeseries by quantifying the relevance of the model input at each time instance [45, 46]. Moreover, the framework has also shown promising results to determine the importance between features in multivariate timeseries [47].

This chapter elaborates on the prediction of follower jumps occurring in cam-follower mechanisms using LSTM networks with physics-inspired input features. An extensive dataset is generated on a laboratory setup to assess the

prediction performances of the learned recurrent data-driven models. Section 5.2 elaborates on this setup and corresponding dataset, generated for a wide range of operating conditions. This unique dataset incorporates timeseries captured for various system modifications (e.g., cam shape and follower mass) to benchmark the generalization capabilities of the presented models. The main objective of this chapter is, furthermore, twofold. First, an LSTM network $\mathcal{R}$ is combined with physics-inspired input features to obtain improved prediction performances of the follower jumps in cam-follower mechanisms, as illustrated in Fig. 5.1. Section 5.3 details the design of well-chosen physics-inspired features, such as the interface force and the cam dynamics, to enable enhanced generalization capabilities of the prediction model $\mathcal{R}$. In Section 5.4, we elaborate on the prediction performances by benchmarking the generalization capabilities of the physics-informed architectures against their data-driven counterparts. Second, we determine the contributions of all features provided to the LSTM network in Section 5.5. We hereby wish to quantify the actual importance of the physics-inspired features by means of an explanation model g, as illustrated in Fig. 5.1. This quantification requires a dedicated implementation of the SHAP (DeepSHAP) framework to interpret the importance of features in multivariate timeseries of the cam-follower mechanism. This way, we establish an objective manner to show the importance of incorporating expert knowledge into recurrent deep learning models. Finally, we present the conclusions and future work in Section 5.6.

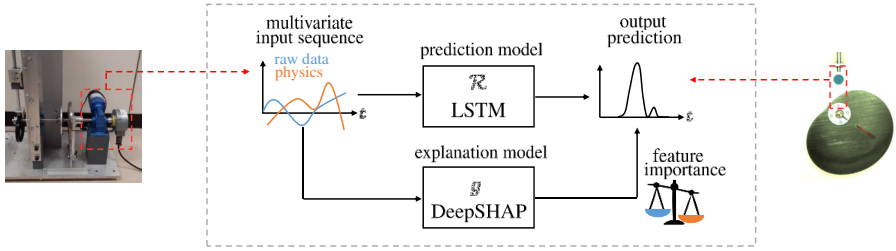


Figure 5.1: Schematic representation of the objectives. First, an LSTM model $\mathcal{R}$ is augmented with physics-inspired input features to better predict the occurrence of follower jumps in cam-follower mechanisms. Secondly, an explanation model g is developed, based on the DeepSHAP framework, to quantify the contributions of the physics-inspired features to the model predictions.

5.2 Cam-follower mechanism

A cam-follower mechanism translates a rotary motion into a linear displacement. Figure 5.2 illustrates a lumped-parameter representation of the mecha-

nism. The follower, characterized by mass m, is subject to an interface force F delivered by the rotating cam. The follower motions are influenced by friction phenomena, summarized by force F_b. Traditional cam-follower systems are typically equipped with a return spring, to enforce the contact between cam and follower. In this research we omitted the use of a compression spring to invoke the occurrence of follower jumps. The cam shape is defined by a displacement function $h(\theta)$ that defines the position of the cam perimeter for given rotation angle θ. In a nominal case, the displacement of the follower y perfectly tracks the cam profile h. However, for some operating conditions, the follower detaches from the cam, leading to harmful bouncing behavior. Therefore, we research the ability to monitor the displacement $\epsilon = y - h$, solely by using variables deduced from the rotary encoder measurements θ.

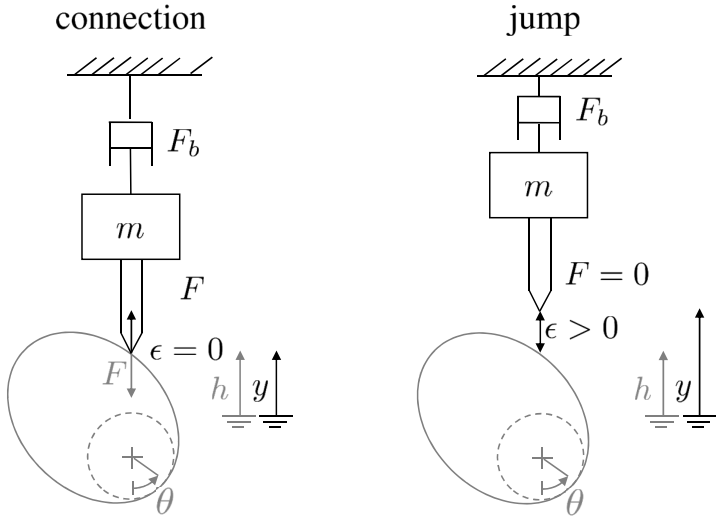


Figure 5.2: Representation of a cam-follower mechanism as a mass-damper model. The left plot is indicative of a nominal operating condition, implying a continuous contact between cam and follower (i.e., $\epsilon = 0$). The right plot illustrates an undesired situation at which follower jump occurs (i.e., $\epsilon > 0$).

5.2.1 Cam-follower setup

The presented prediction models are trained and validated based on measurement data of the cam-follower setup depicted in Fig. 5.3. The cam is driven by a 60 W DC motor, which is controlled by a dSPACE 1104 control unit that governs the voltage V applied to the motor. The motor is equipped with a highly accurate incremental rotary encoder (10 000 lines) to measure θ. The

high accuracy allows numerical differentiation to deduce the rotational speed ω and acceleration $\dot{\omega}$. The position of the follower y is measured by a high-precision linear encoder with a resolution of 2 μm. These sensors capture the data at 2000 Hz. There is no spring attached to the follower, to increase the occurrence of jump phenomena. This modular setup enables various system modifications, such as a varying load m, which can be modified by mounting additional discs to the follower mechanism.

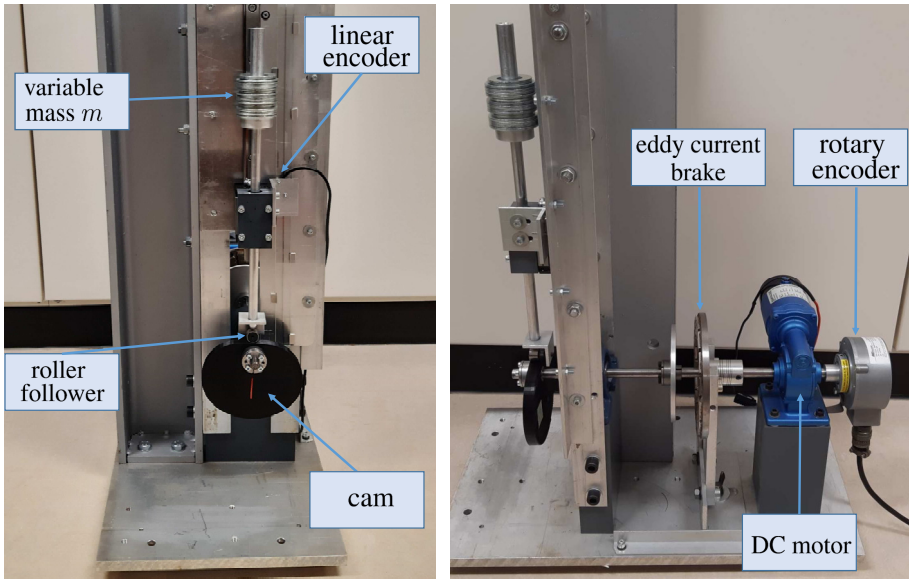


Figure 5.3: Cam-follower setup.

In addition, the setup is equipped with a set of 20 interchangeable cams, as shown in Fig. 5.4. Each cam is characterized by a cycloidal displacement function $h_c(\theta)$ without dwell, which is defined by:

$$h_c = \begin{cases} \frac{H\theta}{\beta} - \frac{H}{2\pi}\sin(2\pi\frac{\theta}{\beta}), & \text{if } \theta \in [0, \beta] \\ H\frac{(2\pi-\theta)}{(2\pi-\beta)} - \frac{H}{2\pi}\sin(2\pi\frac{(2\pi-\theta)}{(2\pi-\beta)}), & \text{if } \theta \in]\beta, 2\pi] \end{cases} \tag{5.1}$$

This type of cam profile exhibits continuous derivatives $\frac{\mathrm{d}h_c}{\mathrm{d}\theta}$ and $\frac{\mathrm{d}^2h_c}{\mathrm{d}\theta^2}$, which result in smooth dynamics of the mechanism [1]. Consequently, each cam is characterized by only two parameters: the maximum displacement H and the skewness angle β. The corresponding theoretical displacement functions h_c are shown in Fig. 5.5. Note that the setup is equipped with a roller follower with radius $r_f = 0.008$ m. Therefore, the displacement y does not perfectly tracks the theoretical cam profile h_c. Consequently, the actual cam displacement h is approximated by the pitch curve (i.e., h_p). For further details about the pitch curve and the key cam characteristics we refer to Appendix 5.7.

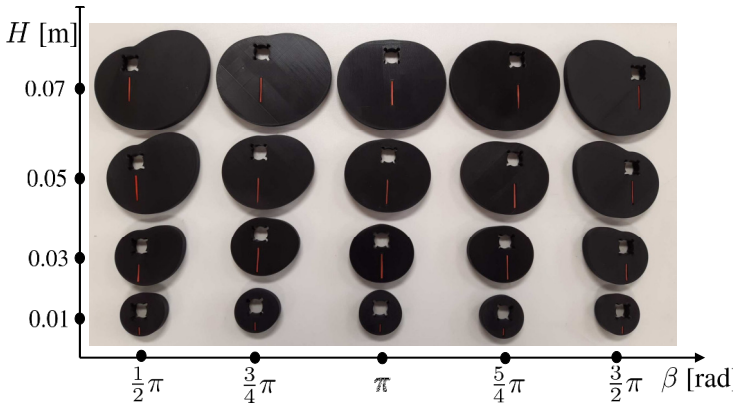


Figure 5.4: Cam shapes parameterized by β and H.

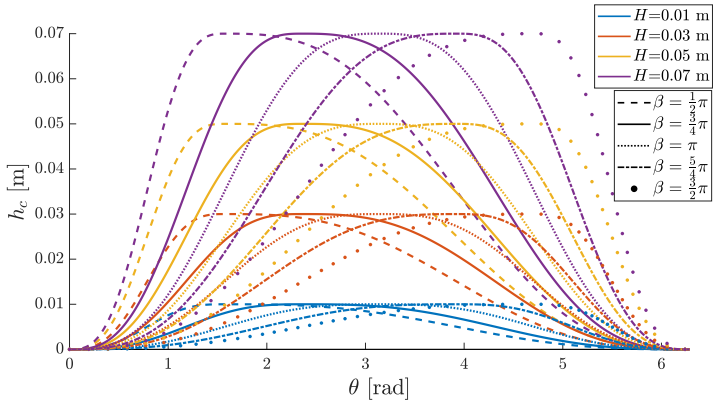


Figure 5.5: Theoretical displacement functions parameterized by β and H.

5.2.2 Measurement analysis

The setup is used to analyze the influence of various system properties $\mathbf{q}_s = \{m, H, \beta\}$, related to the design of the cam-follower mechanism, on the tendency for the follower to detach from the cam surface. Each system configuration $\mathbf{q}_s$ is tested for a wide range of voltages V, inputted to the motor, to assess the behavior of the follower. Each experiment begins from a standstill (i.e., $\omega = 0$) at an angular displacement $\theta = 0$. Figure 5.6 illustrates the influence of various system parameters $\mathbf{q}_s$ and voltages V. The high speeds invoked by high voltages V result in more pronounced bouncing behavior. Similarly, the height of the cam H and skewness angle β clearly affect the tendency to bounce. Note that higher masses m lead to higher gravitational forces, thus

resulting in better contact between cam and follower. Figure 5.7 shows the relations between the motion discrepancy ϵ and the corresponding angular displacement θ. Although exceptions exist, this figure clearly indicates that the follower jump typically initiates during the fall phase (i.e., $\frac{dh}{d\theta} < 0$) of the linear motion.

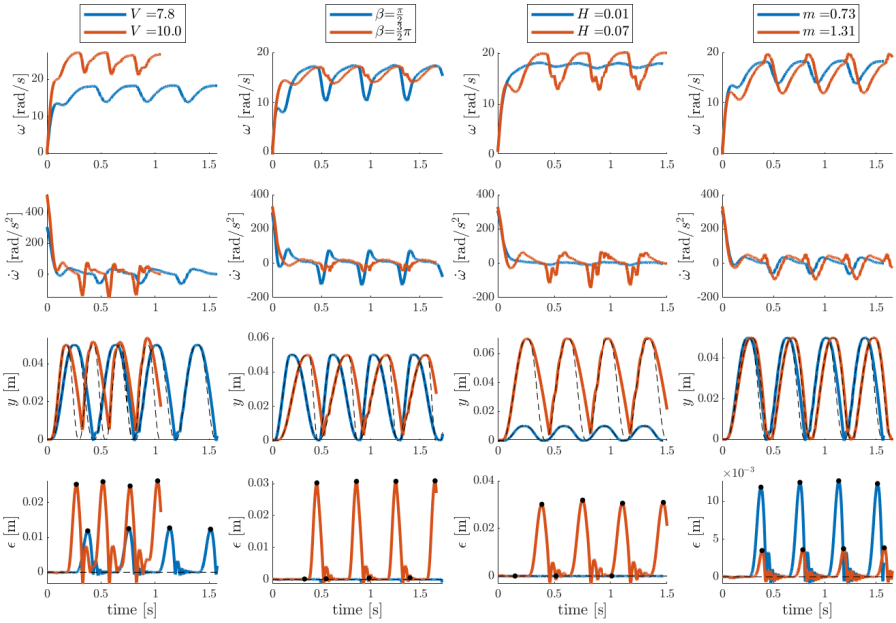


Figure 5.6: System measured for different system parameters. The non-listed properties are chosen $V = 7.77$ V, $m = 0.73$ kg, $\beta = \pi$ and $H = 0.05$ m.

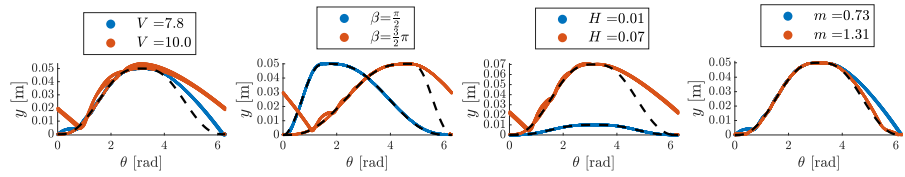


Figure 5.7: Difference between the displacement function h and the actual followed trajectory y for different system parameters. The non-listed properties are chosen $V = 7.77$ V, $m = 0.73$ kg, $\beta = \pi$ and $H = 0.05$ m.

The influence of the system parameters is assessed by performing a wide range of experiments. Each of the 20 cams, characterized by H and β, is used in combination with 4 different masses m. This approach enabled 80 different system modifications $\mathbf{q}_s$ that are tested for 20 different voltages V and

spans a measurement campaign of 1600 experiments, as shown in Figure 5.8. For each of the experiments, motor measurements and corresponding displacements were captured at 2000 Hz. The resulting dataset is available at `https://github.com/wannesdegroote/cam-follower-dataset`. We define ϵ_M as the maximum detachment variable observed during a period, illustrated by the black dots in Fig. 5.6. The average of the maximum detachment variable of all periods is denoted by $\bar{\epsilon}_M$. Empirically, we define the condition $\frac{\bar{\epsilon}_M}{H} > 1.3\%$ to define a signal being detached. An overview of all operating conditions for which follower jumps occurred is given in Figure 5.8. Note that for some conditions, the voltage V is too low so that a stall occurs, and the machine does not rotate.

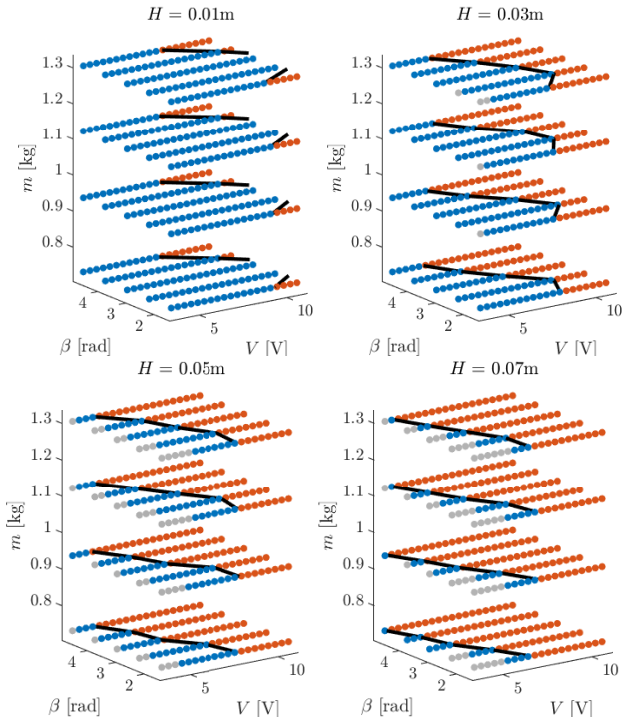


Figure 5.8: Overview of the occurrence of follower jumps, quantified by $\bar{\epsilon}_M$ (blue: contact, orange: jump, gray: stall).

5.3 Physics-inspired features

This research aims to predict the detachment variable ϵ, being the difference between the cam and follower motion, for given motor measurements (i.e., θ, ω ,$\dot{\omega}$) by learning directly from the data. We focus on providing some addi-

tional features to the model to increase the accuracy and generalization performances. The lumped parameter model shown in Fig. 5.2 can be used to deduce a simplified representation of the system dynamics. The dynamics of a theoretical knife-edge follower (or roller follower when we consider $h := h_p$ as discussed in Appendix 5.7), can be described by deriving the dynamic force equilibrium along the vertical axis. Note that the cam can only enforce an upward force $F \geq 0$ to the follower. In addition, the follower position $y(t)$ is physically restricted in space by $y(t) \geq h(\theta(t))$.

$$\begin{aligned} m\ddot{y}(t) + F_b(t) &= F(t) - mg \\ \text{s.t. } y(t) &\geq h(\theta(t)) \end{aligned} \tag{5.2}$$

For notational convenience, we omit the explicit notation of time dependency t, which is still implicitly assumed. The follower motion is constrained by a linear guide rail, inducing non-negligible friction forces. This phenomenon is modeled by imposing a friction model F_b that incorporates both a Coulomb and viscous damping component:

$$F_b = b_0 \text{sign}(\dot{y}) + b_1 \dot{y} \tag{5.3}$$

The friction parameters $b_0 = 0.5$ and $b_1 = 12$ are empirically determined as optimization parameters during a dedicated identification process. The dynamics of the cam displacement h are related to the behavior of the rotating cam and are derived by applying the chain rule of differentiation.

$$\begin{aligned} \dot{h} &= \frac{\mathrm{d}h}{\mathrm{d}\theta}\omega \\ \ddot{h} &= \frac{\mathrm{d}^2h}{\mathrm{d}\theta^2}\omega^2 + \frac{\mathrm{d}h}{\mathrm{d}\theta}\dot{\omega} \end{aligned} \tag{5.4}$$

The detachment variable $\epsilon = y - h$ defines the distance between the follower and the cam perimeter. In nominal circumstances, the follower perfectly tracks the cam, implying that $\epsilon = 0$. In this case, we can combine Eq. (5.2) and (5.4) to model the interface force F by

$$F^\dagger = m\left(\frac{\mathrm{d}^2h}{\mathrm{d}\theta^2}\omega^2 + \frac{\mathrm{d}h}{\mathrm{d}\theta}\dot{\omega} + g\right) + F_b\left(\frac{\mathrm{d}h}{\mathrm{d}\theta}\omega\right) \tag{5.5}$$

Negative accelerations $\ddot{h}$ tend to reduce the cam-follower interface force F, and if the acceleration is sufficiently large, surface contact between cam and follower can be lost, as shown in Fig. 5.6. Hence, we note the interface force $F = 0$ if $\epsilon > 0$. Subsequently, the force relations $F \geq 0$ can be modeled by

$$F = \begin{cases} F^\dagger, & \text{if } \epsilon = 0 \\ 0, & \text{if } \epsilon > 0 \end{cases} \tag{5.6}$$

The expression for $F^{\dagger}$ in (5.5) can be a powerful feature for monitoring the occurrence of follower jumps. Figure 5.9a illustrates an example for which continuous contact is assured (i.e., $\epsilon \approx 0$). The derived values for $F^{\dagger}$ correctly remain positive for all timesteps. In contrast, Fig. 5.9b shows an example for which follower jumps occur (i.e., $\epsilon > 0$). The metric $F^{\dagger}$ achieves unfeasible negative values, indicating that contact between cam and follower is lost the moment that $F^{\dagger} \approx 0$ holds, shown by the red line.

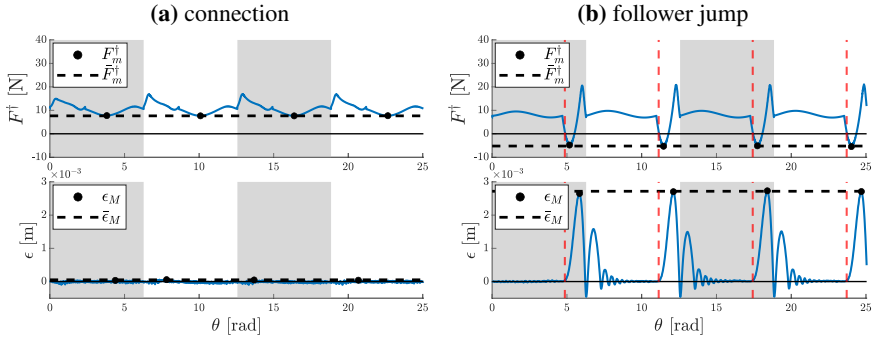


Figure 5.9: Example of $F^{\dagger}$ being used as an accurate metric to detect the occurrence of follower jumps. The value of $\bar{F}^{\dagger}$ stays positive if contact is assured (i.e., $\bar{\epsilon}_M \approx 0$) and becomes negative if a jump occurs (i.e., $\bar{\epsilon}_M > 0$).

The potential of this metric to be used as a physics-inspired input feature to predict the trajectory of ϵ, depends on the metric quality over the entire dataset. Consequently, we validate the accuracy of this metric by calculating $F^{\dagger}$ for all experiments. Consequently, we derive the minimum value for each period $F_m^{\dagger}$ and the corresponding average value $\bar{F}_m^{\dagger}$ for the entire time signal (cf. Fig. 5.9). Figure 5.10 illustrates $\bar{F}_m^{\dagger}$ for all experiments. The blue line indicates the point for which $\bar{F}_m^{\dagger} = 0$ and, thus, the follower jump begins.

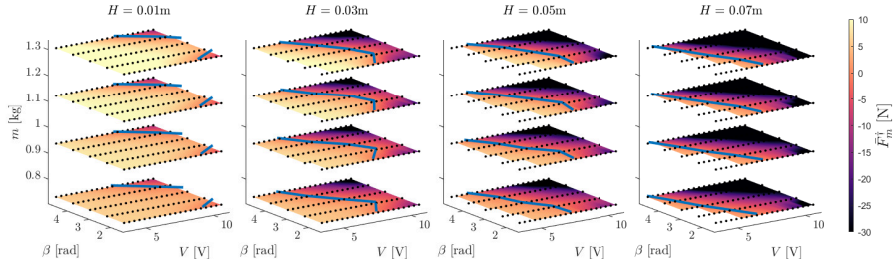


Figure 5.10: Overview of the derived metric $\bar{F}_m^{\dagger}$ derived for each experiment.

The accuracy of $\bar{F}_m^{\dagger}$ is shown in Figure 5.11. The idea is that the value of

$\bar{F}_m^\dagger$ should be positive if $\bar{\epsilon}_M \approx 0$ and strongly negative if $\bar{\epsilon}_M \gg 0$. This property implies that in an ideal case, all values should be located in the upper left (i.e., contact) or lower right (i.e., jump) corner. This figure illustrates that although there are some misclassifications (orange dots), the metric $F^\dagger$ is highly correlated to the measured motion discrepancy ϵ and, thus, a good candidate to be used as input to the prediction model.

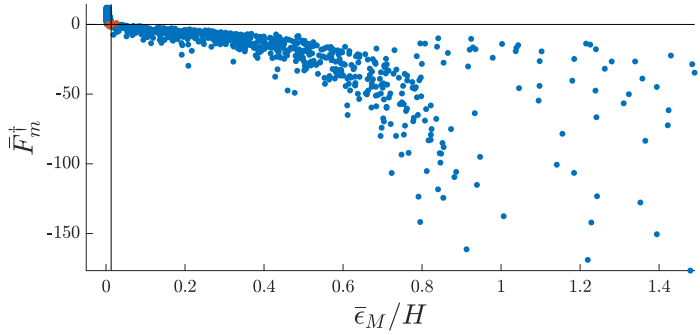


Figure 5.11: Comparison between the physics-inspired features $\bar{F}_m^\dagger$ with the measured detachment variables $\bar{\epsilon}_M$ for all experiments.

5.4 Physics-inspired data-driven model for the prediction of follower jumps in cam-follower mechanisms

This research studies the ability to monitor the detachment variable ϵ, which is typically not directly measurable, for given motor measurements (i.e., θ, ω and $\dot{\omega}$) for varying system properties $\mathbf{q}_s = \{m, H, \beta\}$. The first objective of this chapter aims for improved prediction performances by adding physics-inspired features (e.g. $F^\dagger$, h, $\dot{h}$...) to the motor measurements, resulting in an overall model input $\boldsymbol{\tau}_k$ at time instance k. It is, however, not possible to determine a direct relation between a momentaneous sample $\boldsymbol{\tau}_k$ and the corresponding value for ϵ_k. Note that the height of a follower jump (i.e., $\epsilon > 0$) does not depend on the momentaneous rotational measurements but relies on the energy build-up created when contact was still assured (i.e., $\epsilon = 0$). Therefore, we consider a modified recurrent neural network (RNN), defined by $\mathcal{R}$, that can process an entire trajectory sequence $\{\boldsymbol{\tau}_j\}_{j=1}^{j=k}$ by means of an internal memory, as will be discussed in Section 5.4.1 and 5.4.2.

$$\{\epsilon_1, \ldots, \epsilon_k\} = \mathcal{R}(\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_k; \mathbf{q}_s) \tag{5.7}$$

These models typically ingrain many trainable parameters that should be optimized to reduce the discrepancy between the actual and predicted output pre-

diction, referred to as a loss function. In practice, RNN models have gained popularity due to their possibility to deduce an analytical gradient of the loss function with respect to the model parameters, derived via backpropagation through time (BPTT) [48]. This gradient can be calculated for each timestep, enabling an iterative optimization process of the model parameters. However, in practice, the optimization process of RNNs that incorporate longer trajectory sequences is plagued by a numerical burden because the internal feedback loops can cause excessive attenuations/amplifications of the gradient information [49].

5.4.1 Long short term memory (LSTM) network

To predict the momentaneous detachment variable ϵ_k, the RNN should cope with long trajectory sequences that incorporate data prior to the jump initiation. Therefore, we implement an LSTM neural network, which is widely used to learn the temporal structure of long horizon time-series data [50]. In contrast to traditional RNN architectures, these topologies do not treat each time step as of equal importance. These network architectures learn to distill important events in timeseries by selectively remembering patterns for a long duration of time. The network can ignore irrelevant inputs so that the gradient flows through the network without vanishing. Figure 5.12 shows the architecture of an LSTM cell. The σ and $\tanh$ symbols indicate, respectively, the sigmoid and the hyperbolic tangent function. The memory of the LSTM cell is defined by a cell state $\mathbf{c}_k$ and hidden state $\mathbf{s}_k$, which are updated during each time step. The external input $\boldsymbol{\tau}_k$ is fed to the model for each time instance k. Consequently, the cell provides the corresponding cell output $\mathbf{s}_k$.

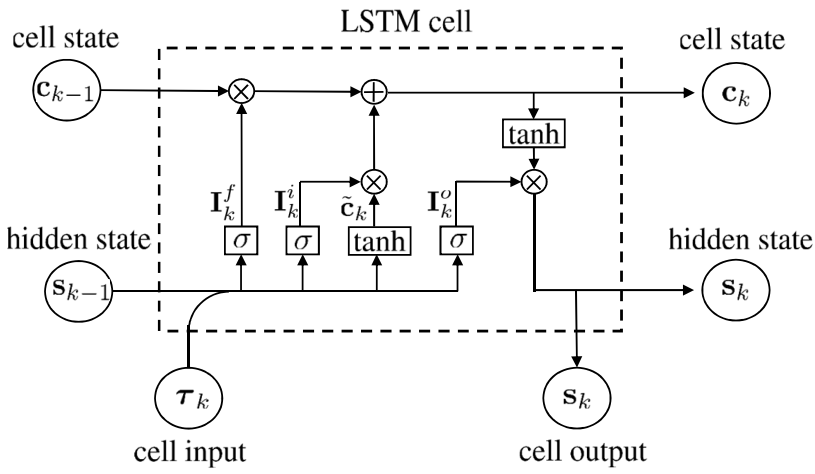


Figure 5.12: Detailed overview of the internal operations of an LSTM cell.

The general concept of an LSTM cell involves combining the information of the hidden state $\mathbf{s}_{k-1}$ of the prior time step with the new input $\boldsymbol{\tau}_k$ to update the cell state $\mathbf{c}_k$. Consequently, a filtered version of $\mathbf{c}_k$ is used to update the hidden state $\mathbf{s}_k$, which also serves as output of the cell.

The LSTM cell contains three so-called gates that regulate the influence of new information in $\boldsymbol{\tau}_k$ to the cell and hidden state. Each of these gates includes a neural layer that exploits the σ operator to create a vector, denoted by $\mathbf{I}_k$, that contains values between 0 and 1. Consequently, each element in $\mathbf{I}_k$ determines whether the corresponding state cell element should be (partially) maintained or erased.

The first gate, referred to as the *forget gate*, determines during each time instance k what part of the information of the prior cell state $\mathbf{c}_{k-1}$ becomes irrelevant, based on the new information in $\boldsymbol{\tau}_k$ and the prior hidden state $\mathbf{s}_{k-1}$. Mathematically, this operation can be denoted as:

$$\mathbf{I}_k^f = \sigma(\mathrm{W}_1^s \mathbf{s}_{k-1} + \mathrm{W}_1^\tau \boldsymbol{\tau}_k + \mathbf{b}_1) \tag{5.8}$$

The *input gate* determines the relevant information ingrained in the combination of the input $\boldsymbol{\tau}_k$ and the prior hidden state $\mathbf{s}_{k-1}$. The incorporation of this new information to the cell state consists of two mathematical operations. First, the vector $\mathbf{I}_k^i$ is calculated, which determines the elements of $\mathbf{c}_{k-1}$ that should be updated by the new information. The actual information that should be added to the prior cell state $\mathbf{c}_{k-1}$ is typically included in $\tilde{\mathbf{c}}_k$. The hyperbolic tangent operator compacts the information included in the prior hidden state $\mathbf{s}_{k-1}$ and current input $\boldsymbol{\tau}_k$ between -1 and 1, therefore helping to regulate the network.

$$\begin{aligned} \mathbf{I}_k^i &= \sigma(\mathrm{W}_2^s \mathbf{s}_{k-1} + \mathrm{W}_2^\tau \boldsymbol{\tau}_k + \mathbf{b}_2) \\ \tilde{\mathbf{c}}_k &= \tanh(\mathrm{W}_3^s \mathbf{s}_{k-1} + \mathrm{W}_3^\tau \boldsymbol{\tau}_k + \mathbf{b}_3) \end{aligned} \tag{5.9}$$

The information obtained from the forget- and input gate can be combined to determine the updated cell state $\mathbf{c}_k$. The sigmoid outputs $\mathbf{I}_k^f$ and $\mathbf{I}_k^i$ determine, respectively, what information of the prior cell state $\mathbf{c}_{k-1}$ and the new information in $\tilde{\mathbf{c}}_k$ should be included in the updated cell state $\mathbf{c}_k$ by performing a element-wise multiplication:

$$\mathbf{c}_k = \mathbf{I}_k^f \circ \mathbf{c}_{k-1} + \mathbf{I}_k^i \circ \tilde{\mathbf{c}}_k \tag{5.10}$$

The last operation involves the *output gate*, which defines the output $\mathbf{s}_k$ of the LSTM cell. This is a filtered version of the updated cell state $\mathbf{c}_k$. The elements that should be omitted from the cell state $\mathbf{c}_k$ are defined in the vector $\mathbf{I}_k^o$, which in turn depends on the information in $\boldsymbol{\tau}_k$ and $\mathbf{s}_{k-1}$.

$$\begin{aligned} \mathbf{I}_k^o &= \sigma(\mathrm{W}_4^s \mathbf{s}_{k-1} + \mathrm{W}_4^\tau \boldsymbol{\tau}_k + \mathbf{b}_4) \\ \mathbf{s}_k &= \mathbf{I}_k^o \circ \tanh(\mathbf{c}_k) \end{aligned} \tag{5.11}$$

In conclusion, the memory of the LSTM cell is defined by a cell state $\mathbf{c}_k \in \mathbb{R}^{n_s}$ and hidden state $\mathbf{s}_k \in \mathbb{R}^{n_s}$, which are updated for each time step. The initial cell state $\mathbf{c}_0$ and initial hidden state $\mathbf{s}_0$ are initialized by zero. Having an input $\boldsymbol{\tau}_k \in \mathbb{R}^{n_i}$, then the trainable weight matrices can be defined by $\mathrm{W}_j^s \in \mathbb{R}^{n_s \times n_s}$ and $\mathrm{W}_j^\tau \in \mathbb{R}^{n_s \times n_i}$ for $j \in \{1, .., 4\}$. The corresponding bias vectors are defined by $\mathbf{b}_j \in \mathbb{R}^{n_s}$ for $j \in \{1, .., 4\}$..

5.4.2 Implementation of the prediction model

The LSTM cell is considered a key subnetwork in the overall RNN model $\mathcal{R}$ that distills the importance information of $\{\boldsymbol{\tau}_j\}_{j=1}^{j=k}$ into a sequence of hidden states $\{\mathbf{s}_j\}_{j=1}^{j=k}$. Figure 5.13 gives an overview of the overall network model $\mathcal{R}$. The dimension of $\mathbf{s}$ (i.e., n_s) defines the complexity of the LSTM cell and, consequently, does not match the target data, being the one-dimensional detachment variable ϵ. Therefore, the LSTM cell is succeeded by a static neural network η to map the cell output $\mathbf{s}_k$ to the corresponding output ϵ_k. At each time instance k, the system parameters $\mathbf{q}_s \in \mathbb{R}^{n_q}$ are added to $\mathbf{s}_k$, to inform η about the system configuration. In practice, η is implemented as a one-hidden-layer rectified linear unit (ReLU) network [51, 52]:

$$\begin{aligned} \hat{\epsilon}_k &= \mathrm{W}_6 \Upsilon \left(\mathrm{W}_5^s \mathbf{s}_k + \mathrm{W}_5^q \mathbf{q}_s + \mathbf{b}_5 \right) + b_6 \\ \Upsilon(\cdot) &= \max\left(0, \cdot\right) \end{aligned} \tag{5.12}$$

This static mapping is defined by its weight matrices $\mathrm{W}_5^s \in \mathbb{R}^{n_\eta \times n_s}$, $\mathrm{W}_5^q \in \mathbb{R}^{n_\eta \times n_q}$, $\mathrm{W}_6 \in \mathbb{R}^{1 \times n_\eta}$ and corresponding bias vectors $\mathbf{b}_5 \in \mathbb{R}^{n_\eta}$, $b_6 \in \mathbb{R}$. All weights W and biases $\mathbf{b}$ in $\mathcal{R}$ are simultaneously optimized by minimizing the mean squared error (MSE) between the predicted $\hat{\boldsymbol{\epsilon}}_k$ and measured $\boldsymbol{\epsilon}_k$ signals by means of an Adam optimizer [53]. In practice, this entire framework is implemented by using the Keras API [54] with TensorFlow [55] backend in Python. This library employs automatic differentiation, which provides analytical expressions for the gradient, enabling an enhanced interface for deep learning approaches. The complexity, and thus the predictive capabilities, of the data-driven model $\mathcal{R}$ is determined by the hyperparameters n_s and n_η. In practice, we found that the model has sufficient flexibility when we choose $n_s = 32$ and $n_\eta = 32$. However, to prevent the neural network from overfitting, a dropout regularization technique is applied on the hidden layer of η [56]. This technique randomly ignores certain nodes in a layer during each training iteration, leading to an overall better generalization of the network. Due to the simplicity of $\mathcal{R}$, we implemented a probability of 5% for each node to be ignored. The model $\mathcal{R}$ can be evaluated on sequences of any length; however, the gradient updates during training are averaged along various samples contained in mini-batches of 32 trajectory samples. Therefore, additional zeros are added, commonly referred to as zero-padding [57],

to equalize the sequence length during training. Figure 5.14 illustrates the different sequence lengths of all trajectories. To avoid an unnecessary amount of padded timesteps in each trajectory, we truncate the signals at 4000 steps, enabling a clear trade-off between signals at which jumps occur and signals at which full contact remains. In addition, we found that due to the high sampling rate, the training time becomes excessively long. Therefore, we downsample all trajectories, having an initial sampling rate of 2000 Hz, to trajectory sequences at 100 Hz. In addition, all signals are scaled by their variance to cope with the different sizes of magnitude between the features.

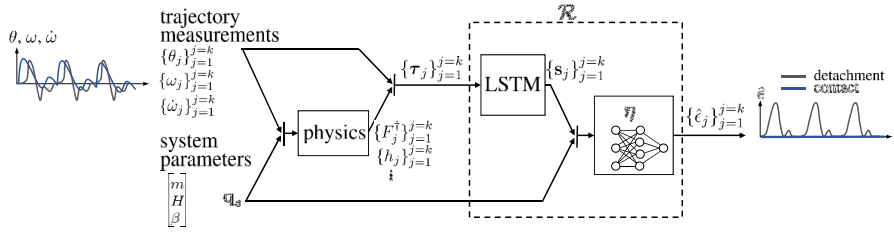


Figure 5.13: Overview of the recurrent neural network model used to predict the follower jumps.

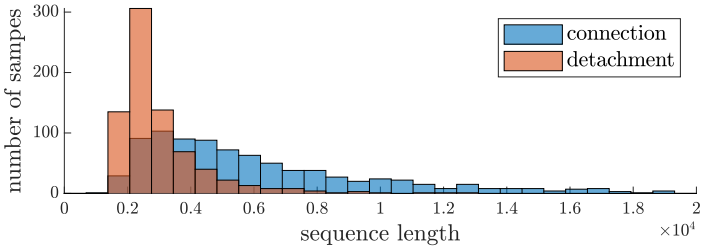


Figure 5.14: Overview of the different sequence lengths of the original trajectories sampled at 2000 Hz.

5.4.3 Prediction results

The neural network model $\mathcal{R}$ predicts the trajectory of the detachment variable $\{\epsilon_j\}_{j=1}^{j=k}$, for given input sequence and system parameters defined in $\mathbf{q}_s$. We assume that all the system properties $\mathbf{q}_s = \{m, H, \beta\}$ are provided to the model unless otherwise stated. Emphasis lies on analyzing the effect of including physics-inspired features in $\boldsymbol{\tau}$ on the generalization performances of the model. The variables θ, ω and $\dot{\omega}$ are directly derived from the rotary encoder measurements. The angles θ are replaced by their goniometric properties to constrain the input space of the neural network. The first physics-inspired

feature is the interface force metric $F^{\dagger}$, defined in (5.5), that describes the tendency of the follower to detach from the cam. In addition, experiments are performed that translate the cam properties (i.e., H and β) into the cam features h, $\frac{dh}{d\theta}$ and $\frac{d^2h}{d\theta^2}$ for each rotational position θ, as discussed in Appendix 5.7. Lastly, we validate the influence of transforming all rotary dynamics into the linear references (i.e., $\dot{h}$ and $\ddot{h}$), based on the relations described in Eq. (5.4). Figure 5.15 illustrates the influence of using different features in $\boldsymbol{\tau}$ on some particular samples from the test set. Each prediction sequence $\{\hat{\epsilon}_j\}_{j=1}^{j=k}$ presents the median trajectory prediction, obtained by retraining the model 10 times on the provided training dataset, which consists of 5% random samples of the overall dataset. The prediction accuracy of a specific trajectory prediction is quantified by the mean squared error (MSE). Although not explicitly mentioned, we will always implicitly assume the median behavior of $\{\hat{\epsilon}_j\}_{j=1}^{j=k}$ when comparing prediction performances between different experiments. Note that the model correctly predicts the occurrence of a follower jump while also predicting $\hat{\epsilon} \approx 0$ if continuous contact is assured.

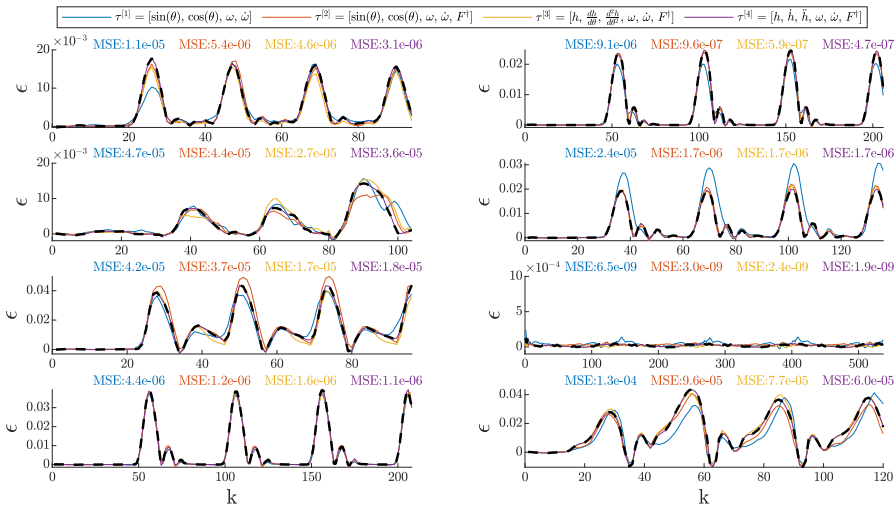


Figure 5.15: Trajectory predictions of the detachment variable ϵ for different amount of physics-inspired features included in $\boldsymbol{\tau}$ (resampled at 100 Hz).

To make a fair comparison between the different models, we summarize the prediction performances (MSE) of the test trajectories (i.e., excluded from the training set) by the quartiles shown in Fig. 5.16. The first experiment (left) illustrates the prediction performances for including different amounts of data into the (uniformly distributed) training dataset. The samples are randomly chosen from the grid shown in Fig. 5.8, whereas the remaining samples are used for testing purposes. The results clearly indicate that the more physics-

inspired features we include, the better the prediction results. Moreover, one can observe that the effect is more pronounced (note the logarithmic scale) when a scarce training dataset is used. These interpolation results do not, however, say anything about the generalization performances of the model because the training data is uniformly distributed along the parameter space. Therefore, a second experiment is performed for which the model is trained by data retrieved from specific system configurations (e.g., cams or follower masses) and validated on a group of fully excluded system settings. The results in Fig. 5.16 (right) illustrate the prediction performances for subsequently dividing the dataset based on the system parameters H, β and m. These results indicate that compared to the prior experiment that incorporated uniform training data, the inclusion of adding physics-inspired features is even more pronounced when the training data is not uniformly distributed along the parameter space.

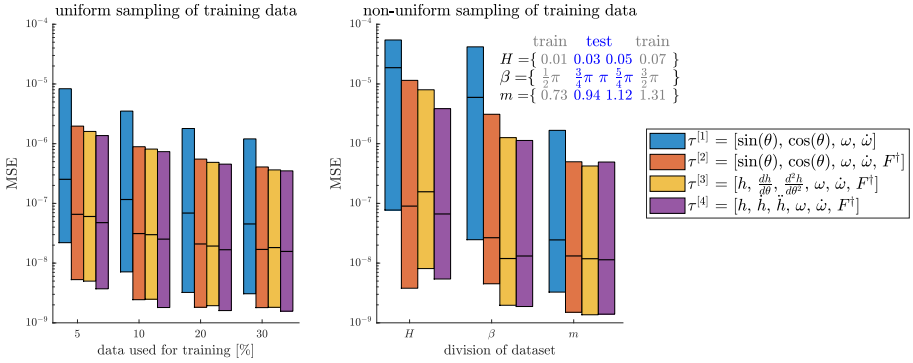


Figure 5.16: Influence of the inclusion of physics-inspired features in $\boldsymbol{\tau}$ by learning the model from a uniformly distributed (left) and a non-uniformly distributed (right) training dataset. The dataset used for each experiment in the right plot is divided according to one of the system parameters in $\mathbf{q}_s = \{m, H, \beta\}$, following the data split presented above.

An interesting metric of the predicted trajectory of ϵ is the average maximum value $\bar{\epsilon}_M$, which is indicative of the hazardousness of the corresponding operating condition. Consequently, we derived the predicted value of $\bar{\epsilon}_M$ by post-processing the trajectory sequences obtained by $\mathcal{R}$. Figure 5.17a and 5.17b illustrate the prediction accuracy for models trained on a datasplit based on H and β, respectively. These results clearly show the improved accuracy due to the inclusion of physics-inspired features in $\boldsymbol{\tau}$. Note that one can observe that the model based on raw data (i.e., $\boldsymbol{\tau}^{[1]}$) often predicts a jump phenomenon when there is actually full contact and vice-versa. These false estimations of the system behavior are fully diminished when physics-inspired features are given to the model. Consequently, these additional features help

to generalize the prediction model by providing the required information when predicting the trajectory of ϵ for unseen system modifications.

(a) training: $H \in \{0.01, 0.07\}$, testing: $H \in \{0.03, 0.05\}$

(b) training: $\beta \in \{\frac{1}{2}\pi, \frac{3}{2}\pi\}$, testing: $\beta \in \{\frac{3}{4}\pi, \pi, \frac{5}{4}\pi\}$

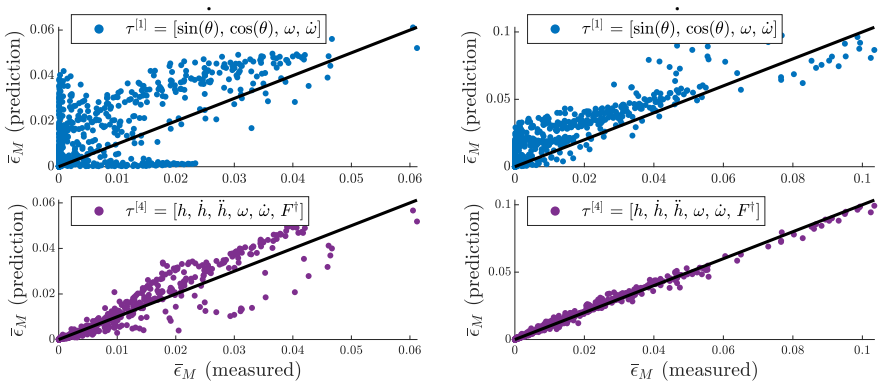


Figure 5.17: Accuracy of the average maximum detachment variable $\bar{\epsilon}_M$ of test trajectories. Dataset divided based on H (left) and β (right).

The aforementioned results discussed the influence of changing the features in $\boldsymbol{\tau}$. However, the information given by $\mathbf{q}_s$ remained fixed ($\mathbf{q}_s = \{m, H, \beta\}$). Therefore, we repeat the experiments by training a model $\mathcal{R}$ on 5% of the (uniformly distributed) dataset, but this time we do not incorporate direct information about the system parameters (i.e., empty set $\mathbf{q}_s = \emptyset$). Figure 5.18 shows some histograms that illustrate the difference between the MSE of the test trajectories trained with and without the system parameters. For the sake of completeness, we only incorporate the test signals for which a jump occurs, indicated by the orange dots in Fig. 5.8. The quartiles are indicated by the vertical black lines. These results show that the model trained on raw measurements (i.e., $\boldsymbol{\tau}^{[1]}$) deteriorates significantly when the information in $\mathbf{q}_s$ is omitted. By contrast, the accuracy of the models that include the physics-inspired features, derived from $\mathbf{q}_s$, barely change. This finding empirically shows that the model $\mathcal{R}$ can extract the required information about the system modifications directly from the physics-inspired features in $\boldsymbol{\tau}$ and, therefore, relies less on $\mathbf{q}_s$.

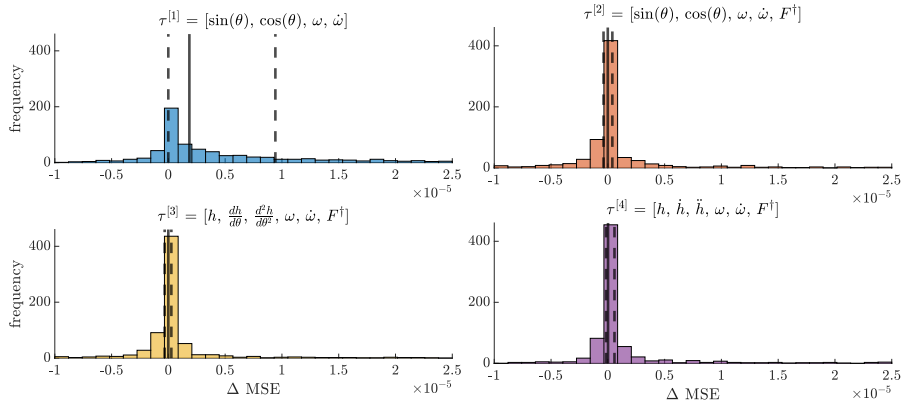


Figure 5.18: Comparison of ΔMSE = MSE($\mathbf{q_s} = \emptyset$) -MSE($\mathbf{q_s} = \{m, H, \beta\}$) for the trajectories at which a jump occurs.

5.5 Explanation models to interpret the model predictions

The aforementioned results indicate the influence of adding physics-inspired features to the model input trajectory $\boldsymbol{\tau}$. We empirically observed that the model performs more accurately and clearly benefits from the information given by the physics-inspired features. However, this does not mean that the model actually prioritizes the incorporation of physics-inspired expert knowledge. Therefore, as a second objective of this chapter, we present an elaborate analysis in which we aim to gain further insights into the inclusion of physics-inspired relations in recurrent neural networks.

5.5.1 Methodology

Additive feature attribution methods

The complexity of a deep learning model such as an LSTM does not allow to interpret the prediction directly from the model. Due to the ingrained complexity and the excessive amount of parameters, we are forced to deduce a simplified, so-called, explanation model g that locally describes the influence of each input. For notational convenience, we consider a prediction model f that accepts an input vector $\mathbf{x} \in \mathbb{R}^n$ as an abstraction of our presented recurrent model $\mathcal{R}$. This chapter places emphasis on a specific class of explanation models g (i.e., additive feature attribution methods) [42]. These methods define the local approximator $g(\mathbf{z})$ as a linear function of binary values $\mathbf{z} \in \{0, 1\}^{n_z}$. Each so-called simplified input feature in $\mathbf{z}$ can typically be interpreted as being one if a (group of) corresponding feature(s) in $\mathbf{x}$ is "present" and zero

otherwise. For this particular case, we consider $n_z = n$, which assigns a specific contribution $\phi_i \in \mathbb{R}$ to each corresponding feature x_i. Consequently, the expression for the explanation model $g(\mathbf{z})$ can be given by:

$$g(\mathbf{z}) = \phi_0 + \sum_{i=1}^{n} \phi_i z_i \tag{5.13}$$

Note that by summing all the effects of the features attributions ϕ_i, we approximate the output $f(\mathbf{x})$ of the original model. This implies that there exists a specific input $\mathbf{z}^*$, typically an all-ones vector, for which one can define a relation $\mathbf{x} = \mathcal{M}_x(\mathbf{z}^*)$ that maps the simplified input $\mathbf{z}^*$ to the original feature vector $\mathbf{x}$. If the original prediction model f is linear, the features can be directly deduced from the model's coefficients [58]. However, for highly nonlinear models, the task to find values for ϕ_i that fairly quantify the contribution of a feature is less trivial. In practice, as discussed in [42], various existing explanation methods, although not always intended by the developers, can be reformulated to an additive feature attribution method that satisfies expression (5.13). Currently used methodologies such as LIME [43], Layer-Wise Relevance Propagation [41], DeepLIFT [39], Shapley Regression Values [59], Shapley Sampling Values [58], and Quantitative Input Influence [60] use the same underlying explanation model g.

DeepLIFT algorithm

A particular additive feature attribution method that raises our interest is the DeepLIFT (Deep Learning Important FeaTures) algorithm due to its proven effectiveness on deep learning models [39]. This framework aims to explain, for a given difference between an input feature $\mathbf{x}$ and a chosen reference $\mathbf{r}$, the difference between the corresponding outputs $f(\mathbf{x})$ and $f(\mathbf{r})$. In accordance with the expression of $g(\mathbf{z})$ in Eq. (5.13), we define $\phi_0 = f(\mathbf{r})$. The mapping $\mathcal{M}_x$, which converts the multivariate binary values $\mathbf{z}$ to the original input domain of $\mathbf{x}$, is defined in such way that it assigns x_i to each element for which $z_i = 1$ holds and assigns the reference value r_i otherwise. Trivially, the mapping $\mathcal{M}_x$ operates on vector quantities elementwise:

$$\mathcal{M}_x(\mathbf{z}) = \begin{cases} x_i & \text{if } z_i = 1 \\ r_i & \text{if } z_i = 0 \end{cases} \tag{5.14}$$

Note that if we define $\mathbf{z}^*$ as being an all-ones vector, the required relation $\mathbf{x} = \mathcal{M}_x(\mathbf{z}^*)$ still holds. Next, we define a multiplier $m_{\Delta x_i \Delta y} = \frac{\phi_i}{\Delta x_i}$, which captures the contribution ϕ_i divided by the difference $\Delta x_i = x_i - r_i$. Once the value of each multiplier $m_{\Delta x_i \Delta y}$ is determined, the derivation of the ϕ_i

becomes trivial:

$$\phi_i = m_{\Delta x_i \Delta y} \cdot (x_i - r_i) \tag{5.15}$$

The ingenuity of the DeepLIFT algorithm lies in the methodology to derive the multiplier $m_{\Delta x_i \Delta y}$ in such a way that a fair contribution ϕ_i to all features is assigned and $f(\mathbf{x}) = f(\mathbf{z}^*) = \phi_0 + \sum_{i=1}^{n} \phi_i$ holds. The DeepLIFT framework is inspired on partial derivatives that quantify an infinitesimal change in the output caused by an infinitesimal change in the input. Note that for a small difference $\Delta x_i \rightarrow 0$, the value of $m_{\Delta x_i \Delta y}$ approximates the partial derivative $\frac{\partial \phi_i}{\partial x_i}$. In practice, deep learning algorithms have many nonlinear activation functions (e.g., ReLU) that have a gradient equal to zero for wide ranges. These properties can zero out important features, leading to wrong estimations. The DeepLIFT algorithm copes with this issue by considering finite differences Δx_i instead of infinitesimal ones. More precisely, the algorithm compares the activation for each neuron to its reference (i.e., based on the reference input $\mathbf{r}$) and assigns a contribution score according to the difference. Furthermore, the obtained scores of each neuron can, in analogy to the training process of the neural network, be backpropagated to obtain the contribution of each input feature to the model output. For example, consider a one-layer neural network with neurons $\{\mathcal{N}_1, \mathcal{N}_2 ...\}$ making a mapping between an input $\mathbf{x}$ to an output y; the overall multiplier of the network can be defined by:

$$m_{\Delta x_i \Delta y} = \sum_j m_{\Delta x_i \Delta \mathcal{N}_j} \cdot m_{\Delta \mathcal{N}_j \Delta y} \tag{5.16}$$

In analogy, the calculation of an entire deep learning network $m_{\Delta x_i \Delta y}$ essentially involves calculating the multipliers (i.e., linearization) of all individual neurons $\mathcal{N}$ and combining them according to a composition rule. In practice, the high nonlinearity of these neurons leads to improper results if the linearization would only imply a finite difference approach. Therefore, the DeepLIFT algorithm develops various ingenious implementations, such as giving separate consideration to positive and negative contributions, as well described in the literature [39].

SHAP (SHapley Additve exPlanation) values

Each of the aforementioned additive feature attribution methods results has their own solutions for the features attributions ϕ_i in (5.13). Although algorithms such as LIME and DeepLIFT can give intuitive results, we are never sure that the obtained values for ϕ_i represent the "actual" contributions. Therefore, Lundberg & Lee worked towards a unified approach and proposed three desirable properties that the obtained explanation model g should obey [42].

Property 1: Local accuracy

When approximating a function f for a specific input $\mathbf{x}$, the local accuracy property requires that the explanation model g evaluated at $\mathbf{z}^*$, for which $\mathbf{x} = \mathcal{M}_x(\mathbf{z}^*)$ holds, matches f exactly.

$$f(\mathbf{x}) = g(\mathbf{z}^*) = \phi_0 + \sum_i^n \phi_i z_i^* \tag{5.17}$$

Property 2: Missingness
The simplified input $\mathbf{z}^*$ could, in theory, have some zero entries (e.g., when a feature is constant for the entire dataset). This would imply that the corresponding feature attribution ϕ_i can be any value to satisfy the local accuracy property (5.17). Therefore, the missingness property states that features missing in the original input should have no impact.

$$z_i^* = 0 \Rightarrow \phi_i = 0 \tag{5.18}$$

Property 3: Consistency
The consistency property states that if a model changes so that the marginal contribution of a feature value increases or stays the same (regardless of other features), the input's attribution also increases or remains the same.

Let $f_x(\mathbf{z}) = f(\mathcal{M}_x(\mathbf{z}))$ and $\mathbf{z}_{ni}$ indicate that $z_i = 0$.
For any two models f and f' that satisfy:

$$f'_x(\mathbf{z}) - f'_x(\mathbf{z}_{ni}) \geq f_x(\mathbf{z}) - f_x(\mathbf{z}_{ni}) \tag{5.19}$$

for all inputs $\mathbf{z} \in \{0,1\}^n$, then:

$$\phi_i(f', x) \geq \phi_i(f, x)$$

The DeepLIFT algorithm, discussed in Section 5.5.1, only obeys properties 1 & 2, but cannot grantee consistency (property 3). Lundberg and Lee showed that there even exist only one solution for the attribution values ϕ_i of the explanation model g defined in (5.13) that satisfies properties 1, 2, and 3:

$$\phi_i(f, x) = \sum_{\mathbf{z} \subseteq \mathbf{z}^*} \frac{|\mathbf{z}|!(n - |\mathbf{z}| - 1)!}{n!} [f_x(\mathbf{z}) - f_x(\mathbf{z}_{ni})] \tag{5.20}$$

where $|\mathbf{z}|$ is the number of non-zero entries in $\mathbf{z}$, and $\mathbf{z} \subseteq \mathbf{z}^*$ represents all $\mathbf{z}$ vectors where all non-zero entries are a subset of the non-zero entries in $\mathbf{z}^*$. Recall that the function $f_x(\mathbf{z})$ is a short notation for $f(\mathcal{M}_x(\mathbf{z}))$. Consequently, this unique solution for the explanation model g in Eq. (5.13) is obtained by imposing:

$$f_x(\mathbf{z}) = f(\mathcal{M}_x(\mathbf{z})) = E[f(X)|X_S = \mathbf{x}_S] \tag{5.21}$$

for which X is a stochastic variable spanning the input space, and S indicates the set of nonzero indexes in $\mathbf{z}$. Here, we are essentially saying that the attributions ϕ_i are the Shapley values of a conditional expectation function of the original model $f(\mathbf{x})$. In addition, this means that for $\mathbf{z} = \mathbf{0}$, we obtain an empty set S , resulting in $\phi_0 = E[f(X)]$. Consequently, the SHAP values ϕ_i associated to an individual prediction can be interpreted as the attribution of each feature x_i towards the model prediction $f(\mathbf{x})$, as shown in Fig. 5.19.

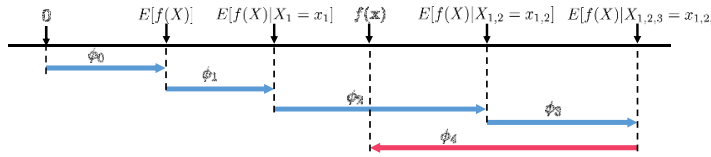


Figure 5.19: Schematic representation of the SHAP values, which attribute to each feature the change in the expected model prediction when conditioning on that feature. Hence, these values explain how to deviate from the base value $E[f(X)]$, which would be predicted if no features were known. Note that for nonlinear models or models with non-independent input features, the ordering depends on the order at which the features are added. Consequently, the SHAP values are obtained by averaging the ϕ_i values across all possible orderings.

Up to this point, the SHAP framework remains a theoretical concept because the precise conditional expectations cannot be easily determined. Lundberg & Lee state that the DeepLIFT algorithm described in Section 5.5.1 does not fulfill the consistency property (5.19). However, in [61], they demonstrate that the average solution of single-reference SHAP values approaches the true SHAP values for a given distribution. This property enabled the development of the DeepSHAP framework, which represents a layer-wise propagation of Shapley values that builds upon DeepLIFT. Therefore, the DeepSHAP framework provides a means to quantify the feature attribution ϕ_i according to the layer-wise propagation rules initiated in Section 5.5.1 while still fulfilling all three properties (5.17),(5.18) and (5.19). The practical implementation requires the definition of a so-called background set $\mathcal{B} = \{\mathbf{r}^{[1]}, \mathbf{r}^{[2]}, \ldots\}$, which contains all reference inputs $\mathbf{r}$ that are used to explain the output associated to $\mathbf{x}$. This way, the DeepSHAP framework approximates the SHAP values by averaging the solutions obtained for all reference inputs in $\mathcal{B}$.

DeepSHAP applied to the recurrent prediction model $\mathcal{R}$

The DeepSHAP framework is applied to the recurrent model $f \equiv \mathcal{R}$ to quantify the contribution of the physics-inspired features, responsible for

the enhanced predictions discussed in Section 5.4.3. We implemented the DeepSHAP library, provided by [42], on the considered model $\mathcal{R}$. We define $\mathbf{x}_k$ as the set of input features associated to the prediction $\hat{\epsilon}_k$. This contains the parameter settings $\mathbf{q}_s$ and the trajectory sequence $\{\boldsymbol{\tau}_j\}_{j=1}^{j=k}$, which does not include future values due to causality reasons. As shown in Fig. 5.20, the total amount of features in $\mathbf{x}_k$ to explain $\hat{\epsilon}_k$ equals $n_k = k \cdot n_\tau + n_q$. This results into an equal amount of SHAP values $\boldsymbol{\phi}$. Note that to know the overall attribution of a specific feature i in $\boldsymbol{\tau}$, we have to take the sum of all k timesteps to obtain the overall contribution $\Phi_{i,k}$.

$$\Phi_{i,k} = \sum_{j=1}^{j=k} \phi_{i,j} \tag{5.22}$$

This provides a set of attributions $\Phi_{i,k}$, which explains for each time step k the contribution to deviate from the expected value $E[f(\mathcal{B}_k)]$. However, for this research, we are not interested in explaining the deviation from the average prediction, but rather we want to determine the key inputs that cause a follower jump prediction (i.e., $\hat{\epsilon}_k > 0$). Therefore, we choose our background set $\mathcal{B}_k$ to contain all input references $\mathbf{r}_k$ associated to the trajectories at which continuous contact is assured (i.e., trajectory measurements associated to the blue dots in Fig. 5.8), imposing $\Phi_{0,k} = E[f(\mathcal{B}_k)] \approx 0$. Consequently, this approach enables a contrastive model g that explains for a given input sequence associated to a follower jump prediction how much each feature contributed to the corresponding model output.

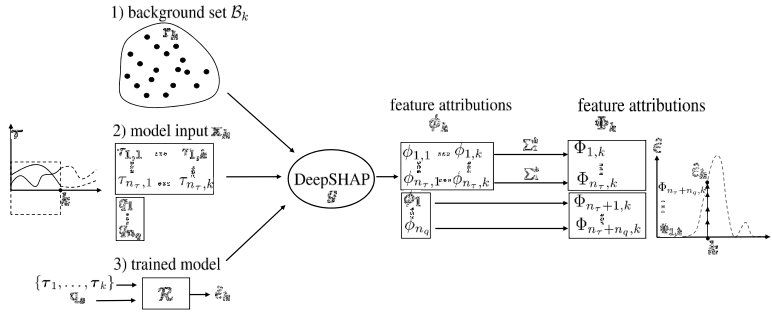


Figure 5.20: Schematic representation of the DeepSHAP framework, which is used to explain the importance of each feature on the corresponding output prediction.

5.5.2 Results of DeepSHAP

The DeepSHAP algorithm is applied on the recurrent network models $\mathcal{R}$, trained on 30% of the experimental dataset. A set of SHAP values

$\{\Phi_{i,k}\}_{i=1}^{i=n_\tau+n_q}$ that clarifies the contribution of each feature is deduced for each timestep k of the predicted follower jump. As previously explained, we only include the non-detached experiments in the background set, (i.e., $\Phi_{0,k} = E[f(\mathcal{B}_k)] \approx 0$)), enabling a contrastive explanation that quantifies how much each input feature contributed to the follower jump prediction (i.e., $\hat{\epsilon} > 0$). Figure 5.21 details a particular signal for which follower jumps occur. The prediction models $\mathcal{R}$ include the system parameters $\mathbf{q}_s = \{m, H, \beta\}$ and are trained for different trajectory sequences $\boldsymbol{\tau}$. In addition, we provide the (normalized) model inputs that are used as the input features in $\boldsymbol{\tau}$. Although

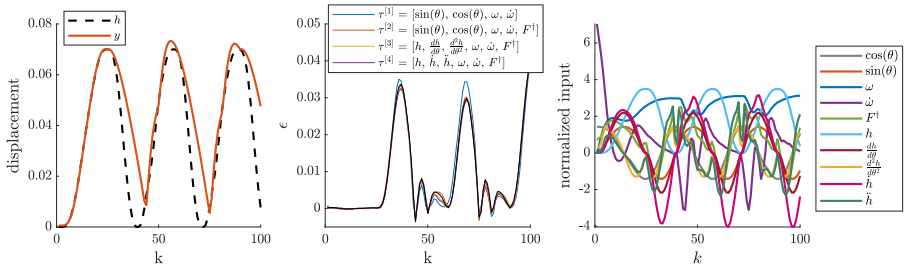


Figure 5.21: Left: measured displacement, center: predicted vs. measured follower jumps for different model inputs $\boldsymbol{\tau}$, right: (normalized) features to be incorporated in $\boldsymbol{\tau}$.

all models $\mathcal{R}$ clearly approximate the actual jump trajectory (indicated by the black line), they retrieved their information from different input sequences τ. The contribution of all input features, derived by the DeepSHAP framework, is shown in Fig. 5.22. Positive values are indicative of features that push the prediction higher, whereas negative contributions attenuate the jump prediction. The prediction of the model that only accepts raw measurements (i.e., $\boldsymbol{\tau}^{[1]}$) is substantiated by many inputs that are given a significant contribution. By contrast, the model in which we provide the feature $F^\dagger$ (i.e., $\boldsymbol{\tau}^{[2]}$) seems to have a significant shift in the feature attributions. The model clearly learned to assign high importance to the given physics-inspired feature $F^\dagger$. In addition, models that incorporate more physics-inspired features (i.e., $\boldsymbol{\tau}^{[3]}$ and $\boldsymbol{\tau}^{[4]}$) appear to further divide the contributions among these physics-inspired features. This finding clearly indicates that the model experiences the physics-inspired features as an advantage and, therefore, assigns high contributions. To obtain a clear overview of the importance a model assigns to each input feature, we calculated the (relative) SHAP values for all trajectories associated to operating conditions at which a jump occurs (cf. orange dots in Fig. 5.8). Figure 5.23 illustrates the corresponding average (relative) magnitudes of the SHAP values for different amounts of physics-inspired features included in $\boldsymbol{\tau}$. The model for which only raw measurements are used (i.e., $\boldsymbol{\tau}^{[1]}$) assigns high importance to both the information in $\boldsymbol{\tau}$ and $\mathbf{q}_s$. By contrast, we observe

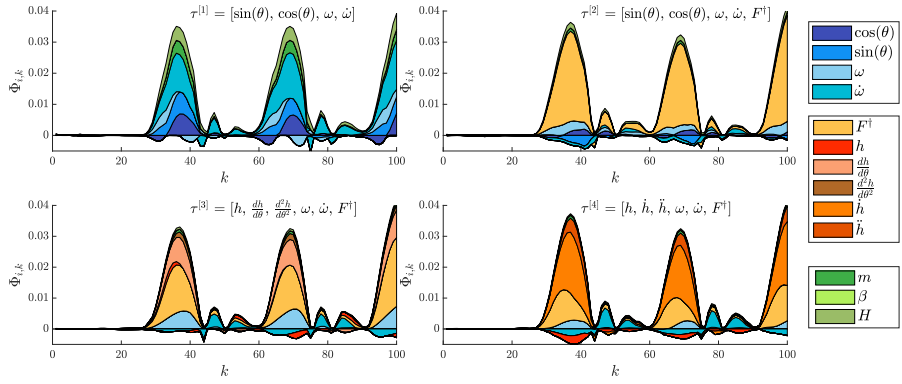


Figure 5.22: Contributions $\Phi_{i,k}$ of all prior model inputs i to obtain a model prediction at time instance k.

that once we begin to add more physics information, the model prefers to retrieve its required information from the physics-inspired features instead of the parameter settings in $\mathbf{q}_s$. These observations clearly illustrate that the model experiences the addition of physics-inspired features as extremely valuable and, therefore, prioritizes the incorporation of these features during the learning process, leading to the improved generalization performances shown in Section 5.4.3.

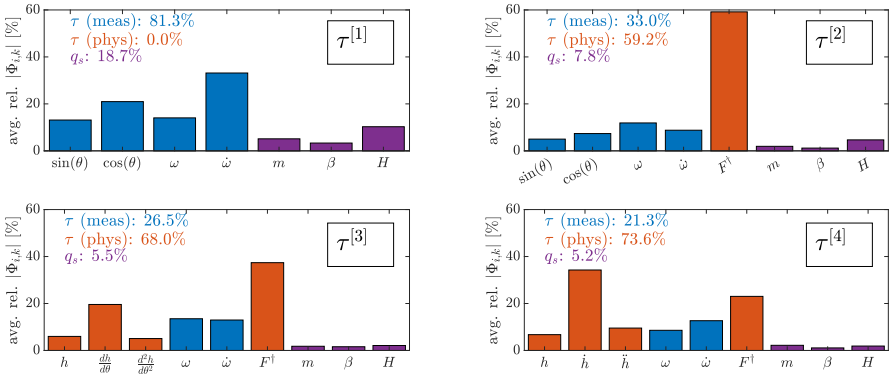


Figure 5.23: Overview of the relative contributions of all input features, averaged over all trajectories at which a jump occurs.

5.6 Conclusion

This chapter presents a detailed analysis of the contribution of physics-inspired features given as input to a recurrent data-driven prediction model of a cam-follower mechanism. The trajectory of the detachment variable, defined as the discrepancy between cam and follower motion, is learned by an LSTM neural network model that receives a combination of direct measurements, physics-inspired features, and system properties. The prediction performances are extensively tested on experimental data generated by a modular laboratory setup. The dataset is made open to allow reproducibility of the results. Timeseries sequences are measured for varying cam shapes, follower masses, and voltage inputs, enabling a full-factorial design of 1600 operating conditions. The objective of this research was twofold. First, we improved the prediction capabilities of an LSTM neural network model by including physics-inspired features to the input sequence. An elaborate empirical analysis on unseen data confirmed the enhanced generalization capabilities. The obtained improvements were more pronounced if the model was learned from a scarce dataset or a dataset for which the measurements were not uniformly distributed along the parameter space. Second, we hypothesized that the reason for the improved prediction capability lies with the addition of the physics-inspired features. Therefore, we determined in a second stage the contribution of all features within the multivariate timeseries on the output of a recurrent neural network model by means of the DeepSHAP framework. This way, we could objectively quantify the influence of physics-inspired features on the model predictions. This approach revealed that the model prioritizes the inclusion of physics-inspired expert knowledge, explaining the enhanced generalization performances that were observed on the dataset. The presented results can serve as an incentive to augment other data-driven prediction models with expert knowledge to enable improved monitoring of various other inconvenient phenomena occurring in mechatronic applications. Nevertheless, future research should be devoted to incorporate the influence of system variations, such as thermal effects and external disturbances, to assure robust monitoring over long time horizons.

5.7 Appendix: Characteristics of the cam-follower mechanism

The function h describes the motion of the follower when perfect connection between cam and follower is assured. The use of the theoretical displacement function h_c, based on the geometrical shape of the cam, is only valid when the follower has an infinitesimal contact point (i.e., knife-edge follower). In practice, the described setup has a high-carbon steel roller follower with radius

$r_f = 0.008$ m, leading to a line contact of 0.011 m with the 3D-printed (PLA) cams. Consequently, we define $h := h_p$, with h_p being the pitch curve that describes the trajectory of the center of the roller when contact with the cam is assured. The difference between the actual cam perimeter and the pitch curve for a base circle $r_b = 0.02$ m is graphically illustrated in Fig. 5.24. The expression of h_p can be directly deduced from h_c. First, we define the points $\mathbf{c}_0$, which describe the cam perimeter for an angular displacement $\theta = 0$:

$$\mathbf{c}_0(\phi) = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} (r0 + h_c(\phi))\sin(\phi) \\ (r0 + h_c(\phi))\cos(\phi) \end{bmatrix} \tag{5.23}$$

For the sake of completeness, a cam coordinate $\mathbf{c}$ w.r.t. the fixed reference frame for given rotation θ can be determined by applying the following linear transformation:

$$\mathbf{c}(\theta, \phi) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \mathbf{c_0}(\phi) \tag{5.24}$$

The expression of the pitch curve $\mathbf{p}_0$ (for $\theta = 0$) is determined by calculating the unit normal vectors on each cam perimeter coordinate $\mathbf{c}_0$. The center point $\mathbf{p}_0$ of the follower is then situated at a distance r_f along the normal unit vector. Mathematically, this process can be summarized in the following:

$$\mathbf{p_0}(\phi) = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x_c - r_f \frac{y'_c}{\sqrt{x'^2_c + y'^2_c}} \\ y_c + r_f \frac{x'_c}{\sqrt{x'^2_c + y'^2_c}} \end{bmatrix} \tag{5.25}$$

The derivative information x' and y' of the cam functions are analytically deduced from (5.23) as:

$$\begin{aligned} x'_c &= \frac{\mathrm{d}x_c}{\mathrm{d}\phi} = (r_0 + h_c)\cos(\phi) + \frac{\mathrm{d}h_c}{\mathrm{d}\phi}\sin(\phi) \\ y'_c &= \frac{\mathrm{d}y_c}{\mathrm{d}\phi} = -(r_0 + h_c)\sin(\phi) + \frac{\mathrm{d}h_c}{\mathrm{d}\phi}\cos(\phi) \end{aligned} \tag{5.26}$$

By analogy, the expression for $\mathbf{p}$ for any displacement angle θ can be determined by applying the same transformation described in (5.24). The actual displacement function h_p is numerically determined by fitting a prime circle, having radius r_p, through the obtained profile $\mathbf{p_0}$. Figure 5.25 compares the measurements of h based on the linear encoder with h_c and h_p, justifying the use of the expression of h_p to approximate the displacement function h. Note that by using cycloidal motion laws we obtain continuous derivatives of the displacement function during both the rise and fall phase of the motion cycle. In addition, both stages have a starting and ending acceleration equal to zero, leading to smooth transitions between rise and fall.

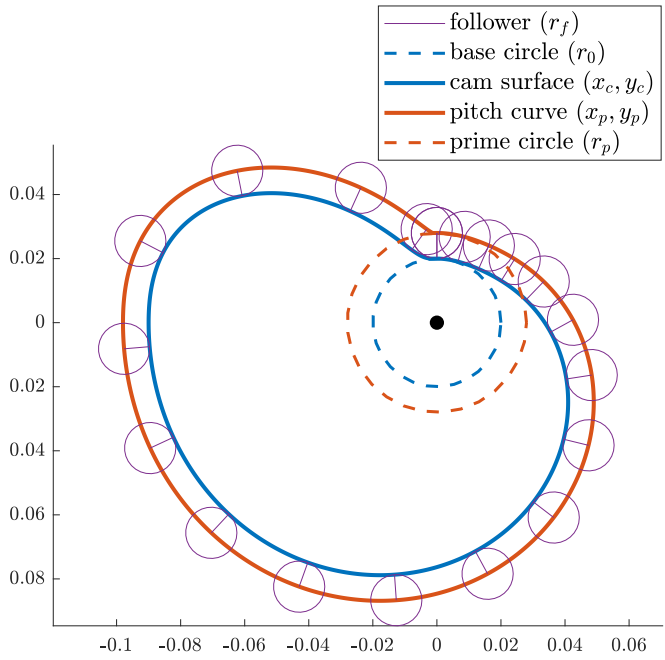


Figure 5.24: The pitch curve describes the trajectory of the center of the follower when there is contact between cam and follower ($\theta = 0$, $H = 0.07$, $\beta = \frac{3}{2}\pi$).

The cams shown in Fig. 5.4 are characterized by a wide range of design parameters, leading to the generation of a diverse dataset that encompasses a broad variety of cam dynamics. Consequently, we should validate the practical usefulness of the obtained cam-designs. For instance, there should be checked if the follower can track the cam perimeter in the concave regions (i.e., $\theta \approx 2\pi$) [62]. Therefore, we define the radius of curvature of the cam ρ_c as

$$\rho_c = \frac{\left((r_0 + h_c)^2 + (\frac{\mathrm{d}h_c}{\mathrm{d}\theta})^2\right)^{\frac{3}{2}}}{(r_0 + h_c)^2 + 2(\frac{\mathrm{d}h_c}{\mathrm{d}\theta})^2 - \frac{\mathrm{d}^2 h_c}{\mathrm{d}\theta^2}(r_0 + h_c)} \tag{5.27}$$

The value of $|\rho_c|$ should be larger than the radius of the follower r_f to assure that the follower can track the entire cam perimeter. The minimal values of the relative radius of curvature depicted in Table 5.1a illustrate that most cams fulfill this condition (i.e., $|\rho_c|/r_f > 1$). However, we can observe that two cams violate this condition, implying that these designs would not be feasible for practical use-cases. An additional design characteristic that should be checked

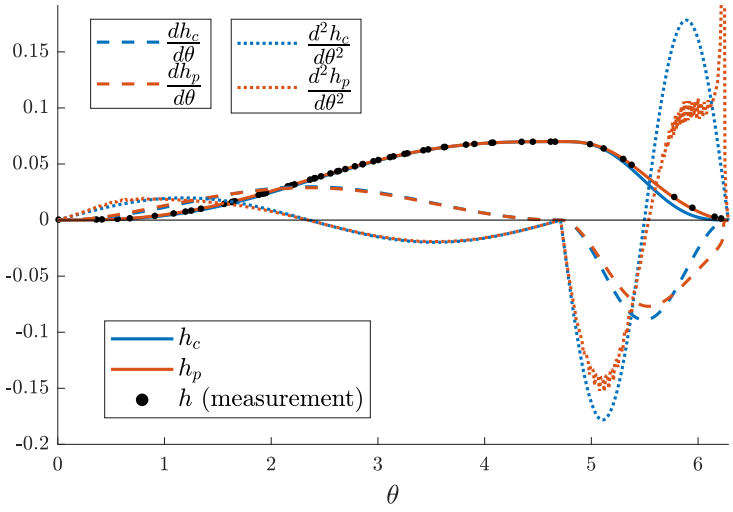


Figure 5.25: Comparison of the displacement function of the cam h_c and pitch curve h_p ($H = 0.07$, $\beta = \frac{3}{2}\pi$).

is the pressure angle α. This parameter is defined as

$$\alpha = \arctan\left(\frac{\frac{\mathrm{d}h_p}{\mathrm{d}\theta}}{r_p + h_p}\right) \tag{5.28}$$

and describes the angle between the direction of motion of the follower and the direction of the axis of transmission. This implies that for $\alpha = 0°$ all transmitted force goes into motion of the follower, while for $\alpha = 90°$ all force is reduced to slip. In practice, most cams typically aim for α being smaller than $\pm 30°$ [62]. Table 5.1b indicates the maximum pressure angle of each cam during the rise phase. One can observe that for some cam profiles the value of α becomes large, explaining why the mechanism jammed for these designs at low voltages, as was shown in Fig. 5.8

Table 5.1: Minimum (relative) radius of curvature and maximum pressure angle being key design parameters of the cam-follower mechanism.

(a) minimum value of $|\rho_c|/r_0$

H \ β	$\frac{1}{2}\pi$	$\frac{3}{4}\pi$	π	$\frac{5}{4}\pi$	$\frac{3}{2}\pi$
0.01	1.98	2.50	2.50	2.50	1.98
0.03	1.84	2.50	2.50	2.50	1.84
0.05	1.12	2.50	2.50	2.50	1.12
0.07	0.86	2.43	2.50	2.43	0.86

(b) maximum value of α [°]

H \ β	$\frac{1}{2}\pi$	$\frac{3}{4}\pi$	π	$\frac{5}{4}\pi$	$\frac{3}{2}\pi$
0.01	21	14	11	9	7
0.03	40	31	24	20	17
0.05	49	39	32	27	23
0.07	54	45	38	32	28

References

[1] H.A. Rothbart. Cam Design Handbook. McGraw-Hill handbooks. McGraw-Hill, 2004.

[2] N. Nayak, P. A. Lakshminarayanan, M. K. Gajendra Babu, and A. D. Dani. Predictions of cam follower wear in diesel engines. Wear, 260(1-2):181–192, 2006.

[3] S. Sarıdemir and H. Saruhan. Experimental analysis of maximum valve lift effects in cam-follower system for internal combustion engines. Journal of Mechanical Science and Technology, 28(9):3443–3448, 2014.

[4] B. A. Paden, S. T. Snyder, B. E. Paden, and M. R. Ricci. Modeling and control of an electromagnetic variable valve actuation system. IEEE/ASME Transactions on Mechatronics, 20(6):2654–2665, 2015.

[5] T. Lenzi, M. Cempini, L. J. Hargrove, and T. A. Kuiken. Design, development, and validation of a lightweight nonbackdrivable robotic ankle prosthesis. IEEE/ASME Transactions on Mechatronics, 24(2):471–482, 2019.

[6] T. Ouyang, P. Wang, H. Huang, N. Zhang, and N. Chen. Mathematical modeling and optimization of cam mechanism in delivery system of an offset press. Mechanism and Machine Theory, 110:100–114, 2017.

[7] T. Xing, Y. Xu, and J. Ruan. Two-dimensional piston pump: Principle, design, and testing for aviation fuel pumps. Chinese Journal of Aeronautics, 2019.

[8] E. Ottaviano, D. Mundo, G. A. Danieli, and M. Ceccarelli. Numerical and experimental analysis of non-circular gears and cam-follower systems as function generators. Mechanism and machine theory, 43(8):996–1008, 2008.

[9] J. Xiang, Z. Cai, Y. Zhang, and W. Wang. A micro-cam actuated linear peristaltic pump for microfluidic applications. Sensors and Actuators A: Physical, 251:20–25, 2016.

[10] R. Alzate, M. Di Bernardo, U. Montanaro, and S. Santini. Experimental and numerical verification of bifurcations and chaos in cam-follower impacting systems. Nonlinear Dynamics, 50(3):409, 2007.

[11] T. K. Naskar and S. Acharyya. Measuring cam–follower performance. Mechanism and Machine Theory, 45(4):678–691, 2010.

[12] S. Sundar, J. T. Dreyer, and R. Singh. Estimation of impact damping parameters for a cam–follower system based on measurements and analytical model. Mechanical Systems and Signal Processing, 81:294–307, 2016.

[13] G. Osorio, M. di Bernardo, and S. Santini. Corner-impact bifurcations: a novel class of discontinuity-induced bifurcations in cam-follower systems. SIAM Journal on Applied Dynamical Systems, 7(1):18–38, 2008.

[14] P. Flores, R. Leine, and C. Glocker. Application of the nonsmooth dynamics approach to model and analysis of the contact-impact events in cam-follower systems. Nonlinear Dynamics, 69(4):2117–2133, 2012.

[15] H. R. Kim and W. R. Newcombe. The effect of cam profile errors and system flexibility on cam mechanism output. Mechanism and Machine Theory, 17(1):57–72, 1982.

[16] P. S. Grewal and W.R. Newcombe. A comparative study of cam motions for high-speed semi-rigid follower cam systems. Transactions of the Canadian Society for Mechanical Engineering, 12(3):121–128, 1988.

[17] S. T. Tümer and Y. S. Ünlüsoy. Nondimensional analysis of jump phenomenon in force-closed cam mechanisms. Mechanism and machine theory, 26(4):421–432, 1991.

[18] U. Chavan and S. Joshi. Synthesis of cam profile using classical splines and the effect of knot locations on the acceleration, jump, and interface force of cam follower system. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 225(12):3019–3030, 2011.

[19] F. W. Flocker. Addressing cam wear and follower jump in single-dwell cam-follower systems with an adjustable modified trapezoidal acceleration cam profile. Journal of engineering for gas turbines and power, 131(3), 2009.

[20] F. W. Flocker. A Versatile Cam Profile for Controlling Interface Force in Multiple-Dwell Cam-Follower Systems. Journal of Mechanical Design, 134(9), 2012.

[21] H. Abderazek, A. Yildiz, and S. Mirjalili. Comparison of recent optimization algorithms for design optimization of a cam-follower mechanism. Knowledge-Based Systems, 191:105237, 2020.

[22] H.-S. Yan, M.-C. Tsai, and M.H. Hsu. An experimental study of the effects of cam speeds on cam-follower systems. Mechanism and Machine Theory, 31(4):397–412, 1996.

[23] H.-S. Yan and W.-J. Tsai. A variable-speed approach for preventing cam-follower separation. Journal of Advanced Mechanical Design, Systems, and Manufacturing, 2(1):12–23, 2008.

[24] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi. Applications of machine learning to machine fault diagnosis: A review and roadmap. Mechanical Systems and Signal Processing, 138:106587, 2020.

[25] I. El-Thalji and E. Jantunen. A summary of fault modelling and predictive health monitoring of rolling element bearings. Mechanical systems and signal processing, 60:252–272, 2015.

[26] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic. Machine learning methods for wind turbine condition monitoring: A review. Renewable energy, 133:620–635, 2019.

[27] M. Cerrada, R. Sánchez, C. Li, F. Pacheco, D. Cabrera, J. V. de Oliveira, and R. E. Vásquez. A review on data-driven fault severity assessment

in rolling bearings. Mechanical Systems and Signal Processing, 99:169–196, 2018.

[28] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[29] R. Zhao, R. Yan, K. Chen, Z. Mao, P. Wang, and R. X. Gao. Deep learning and its applications to machine health monitoring. Mechanical Systems and Signal Processing, 115:213–237, 2019.

[30] O. Janssens, R. Van de Walle, M. Loccufier, and S. Van Hoecke. Deep learning for infrared thermal image based machine health monitoring. IEEE/ASME Transactions on Mechatronics, 23(1):151–159, 2017.

[31] W. Yu, l. Y. Kim, and C. Mechefske. Analysis of different rnn autoencoder variants for time series classification and machine prognostics. Mechanical Systems and Signal Processing, 149:107322, 2021.

[32] J.M. Zhou, L. Dong, W. Guan, and J. Yan. Impact load identification of nonlinear structures using deep recurrent neural network. Mechanical Systems and Signal Processing, 133:106292, 2019.

[33] M. Stender, M. Tiedemann, D. Spieler, D. Schoepflin, N. Hoffmann, and S. Oberst. Deep learning for brake squeal: Brake noise detection, characterization and prediction. Mechanical Systems and Signal Processing, 149:107181, 2021.

[34] N. Mohajerin and S. L. Waslander. Multistep prediction of dynamic systems with recurrent neural networks. IEEE Transactions on Neural Networks and Learning Sysems, 2019.

[35] S. K. Singh, R. Yang, A. Behjat, R. Rai, S. Chowdhury, and I. Matei. Pi-lstm: Physics-infused long short-term memory network. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pages 34–41. IEEE, 2019.

[36] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In Proceedings of the 2019 SIAM International Conference on Data Mining, pages 558–566. SIAM, 2019.

[37] R. Zhang, Y. Liu, and H. Sun. Physics-informed multi-lstm networks for metamodeling of nonlinear structures. arXiv preprint arXiv:2002.10253, 2020.

[38] A. Fisher, C. Rudin, and F. Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. Journal of Machine Learning Research, 20(177):1–81, 2019.

[39] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. arXiv preprint arXiv:1704.02685, 2017.

[40] M. Sundararajan, A. Taly, and Q. Yan. Gradients of counterfactuals. arXiv preprint arXiv:1611.02639, 2016.

[41] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. Plos One, 10(7):e0130140, 2015.

[42] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In Advances in neural information processing systems, pages 4765–4774, 2017.

[43] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016.

[44] L. S. Shapley. A value for n-person games. Contributions to the Theory of Games, 2(28):307–317, 1953.

[45] H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim. Towards a rigorous evaluation of xai methods on time series. In 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pages 4197–4201. IEEE, 2019.

[46] K. E. Mokhtari, B. P. Higdon, and A. Başar. Interpreting financial time series with shap values. In Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering, pages 166–172, 2019.

[47] M. Vega García and J. L. Aznarte. Shapley additive explanations for no2 forecasting. Ecological Informatics, 56:101039, 2020.

[48] P. J. Werbos. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10):1550–1560, 1990.

[49] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In International Conference on Machine Learning, pages 1310–1318, 2013.

[50] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. IEEE transactions on Neural Networks and Learning Systems, 28(10):2222–2232, 2016.

[51] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In IEEE International Conference on Robotics and Automation, pages 3223–3230. IEEE, 2015.

[52] W. De Groote, E. Kikken, E. Hostens, S. Van Hoecke, and G. Crevecoeur. Neural network augmented physics models for systems with partially unknown dynamics: Application to slider-crank mechanism. IEEE/ASME Transactions on Mechatronics, 2021.

[53] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[54] F. Chollet et al. Keras. `https://keras.io`, 2015.

[55] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symp. on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016.

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.

[57] W. Li, L. Zhu, Y. Shi, K. Guo, and Y. Zheng. User reviews: Sentiment analysis using lexicon integrated two-channel cnn-lstm family models. Applied Soft Computing, page 106435, 2020.

[58] E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. Knowledge and information systems, 41(3):647–665, 2014.

[59] S. Lipovetsky and M. Conklin. Analysis of regression in game theory approach. Applied Stochastic Models in Business and Industry, 17(4):319–330, 2001.

[60] A. Datta, S. Sen, and Y. Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In 2016 IEEE symposium on security and privacy (SP), pages 598–617. IEEE, 2016.

[61] H. Chen, S. Lundberg, and S. Lee. Explaining models by propagating shapley values of local components. arXiv preprint arXiv:1911.11888, 2019.

[62] R.L. Norton. Cam Design and Manufacturing Handbook. Industrial Press, 2009.

Chapter 6

Predicting Nonlinear System Dynamics Beyond Nominal Operations

The complementation of an incomplete physics model with a neural network (Chapter 4) and the addition of physical insights to a recurrent neural network model (Chapter 5) both exhibited enhanced prediction performances. Nevertheless, both frameworks were mainly tested for operating conditions close to the training dataset. In this chapter we study the possibility to develop hybrid modeling approaches of the cam-follower mechanism, as considered in Chapter 5, that provide reliable predictions for system configurations far beyond the settings included in the training set. My contributions can be summarized as follows:

- A hybrid modeling approach, encompassing both physics-inspired and neural layers, is developed to predict the cam-follower mechanism behavior for given design and control parameters. The prediction performances are extensively tested for varying amount of data included in the training set. This way, the exploration capabilities of the hybrid approaches were compared against the data-driven baseline.

- The ability to assess unseen system configurations is exploited to discover the parameter space. Consequently, starting from data of nominal operating conditions, I used the trained models to identify the set of critical system parameters for which hazardous detachment behavior occurs.

Physics-Based Neural Network Models for Prediction of Cam-Follower Dynamics Beyond Nominal Operations

W. De Groote, S. Van Hoecke, and G. Crevecoeur

Accepted in "IEEE/ASME Transactions on Mechatronics", 2021

Abstract *Cam-follower mechanisms are key in various mechatronic applications to convert rotary to linear reciprocating motions. The dynamic behavior of these systems relies on the design parameters such as the cam shape and follower mass. It appears that for some combinations of system parameters, continuous contact between the cam and follower cannot be assured, leading to harmful periodic impacts. This research presents a data-driven approach to predict the influence of parameter settings on the system dynamics by learning from a limited data set of nominal operating conditions. More specifically, we present a hybrid model architecture encompassing an ordinary differential equation, consisting of a close interconnection of neural and physics-based network layers. Due to an increased generalization established by the physical laws, these physics-based neural network models exhibit enhanced extrapolation capabilities compared to their black-box counterparts. Consequently, the presented models can accurately simulate the system behavior for parameter settings far beyond the nominal values included in the training data. This way, starting from a limited set of nominal time-series data, we could accurately estimate the set of critical system parameters that lead to hazardous jump phenomena in cam-follower systems.*

6.1 Introduction

Cam-follower mechanisms are mechanical subsystems that translate a rotational displacement of a drivetrain shaft into a reciprocating motion [1]. These mechanisms are typically implemented in combustion engines to regulate the cylinder intake and exhaust valves [2]. Mechatronic implementations of cam-follower mechanisms can be found in, among others, actuators [3] and robotics [4]. They can also be found in high-precision pumps with applications in avionics [5] and biomedical applications [6, 7].

The cam shape is typically designed so that it provides the desired displacement path to the follower while limiting the dynamics induced to the overall system [8]. By controlling the rotational speed of the camshaft, the motion dynamics can be further optimized [9]. It is generally assumed that the follower perfectly and continuously tracks the cam perimeter. However, for increased rotational speed, the follower can detach from the cam, resulting in hazardous bouncing behavior [10, 11]. This unwanted phenomenon can inflict damage to

the system due to the large periodic impacts caused by the follower jumps [12]. Although this behavior is desired for some dedicated machines, such as cutting tools [13], most system designs need to avoid this harmful behavior.

In this research, we endeavor to predict the occurrence of follower jumps for unseen system parameter settings. Recent advances in machine learning have shown the ability to discover complex relations in machine data, enabling a data-driven assessment of the system behavior [14]. However, these algorithms become typically less useful when labeled data of the failure events are scarce or not available at all. A possible approach to overcome this burden is to augment the data set with synthetic data obtained by emulating erroneous situations on high-fidelity physics models [15]. The construction of physics-inspired simulation models typically comprises the definition of a simplified model structure, followed by the identification of the introduced system parameters such as inertia and friction coefficients [16, 17]. Unfortunately, the ingrained system behavior of many mechatronic systems is often too complex, making it very challenging to deduce the physical relations of all interactions at play.

Alternatively, black-box system identification methods can learn the system dynamics directly from the measured time series [18]. In particular, deep learning methods have shown the ability to replace the traditional physics-inspired relations defined in state-space [19], Lagrangian [20] and Hamiltonian [21] representations. Although the high flexibility of these modeling formalisms enables enhanced predictive performances, they typically become unreliable when evaluated on regions for which they have not seen training data. Recent research on combining black-box models with physics-inspired methods showed promise in accommodating this burden. For instance, enhanced generalization can be obtained by enforcing physical consistency (i.e., conservation of energy) in the loss function [22]. Alternatively, the influence of the neural networks can be attenuated by using them as mappings that compensate for prediction discrepancies of simplified physics-based models [23, 24]. Furthermore, neural networks have been used to accommodate specific unknown interactions in incomplete yet accurate physics models [25, 26].

Inspired by the benefits of combining machine learning techniques with physics, we address the challenge of identifying the parameter settings for which unwanted behavior in cam-follower mechanisms occurs by defining the following twofold research objective. First, hybrid structured models, including physics-based and neural network layers, are designed to predict the cam-follower behavior for different design and control parameter settings. Although no complete physics model is available, we present two hybrid modeling architectures in which we gradually increase the amount of (partially) known physics to the network model. Consequently, by learning the system dynamics from a limited set of nominal time-series data, we obtain reliable

predictions for a wide range of unseen system parameter settings. Second, the enhanced generalization capabilities of the hybrid model architectures are exploited to estimate the occurrence of bouncing behavior for unseen design parameters and control settings, as shown in Fig. 6.1. This way, the model, trained on a limited amount of nominal system settings (i.e., gray dots), is deployed to identify the set of parameter settings (Q) that lead up to unwanted follower jumps.

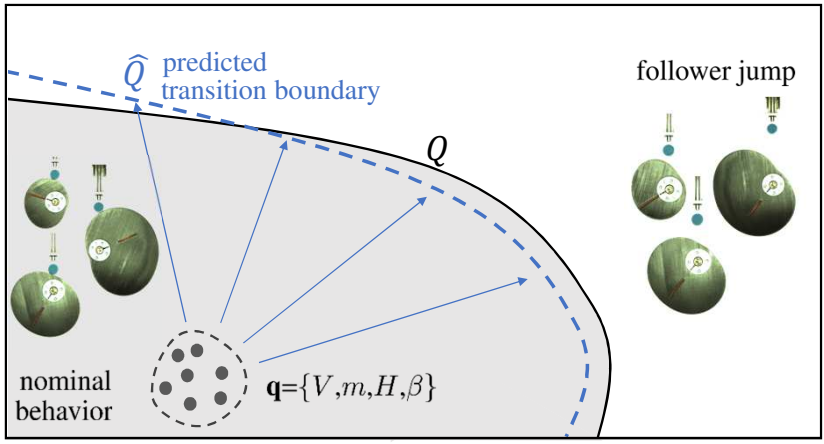


Figure 6.1: Schematic representation of the framework in which time-series data of nominal operating conditions, characterized by the system parameters $\mathbf{q}$, are used to train a model. This model is used to estimate the bifurcation boundary $\mathbf{q} \in Q$ at which the follower detaches from the cam.

6.2 Cam-Follower Mechanism

A cam-follower mechanism converts a rotary motion into a linear displacement. This system is characterized by a cam, which is a profiled shape mounted on a shaft. As the cam rotates, the follower is forced to track the cam perimeter leading into reciprocating motions. The conversion from rotational to linear movements induces various dynamics in both the follower mechanism and drivetrain shaft.

6.2.1 Dynamic Model of the Cam-Follower Mechanism

Follower Dynamics

The follower model assumes a theoretical knife-edge follower that is subject to a vertical motion path [1]. A model of the mechanism is derived using

Newtonian mechanics applied on the cam-follower diagram illustrated in Fig. 6.2. The dynamics of the follower position $y(t)$ are derived by considering the force equilibrium along the direction of the linear movement. The follower, characterized by mass m, is subject to an interface force $F(t)$ delivered by the rotating cam. Typically, a linear spring, characterized by k, is considered to reduce the possibility of having follower jumps. The cam is defined by its displacement function $h(\theta)$ that defines the position of the cam perimeter for given rotation angle θ. Consequently, the follower position $y(t)$ is physically restricted in space by $y(t) \geq h(\theta(t))$.

$$\begin{aligned} m\ddot{y}(t) + F_b(\dot{y}(t)) + ky(t) &= F(t) - mg \\ \text{s.t. } y(t) &\geq h(\theta(t)) \end{aligned} \tag{6.1}$$

For reasons of clarity, we omit the explicit notation of time dependency t, which is still implicitly assumed. The friction force F_b encompasses both a Coulomb and a viscous damping term:

$$F_b(\dot{y}) = b_0 \text{sign}(\dot{y}) + b_1 \dot{y} \tag{6.2}$$

The dynamics of cam displacement h relate to the dynamics of the rotating drivetrain and are derived by applying the chain rule of differentiation. For notational convenience, we define the rotational speed $\omega = \dot{\theta}$.

$$\begin{aligned} \dot{h} &= \frac{\mathrm{d}h}{\mathrm{d}\theta}\omega \\ \ddot{h} &= \frac{\mathrm{d}^2h}{\mathrm{d}\theta^2}\omega^2 + \frac{\mathrm{d}h}{\mathrm{d}\theta}\dot{\omega} \end{aligned} \tag{6.3}$$

Ideally, the follower follows the cam perimeter perfectly, modeled by perfect connection $y = h(\theta)$. In this case, we can combine (6.1) and (6.3) to model the interface force F by the relation $F^\dagger$, described by

$$F^\dagger = m(\frac{\mathrm{d}^2h}{\mathrm{d}\theta^2}\omega^2 + \frac{\mathrm{d}h}{\mathrm{d}\theta}\dot{\omega} + g) + F_b(\frac{\mathrm{d}h}{\mathrm{d}\theta}\omega) + kh \tag{6.4}$$

Negative acceleration tends to reduce the cam-follower interface force F, and if the acceleration is sufficiently large, surface contact between cam and follower can be lost. Hence, we note the interface force $F = 0$ if $y > h(\theta)$. Subsequently, the force relations $F \geq 0$ can be modeled by

$$F = \begin{cases} F^\dagger, & \text{if } y = h(\theta) \\ 0, & \text{if } y > h(\theta) \end{cases} \tag{6.5}$$

Drivetrain Dynamics

The rotational movement of the cam is typically driven by a shaft connected to a motor. The dynamics of the shaft describe the behavior of θ and ω and are, therefore, directly linked with the dynamics of the follower mechanism described in (6.1). The dynamics of the shaft, characterized by inertia J, can be described by the scheme presented in Fig. 6.2. The resulting torque that drives the cam-follower mechanism is denoted by T_d. This variable includes the motor torque, reduced by various counteracting phenomena such as bearing and gearbox imperfections and braking systems. The torque T results from the interactions with the cam-follower mechanism. Based on the law of conservation of energy, the relation between the interface force F and the torque T can be written as

$$T\mathrm{d}\theta = F\mathrm{d}h \tag{6.6}$$

Consequently, the dynamic torque equilibrium of the drivetrain shaft can be expressed as

$$J\dot{\omega} = T_d - F\frac{\mathrm{d}h}{\mathrm{d}\theta} \tag{6.7}$$

By substituting (6.5) in (6.7), we obtain following expression of the drivetrain dynamics:

$$\dot{\omega} = \begin{cases} \dfrac{T_d - T^\dagger}{J^\dagger} & \text{if } y = h(\theta) \\ \dfrac{T_d}{J} & \text{if } y > h(\theta) \end{cases} \tag{6.8}$$

This implies that we obtain an angle-dependent inertia $J^\dagger$ and an equivalent load $T^\dagger$ defined by:

$$\begin{aligned} J^\dagger &= J + m(\frac{\mathrm{d}h}{\mathrm{d}\theta})^2 \\ T^\dagger &= \frac{\mathrm{d}h}{\mathrm{d}\theta}\left(m\frac{\mathrm{d}^2h}{\mathrm{d}\theta^2}\omega^2 + mg + F_b(\frac{\mathrm{d}h}{\mathrm{d}\theta}\omega) + kh\right) \end{aligned} \tag{6.9}$$

6.2.2 Setup

The proposed methodologies are experimentally validated on the cam-follower setup depicted in Fig. 6.3. The mechanism is driven by a 60 W DC motor with an internal gear by controlling the input voltage V via a dSPACE 1104 control unit. The constant load of the mechanism m can be modified by mounting additional discs to the follower mechanism. There is no spring attached ($k = 0$) to invoke the bifurcation problem in which follower jumps occur. The friction, parameterized by $b_0 = 0.5$ and $b_1 = 12$ in (6.2), is mainly caused by the linear

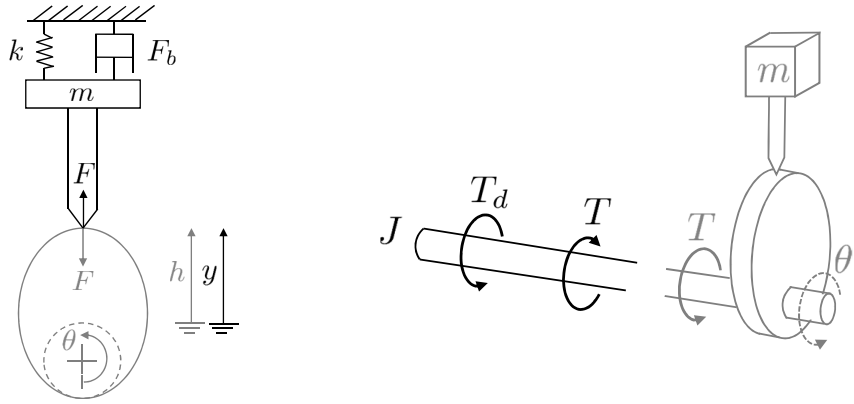


Figure 6.2: Representation of the cam-follower mechanism.

motion guide and is identified via a dedicated identification process. The motor is equipped with a high accuracy encoder (10 000 lines) that captures the angle θ at 2000 Hz. These precise measurements allow numerical differentiation to deduce the rotational speed ω and acceleration $\dot{\omega}$. Moreover, the follower is instrumented by a high-precision linear encoder with a resolution of 2 μm. This encoder provides accurate measurements of the follower position y (2000 Hz), required to determine the possible occurrence of follower jumps.

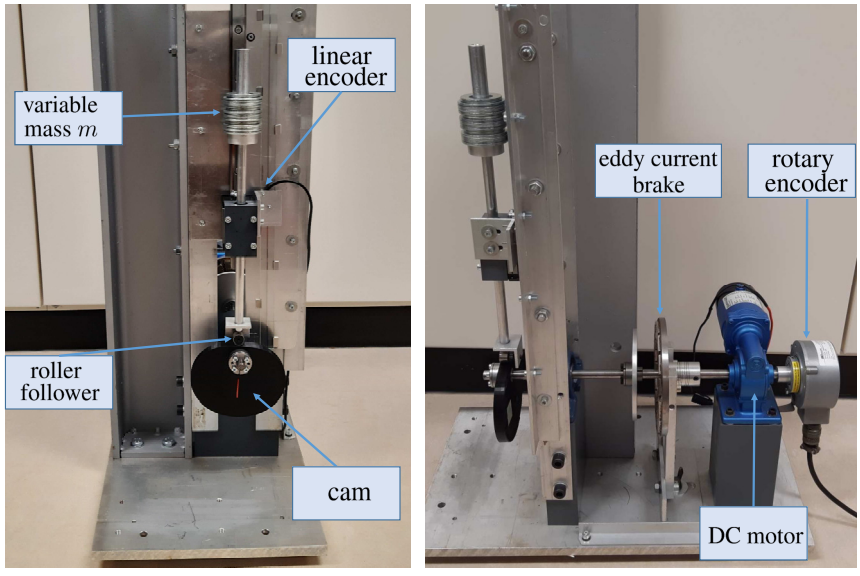


Figure 6.3: Cam-follower setup.

The modular setup is equipped with 20 interchangeable cams, as shown in Fig. 6.4. Each cam shape is uniquely defined by its displacement function $h(\theta)$. The cams studied during this research are characterized by a cycloidal

displacement function without dwell [1]. We can formulate this motion mathematically by the following analytical expression:

$$h(\theta) = \begin{cases} \frac{H\theta}{\beta} - \frac{H}{2\pi}\sin(2\pi\frac{\theta}{\beta}), & \text{if } \theta \in [0, \beta] \\ H\frac{(2\pi-\theta)}{(2\pi-\beta)} - \frac{H}{2\pi}\sin(2\pi\frac{(2\pi-\theta)}{(2\pi-\beta)}), & \text{if } \theta \in]\beta, 2\pi] \end{cases} \tag{6.10}$$

These types of cam profiles enable continuous derivatives $\frac{\mathrm{d}h}{\mathrm{d}\theta}$ and $\frac{\mathrm{d}^2h}{\mathrm{d}\theta^2}$, aiming for smooth dynamics of the overall drivetrain system. As shown in Figure 6.4, the cam profiles used for this research are characterized by two parameters: the maximum displacement H and the skewness angle β.

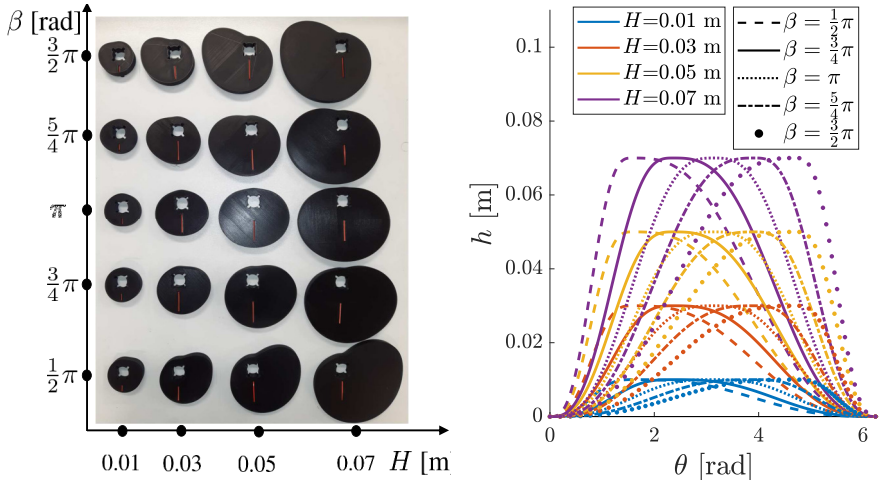


Figure 6.4: Cam shapes parameterized by β and H.

6.2.3 Design of Experiments

The setup is used to generate data for various machine settings, defined by $\mathbf{q} = [V, m, H, \beta]^T$. Each experiment begins from the same angular position ($\theta = 0$) from a standstill ($\omega = 0$). Figure 6.5 illustrates start-up trajectories for different parameters in $\mathbf{q}$. Furthermore, we introduce the variable $\mathcal{T}$ that resembles all trajectory measurements θ, ω and $\dot{\omega}$ associated with a single start-up phenomenon. The influence of the design parameters is examined by testing all 20 cams, characterized by β and H, for four different masses m. Furthermore, each system configuration is extensively tested by applying 20 different constant voltages V to the DC motor. Consequently, this measurement campaign spans a full factorial design of 1600 trajectory measurements $\mathcal{T}$. The resulting data are made open-source at `https://github.com/wannesdegroote/cam-follower-dataset` to allow replication of the research results.

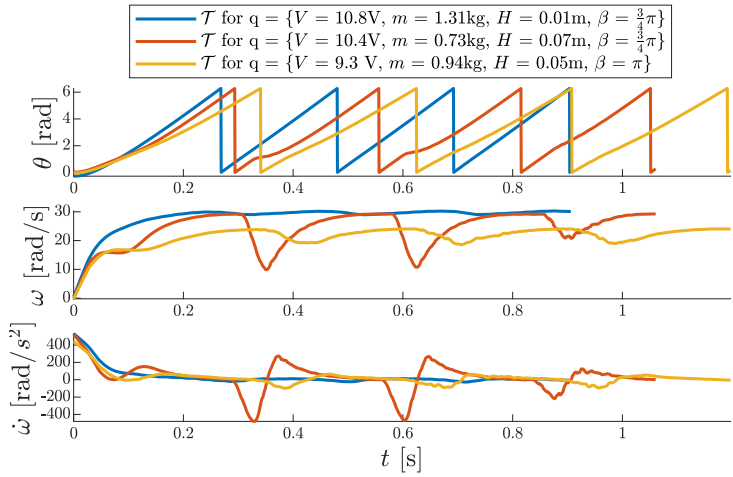


Figure 6.5: Example of different trajectory measurements $\mathcal{T}$ associated with different system settings in $\mathbf{q}$.

6.3 Characterization of the Parameter Influences

6.3.1 Detection of Follower Jumps

The different system variations, defined by $\mathbf{q}$, can either lead to nominal behavior or detachment between cam and follower, as shown in Fig. 6.6. To objectively assess the exploration capabilities of the proposed algorithms, we quantify the system behavior by accurately measuring the follower position y. Figure 6.7 shows an example of the observed displacements y of both nominal and bouncing behavior. The discrepancy ϵ between the follower position y and the cam displacement h clearly depicts the bouncing behavior of the detached follower.

We define ϵ_{max} as the maximum value of ϵ, averaged over each cam rotation. This maximum jump height ϵ_{max}, observed for each system parameter $\mathbf{q}$, serves as an objective metric to quantify the bifurcation phenomena occurring in cam-follower mechanisms. Considering the sensor noise and mechanical vibrations, we determined empirically $\frac{\epsilon_{\text{max}}}{H} > 1.3\%$ to define a follower jump. Figure 6.8 illustrates the observed bifurcation problem for the different cam shapes and masses related to the system parameters in $\mathbf{q}$. The black line indicates the observed set of parameter combinations in $Q \subset \mathbb{R}^4$ that defines the transition between nominal and bouncing behavior. Note that the measurements y are only exploited to determine the ground truth of Q and are thus not used as training data in the proposed exploration algorithms.

(a) (b)

Figure 6.6: Cam-follower mechanism. (a) nominal behavior. (b) follower jump.

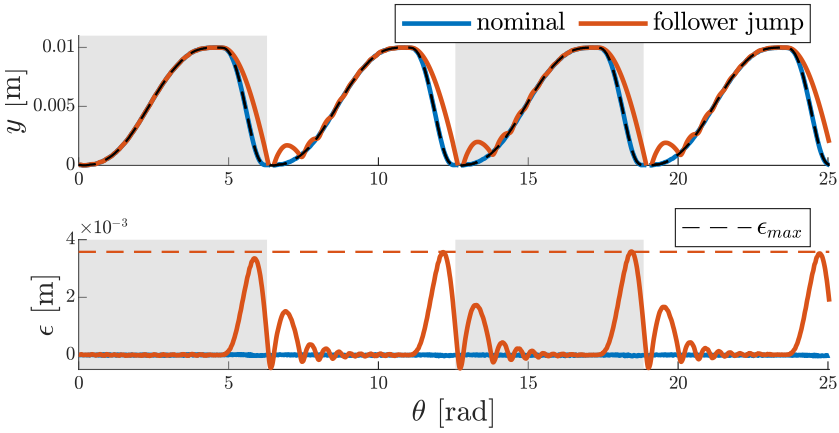


Figure 6.7: Measured follower position y and the related discrepancy ϵ with the cam displacement h.

6.3.2 Minimal Contact Force

The experiments for which bouncing behavior is observed are characterized by a contact force F that becomes zero when follower jumps occur. By analogy, the expression for the interface force $F^{\dagger}$ in (6.4), which is, in essence, only valid when contact between cam and follower is assured, should tend towards zero when the detachment initiates. Therefore, we define $F^{\dagger}_{\text{min}}$ as the minimum value of $F^{\dagger}$ obtained during a trajectory sequence:

$$F^{\dagger}_{\text{min}} = \min\{F^{\dagger}_{k} : k = 0 \dots N\} \tag{6.11}$$

If the minimal value $F^{\dagger}_{\text{min}}$ is larger than zero, the system is assigned to be nominal because this assumes that a positive contact force F is assured along the entire cam revolution. Negative values of $F^{\dagger}_{\text{min}}$ imply unfeasible solutions,

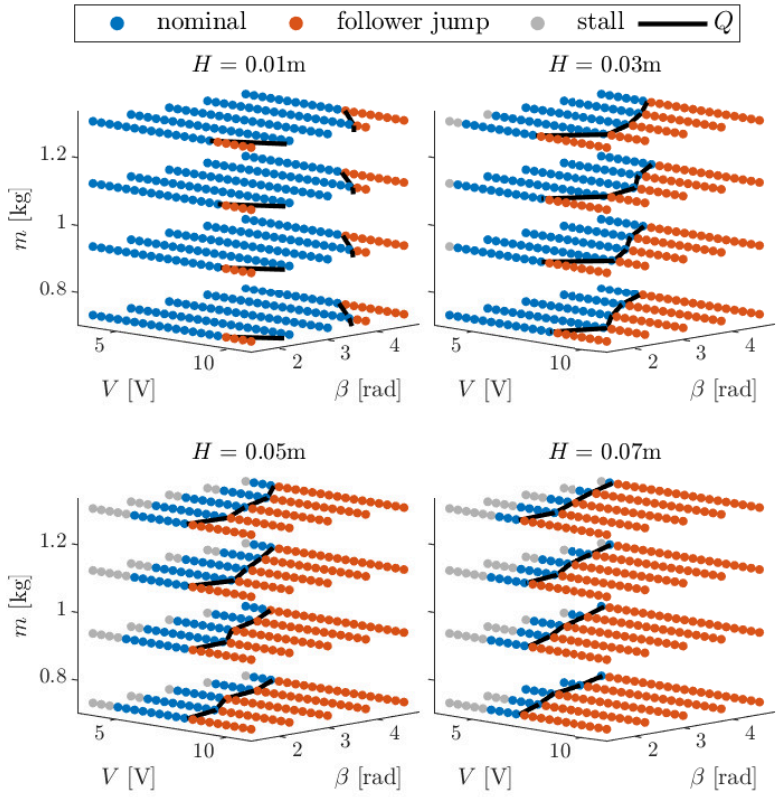


Figure 6.8: The ground-truth of the bifurcation phenomenon related to the system parameters in $\mathbf{q}$, determined via accurate measurements of the follower discrepancy ϵ.

which are indicative of an experiment in which a follower jump initiated at $F^{\dagger} = 0$. Moreover, because the value of $F^{\dagger}_{\text{min}}$ has only a physical interpretation when contact is assured, we introduce the minimal contact force F_{min} as:

$$F_{\text{min}} = \max(F^{\dagger}_{\text{min}}, 0) \tag{6.12}$$

By calculating this value for a given trajectory $\mathcal{T}$, associated to certain operating conditions in $\mathbf{q}$, we can classify the system as nominal if $F_{\text{min}} > 0$ and assign bouncing behavior if $F_{\text{min}} = 0$. The derived values of $F^{\dagger}$ corresponding to the signals shown in Fig. 6.7 are illustrated in Fig. 6.9. Consequently, we can observe that by employing (6.11) and (6.12) on the values of $F^{\dagger}$, we can distinguish between signals of nominal behavior and signals that are indicative for the occurrence of follower jumps.

The derived values of the minimal contact force F_{min} for all system parameters $\mathbf{q}$ are summarized in Fig. 6.10. The black lines indicate the true bifurcation boundary Q, as discussed in Fig. 6.8. These results indicate that the derived minimal interface force F_{min} is a valid metric that can be used to

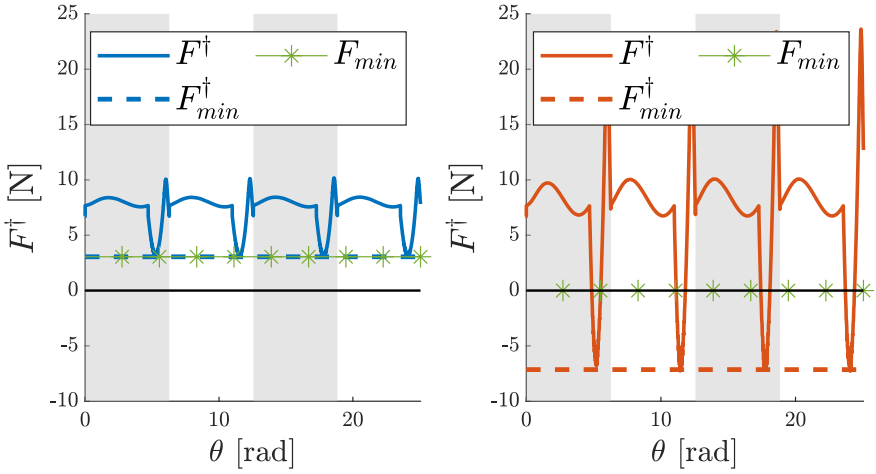


Figure 6.9: Calculated values of the interface force metric $F^\dagger$ based on measured trajectory data $\mathcal{T}$. The left plot correctly assigns nominal behavior ($F_{\text{min}} > 0$) while the right plot is properly identified as an experiment in which follower jumps occurred ($F_{\text{min}} = 0$).

classify the follower behavior, based on given system parameters $\mathbf{q}$ and the related trajectory measurements $\mathcal{T}$.

6.3.3 Ad Hoc Estimation of the Transition Boundary Q

A major advantage arises when the system parameters $\mathbf{q}$ for which undesired behavior occurs are known during the design of mechatronic applications. More specifically, knowledge about the transition boundary Q can help to determine the critical design values. The identification of Q becomes trivial if an extensive data set, such as demonstrated in Fig. 6.10, is available. For sparse datasets, advanced machine learning techniques can be used to solve this binary classification problem [14]. However, in majority of cases, only nominal data are available and observations of undesired behavior are missing in the dataset. Binary classifier techniques are not useful for these situations. Therefore, a hands-on approach to identify Q implies a direct extrapolation of the available values of F_{min}. Hence, the transition boundary Q can be estimated by extrapolating towards the values of $\mathbf{q}$ for which $F_{\text{min}} = 0$ holds. Figure 6.11 illustrates the accuracy of the predicted set Q for direct interpolation of, respectively, 10% and 90 % of the nominal data set. This result clearly illustrates the need for sufficient data close to the transition boundary Q. In practice, data are typically only available for a limited number of highly reliable nominal oper-

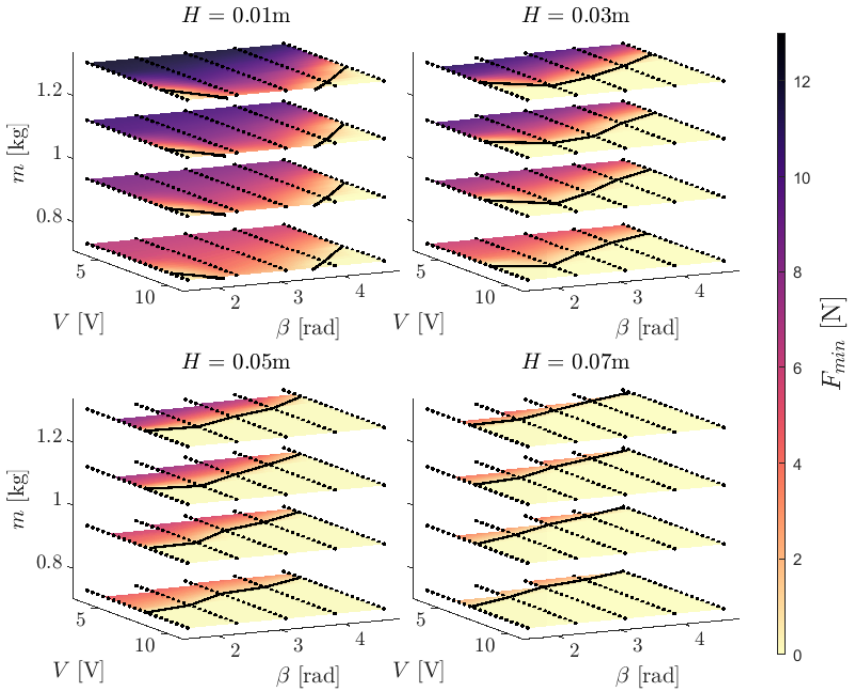


Figure 6.10: Approximation of the bifurcation phenomena related to the system parameters in $\mathbf{q}$, based on the calculated minimal contact force F_{min}.

ating conditions, indicating the need for more advanced techniques to identify the transition boundary Q.

6.4 Methodology

6.4.1 Estimation of the Transition Boundary Q

The exploration of the parameter space $\mathbf{q}$, starting from nominal operating points, fails if the data are not properly distributed along transition boundary Q. Therefore, we propose an indirect assessment of the parameters in $\mathbf{q}$ by analyzing their influence on the system dynamics. A two-step approach is depicted in Fig. 6.12. First, the influence of the system parameters $\mathbf{q}$ is learned by a data-driven model $\mathcal{M}$ from a limited set of nominal time-series data. Next to a black-box model serving as a baseline model, we will present two hybrid approaches combining physics-inspired and black-box components. All the models will be trained to simulate the system behavior for given parameter settings $\mathbf{q}$. By comparing the performances of the presented models, we can determine whether providing physics-inspired information to the model archi-

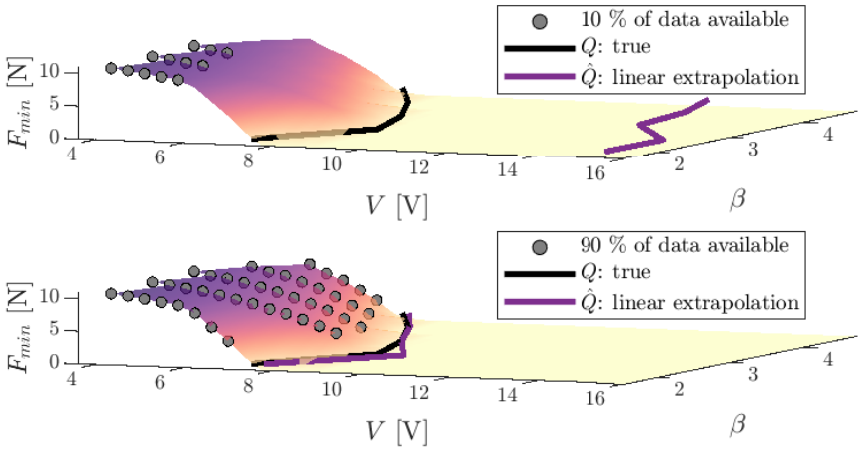


Figure 6.11: An example of the estimation of Q based on extrapolating a limited set of nominal observations. The observed surface considers a slice of the 4-dimensional input space $\mathbf{q}$, defined by choosing $H = 0.03$ m $m = 1.3$ kg. The results obtained by this ad hoc approach serve as reference in further discussions.

tecture improves the predictive capabilities. Second, once the model is trained on the nominal data set, it can be used to simulate the system behavior $\hat{\mathcal{T}}$ for unseen parameter settings $\mathbf{q}$. Because all trajectory segments begin from a standstill, the initial state $\mathbf{x}_0$ should be chosen equal to zero. Thereafter, the simulated trajectory $\hat{\mathcal{T}}$ can be used to calculate the minimal contact force F_{min}. This way, we can estimate the minimal contact force F_{min} for an extensive grid of parameter points $\mathbf{q}$, leading to a full exploration of the parameter space. Subsequently, the explored grid can be used to estimate the transition boundary Q by identifying the parameter values $\mathbf{q}$ for which $F_{\text{min}} = 0$ holds.

6.4.2 Learning the System Dynamics

The exploration of the parameter space highly relies on the ability to capture the parameter dependency $\mathbf{q}$ within the dynamic model $\mathcal{M}$. The behavior of a dynamic system is typically described by a state-space representation that defines the influence of an external control input $\mathbf{u}$ on the behavior of the system state $\mathbf{x}$:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}; \mathbf{q}) \tag{6.13}$$

For most systems, the system parameters $\mathbf{q}$ (e.g., masses, geometrical properties ....) remain constant. However, in this research, we would like to train a model that is sensitive to system changes by defining $\mathbf{q}$ as an explicit input to the model. The system state of the cam-follower mechanism is defined as

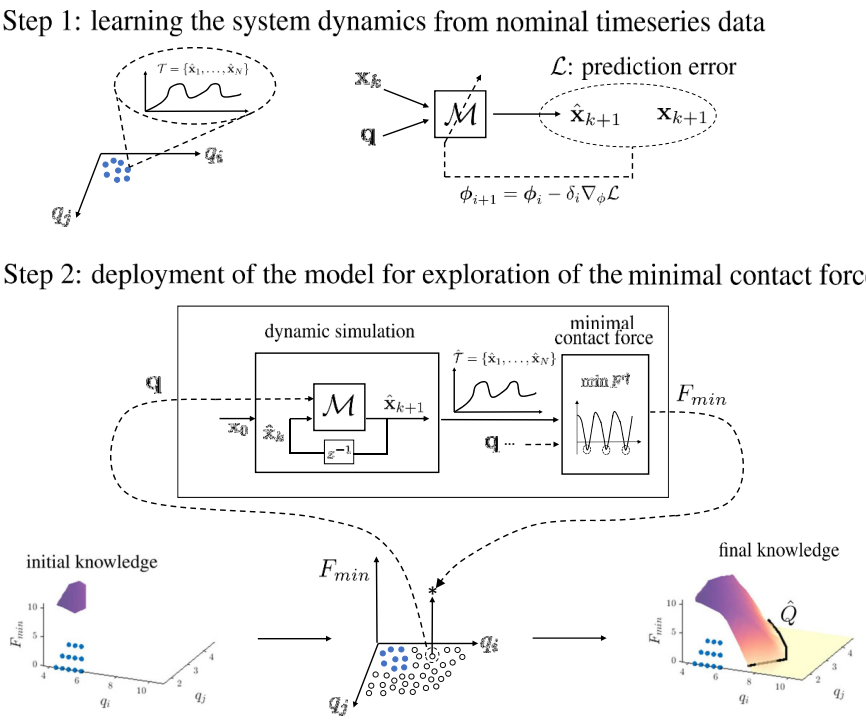


Figure 6.12: Overview of the two-step approach used to explore the influence of the system parameters in $\mathbf{q}$ on the behavior of cam-follower mechanisms.

$\mathbf{x} = \begin{bmatrix}\theta & \omega\end{bmatrix}^T$. The cam-follower mechanism studied in this research is not actively controlled by a time-varying control input $\mathbf{u}$. Note that the voltage V applied to the motor remains constant and, therefore, is considered as a system property absorbed in $\mathbf{q}$. Hence, the state-space representation of the cam-follower system can be simplified to:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}; \mathbf{q}) = \begin{bmatrix} \omega \\ f_\omega(\mathbf{x}, \mathbf{q}) \end{bmatrix} \tag{6.14}$$

Because the relation $\dot{\theta} = \omega$ is determined by definition, the modeling of $\mathbf{f}$ only requires the identification of the ordinary differential equation (ODE) captured in f_ω. If we impose $f_\omega(.|\phi)$ as a mathematical relation defined by its model parameters ϕ, we obtain an overall system model $\mathbf{f}(.|\phi)$. Consequently, we can use this expression to numerically integrate the state behavior $\mathbf{x}$ over a specified time interval. Although various advanced solvers exist [27], we found that a basic forward Euler integration suffices for this problem. Hence, for a given state measurement $\mathbf{x}_k$ at time instance k, we can estimate the subsequent

state as:

$$\hat{\mathbf{x}}_{k+1} = \underbrace{\mathbf{x}_k + \Delta t \cdot \mathbf{f}(\mathbf{x}_k; \mathbf{q}|\boldsymbol{\phi})}_{\mathcal{M}(\mathbf{x}_k, \mathbf{q}|\boldsymbol{\phi})} \tag{6.15}$$

The parameter Δt is determined by the (fixed) sampling rate of the state measurements. Furthermore, we can encapsulate this expression in $\mathcal{M}(.|\boldsymbol{\phi})$, being the one-step ahead prediction model. The identification of $\mathcal{M}(.|\boldsymbol{\phi})$, and thus the system dynamics model $\mathbf{f}(.|\boldsymbol{\phi})$, requires the optimization of the model parameters $\boldsymbol{\phi}$ based on the measured trajectory data. We define the loss function $\mathcal{L}(\boldsymbol{\phi})$ that quantifies the mean squared error (MSE) between the predictions obtained by the model $\mathcal{M}(.|\boldsymbol{\phi})$ and the measured states, over all $N_{\mathcal{I}}$ trajectories in the training set $\mathcal{I}_{\text{train}}$:

$$\mathcal{L}(\boldsymbol{\phi}) = \frac{1}{N_{\mathcal{I}}} \sum_{i=1}^{N_{\mathcal{I}}} \frac{1}{N_i} \sum_{k=1}^{N_i} (\hat{\mathbf{x}}_k^{(i)} - \mathbf{x}_k^{(i)})^T \mathrm{D} (\hat{\mathbf{x}}_k^{(i)} - \mathbf{x}_k^{(i)}) \tag{6.16}$$

The diagonal matrix D in this cost function typically copes with magnitude differences between the state variables. In this case, we define $\mathrm{D} = \mathrm{diag}(0, 1)$ because we only need to penalize the predictions of ω to identify the relation f_w in (6.14). Further, the optimal model parameters in $\boldsymbol{\phi}$ are defined by

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}) \tag{6.17}$$

Once the optimal model parameters $\boldsymbol{\phi}^*$ are determined, we can use the model $\mathcal{M}$ to simulate the system state $\mathbf{x}$ by sequentially feeding the prediction of the prior iteration back as input to the model, as shown in Fig. 6.13.

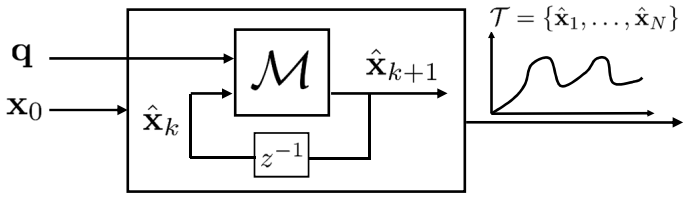


Figure 6.13: Representation of the model $\mathcal{M}$ used to simulate the system state $\mathbf{x}$ for given system setting $\mathbf{q}$.

It becomes obvious that the accuracy of the dynamics captured in $\mathbf{f}$ is key to obtaining reliable system simulations for unseen system parameters $\mathbf{q}$. For many simple mechatronic systems, the relation of $\mathbf{f}$ is being approximated using known physical laws (e.g., Newton's law). However, in practice, a full model of intricate systems is rarely available because these systems exhibit various unknown phenomena that are challenging to capture in the physics-model. Alternatively, the expression for $\mathbf{f}$ can be approximated by considering

a data-driven relation, learned by fitting the available data to the model. However, these black-box models often exhibit poor generalization capabilities if they are learned from sparse data sets. Therefore, in this research, we propose to model $\mathbf{f}$ in a hybrid manner by including partially known physics into these data-driven models.

6.4.3 Black-Box Universal Approximator (ReLU)

First, we consider a traditional black-box model to capture the relation f_ω in (6.14). This model serves as a baseline model to assess the influence of the provided physics in the proposed hybrid approaches. The ability to capture the relations in the data relies on the complexity of the chosen model. Prior research has shown that neural networks are suited to capture the derivative function $\mathbf{f}$ directly from time-series data [19]. Moreover, it has been proven that feedforward neural networks with one hidden layer can approximate any continuous function for inputs within a specific range [28]. Therefore, we define a one-hidden-layer rectified linear unit (ReLU) network η of n_h hidden units. The architecture of this network, which maps an n_i-dimensional input $\mathbf{d}$ to a one-dimensional output, is illustrated in Fig. 6.14. The input $\mathbf{d}$ is scaled by the standard deviation (σ_j, $j = 1, \cdots, n_i$) of the corresponding input elements to eliminate magnitude differences and consequently avoid dominance of particular dimensions. The corresponding transformation matrix $\mathrm{S} = \mathrm{diag}\left(\sigma_1^{-1}, \ldots, \sigma_{n_i}^{-1}\right)$ is determined in advance directly from the measurement data. Subsequently, the input undergoes a linear transformation by the weights $\mathrm{W}_h \in \mathbb{R}^{n_i \times n_h}$ of the hidden layer. The hidden unit activation function Υ will, after adding the bias vector $\mathbf{b}_h \in \mathbb{R}^{n_h}$, include the required nonlinearity in the function η. Subsequently, the output is obtained via a linear output layer, defined by the weights $\mathrm{W}_o \in \mathbb{R}^{n_h \times 1}$ and $b_o \in \mathbb{R}$. We denote the ReLU model as being

$$\begin{aligned} \eta(\mathbf{d}|\boldsymbol{\phi}) &= \mathrm{W}_o^T \Upsilon\left(\mathrm{W}_h^T \mathrm{S}\mathbf{d} + \mathbf{b}_h\right) + b_o \\ \Upsilon(\cdot) &= \max\left(0, \cdot\right) \end{aligned} \tag{6.18}$$

The weights W and biases $\mathbf{b}$ in η represent the trainable parameters included in the optimization variable $\boldsymbol{\phi}$. Although other nonlinear activation functions Υ exist, a ReLU model is chosen due to its demonstrated usefulness in various regression tasks [29, 30]. Furthermore, it is common practice to map the angle θ by its goniometric properties to constrain the input space of the neural network. Hence, the input of η becomes

$$\mathbf{d}_{\mathrm{BB}} = \begin{bmatrix} \sin(\theta) & \cos(\theta) & \omega & V & m & H & \beta \end{bmatrix}^T,$$

as shown in the general overview of the black-box dynamic model $\mathbf{f}_{\mathrm{BB}}$ in Fig. 6.15a.

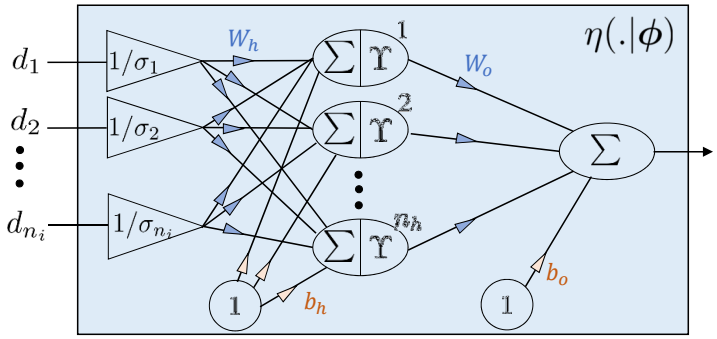


Figure 6.14: One-hidden-layer ReLU neural network η.

6.4.4 Physics-Based Neural Network Model

Traditional neural network models $\mathbf{f}_{\mathrm{BB}}$ can easily capture complex relations from the data due to their flexible model architecture. Nevertheless, a major drawback of these data-driven models is that unreliable predictions can be obtained if they are evaluated outside the region for which they were trained. In contrast, physics-inspired models can better predict the general trend if used for extrapolation purposes. Because a complete physics model of the overall mechatronic system is seldom available, we propose a hybrid approach in which we include the partially known physics into the network structure. Below, we present two hybrid architectures $\mathbf{f}_{\mathrm{H1}}$ and $\mathbf{f}_{\mathrm{H2}}$ in which we gradually increase the inclusion of physical laws of the cam-follower mechanism.

Soft-Hybrid $\mathrm{f}_{\mathbf{H1}}$

A first approach only considers the incorporation of the known cam characteristics into the overall model, as shown in Fig. 6.15b. Therefore, the raw measurements of θ and the cam properties H and β are replaced by $\dfrac{\mathrm{d}h}{\mathrm{d}\theta}$ and $\dfrac{\mathrm{d}^2h}{\mathrm{d}\theta^2}$ to provide more information to the neural network η. These relations are analytically deduced from the displacement function $h(\theta)$ in (6.10). Hence, the input of the ReLU network η becomes

$$\mathbf{d}_{\mathrm{H1}} = \begin{bmatrix} \dfrac{\mathrm{d}h}{\mathrm{d}\theta} & \dfrac{\mathrm{d}^2h}{\mathrm{d}\theta^2} & \omega & V & m \end{bmatrix}^T .$$

Note that the displacement $h(\theta)$ is not explicitly given to the network because the absence of a compression spring (i.e., $k = 0$) makes the information in h meaningless according to (6.9).

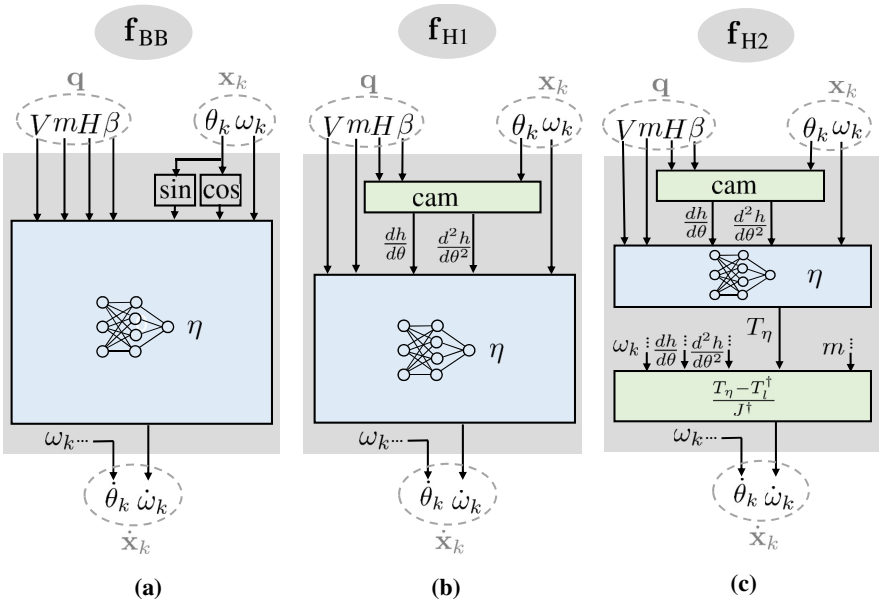


Figure 6.15: Schematic overview of the function $\mathbf{f}$ that captures the system dynamics. The modeling formalisms $\mathbf{f}_{\mathrm{BB}}$, $\mathbf{f}_{\mathrm{H1}}$ and $\mathbf{f}_{\mathrm{H2}}$ differ by the number of physics-inspired insights included in the model architecture.(a) Black-box. (b) Soft-hybrid. (c) Hybrid.

Hybrid $\mathbf{f}_{\mathrm{H2}}$

The number of physics-inspired insights incorporated within the model can be increased by enforcing the partially known ODE relation directly into the model, as shown in Fig. 6.15c. For this case, we included the expressions for $J^\dagger$ and $T^\dagger$, as defined in (6.9), as an additional physics-inspired network layer. Unknown parameters such as the shaft inertia J in $J^\dagger$ are implemented as trainable variables in ϕ that are simultaneously optimized with the variables in η. The output of the neural network η is used as a substitute for the unknown expression of T_d in (6.8). In addition to identifying the influence of $\mathbf{x}$ and $\mathbf{q}$ on T_d, the model η also absorbs the modeling discrepancies in $J^\dagger$ and $T^\dagger$. By choosing $\mathbf{d}_{\mathrm{H2}}$ equal to $\mathbf{d}_{\mathrm{H1}}$, sufficient information is given to η. Note that the network η is not trained in advance because no measurements of T_d are available. The network is trained by propagating the errors of the state predictions through both physics-inspired and neural layers. By adding these physics-inspired layers, we aim to reduce the influence of the neural network η, leading to a more reliable model $\mathbf{f}_{\mathrm{H2}}$.

6.4.5 Practical Implementation

As shown in (6.15), a discrete prediction model $\mathcal{M}$ can easily be deduced for each architecture of $\mathbf{f}$. The dataset is resampled from 2000 Hz to 100 Hz (i.e., Δt = 0.01) to make the evaluations of $\mathcal{M}$ more significant. Only data from nominal conditions are used as training data. The modeling architectures $\mathcal{M}$ are implemented as sequences of both physics-inspired and neural layers by using the Keras API [31] with the TensorFlow [32] backend in Python. The TensorFlow library employs automatic differentiation so that the analytical gradients can be extracted for both the ReLU network and customized physics layers. This enables an analytical expression of the gradient $\nabla_\phi \mathcal{L}$ of the cost function $\mathcal{L}$, defined in (6.16). This gradient information can be used to update the model parameters in $\boldsymbol{\phi}$ based on a gradient descent approach with step size δ.

$$\boldsymbol{\phi}_{i+1} = \boldsymbol{\phi}_i - \delta_i \nabla_\phi \mathcal{L} \tag{6.19}$$

In practice, an Adam optimizer is used that processes the data in mini-batches of 200 samples [33].

The number of neurons n_h in the hidden layer of the ReLU model η determines the model complexity. Consequently, the value of n_h is determined by performing a hyperparameter tuning step before the actual regression task. Figure 6.16 illustrates the influence of n_h on the accuracy of the trajectory simulations. This way, the number of neurons n_h in the hidden layer of η are chosen 128, 32 and 32 for the model architectures $\mathbf{f}_{\text{BB}}$, $\mathbf{f}_{\text{H1}}$ and $\mathbf{f}_{\text{H2}}$, respectively. In addition, we show the prediction results of using a linear black-box model determined by a linear least squares (LLS) regression. The observed failure of the linear model justifies the choice to exploit the nonlinear characteristics of the chosen ReLU model architecture.

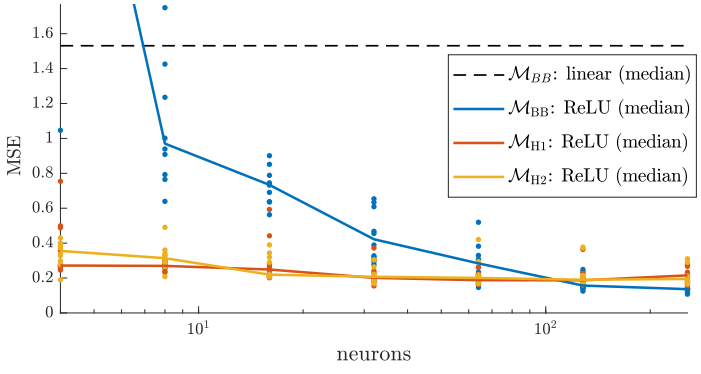


Figure 6.16: Tuning of the number of neurons n_h in the hidden layer of η for the different model topologies. The influence of each value n_h is tested 10 times by training on 30% of the nominal trajectories and validating on the remaining trajectories excluded from the training set.

6.5 Results and Discussion

The ability to explore the parameter space of $\mathbf{q}$ highly relies on the accuracy of the simulations $\hat{\mathcal{T}}$ generated by the learned model $\mathcal{M}$. Consequently, emphasis lies on analyzing the performance of the hybrid model architectures (i.e., $\mathbf{f}_{\text{H1}}$ and $\mathbf{f}_{\text{H2}}$) compared to their black-box counterpart (i.e., $\mathbf{f}_{\text{BB}}$). As explained before in Fig. 6.12, we are interested in learning the dynamics from a limited set of nominal trajectories to assess the system behavior for unseen system settings $\mathbf{q}$. Consequently, we only consider the 754 trajectories for which contact was assured, indicated by the blue dots in Fig. 6.8. The prediction experiments are always validated on samples from a test set $\mathcal{I}_{\text{test}}$, which were excluded from the set $\mathcal{I}_{\text{train}}$ used to train the model. In addition, to make a fair comparison for the exploration task, we sorted the experiments in ten different subsets $\{\mathcal{I}_1, ..., \mathcal{I}_{10}\}$ sorted according to their tendency to detach (i.e., the value of F_{min}). This way, we could easily assess both the interpolation and extrapolation performances, as illustrated in Fig. 6.17.

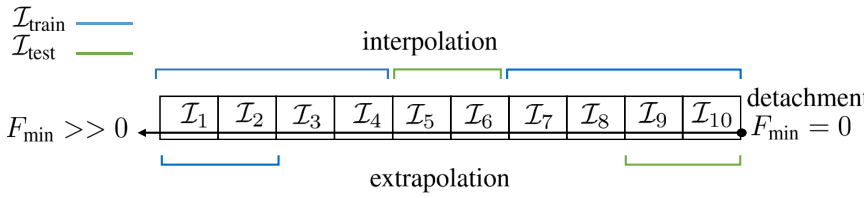


Figure 6.17: Schematic representation of interpolation and extrapolation on the sorted nominal data set.

6.5.1 Simulation Accuracy: Interpolation

In the first stage, we wish to assess the ability to learn the system dynamics from a data set that covers the system parameter space well. Therefore, a 5-fold interpolation experiment is performed in which we subsequently exclude another subset $\mathcal{I}_{\text{test}} = \{\mathcal{I}_j, \mathcal{I}_{j+1}\}$ for $j \in \{1, 3, 5, 7, 9\}$ from the training set. Figure 6.18 shows the simulation results of $\hat{\omega}$ and the corresponding MSE of some signals included in the test set $\mathcal{I}_{\text{test}}$. The left plots indicate the simulation results on a temporal scale. Dynamic models are typically plagued by drift because a small model error during a specific time instance will be propagated during further time instances. Therefore, the forecast will always deviate more from the actual measurements when the prediction horizon increases. To make a fair comparison of the prediction accuracy, each sample of the predicted trajectory $\hat{\mathcal{T}}$ is projected on the corresponding angle θ. This clearly shows that the MSE calculated on the angular reference provides a more fair metric to assess the prediction accuracy. Therefore, further references to the MSE

will implicitly assume a projection to an angular scale. The accuracies of the simulation trajectories $\hat{\mathcal{T}}$ in all test sets of this interpolation experiment are summarized in Fig. 6.19. These results illustrate that all modeling formalisms $\mathcal{M}$ achieve approximately the same prediction performances when sufficient data are included in the training data set $\mathcal{I}_{\text{train}}$.

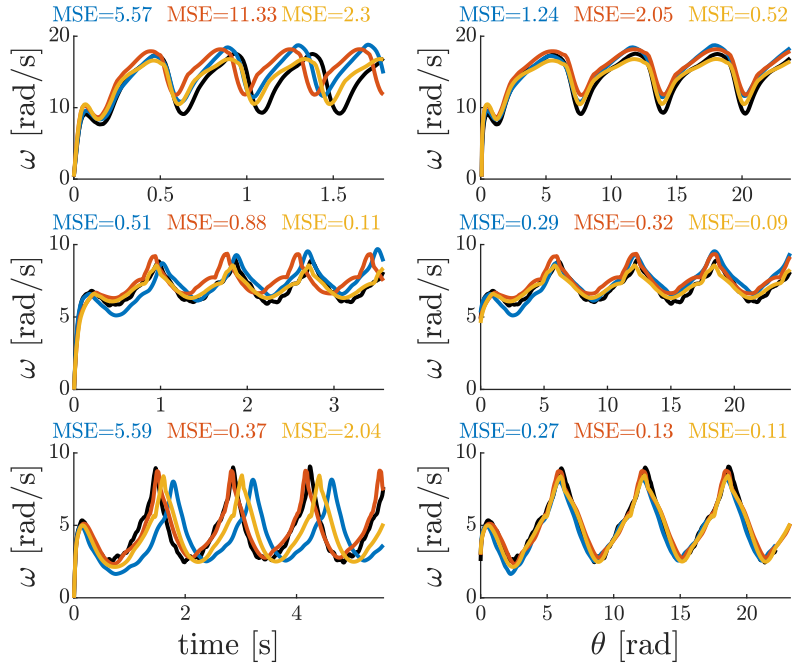


Figure 6.18: Left: Illustration of the drift that occurs when simulating trajectories $\hat{\mathcal{T}}$. Right: projection on the angle θ to have a more fair metric of the simulation accuracy.

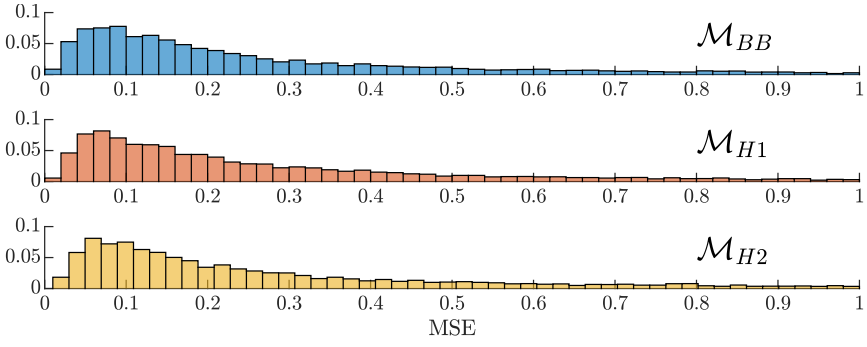


Figure 6.19: Histogram summarizing the prediction accuracy of the simulation trajectories $\hat{\mathcal{T}}$ obtained by a 5-fold validation experiment.

6.5.2 Simulation Accuracy: Extrapolation

The results revealed the ability to learn the system dynamics of nominal behavior for unseen parameter settings $\mathbf{q}$. However, the models were always evaluated for parameter settings $\mathbf{q}$ close to the data included in the training data set. In practice, data are often only available for a small range of operating conditions. Therefore, it is more interesting to assess the extrapolation capabilities, as was illustrated in Fig. 6.17. This evaluation implies that we assess the prediction performances for operating conditions $\mathbf{q}$ close to the transition boundary Q by using models that only have seen highly stable behavior during their training process. Figure 6.20 illustrates the simulation accuracy on the points that are most near to the bifurcation boundary Q for different amounts of data incorporated in the (sorted) training data set. The results clearly indicate that the (soft) hybrid models (i.e., $\mathcal{M}_{\mathrm{H1}}$ and $\mathcal{M}_{\mathrm{H2}}$) outperform the purely black-box model ($\mathcal{M}_{\mathrm{BB}}$). Moreover, the hybrid model $\mathcal{M}_{\mathrm{H1}}$ performs better than the soft-hybrid model $\mathcal{M}_{\mathrm{H2}}$ if limited training data are available. This outcome empirically demonstrates that the physics laws ingrained in the dynamic models provide better generalization and lead to more robust predictions outside the region for which the models are trained. One can also see that the models exhibit about the same accuracy once the training data set includes samples close to the test data set, showing that the benefits of including physics become less pronounced once sufficient data are available.

6.5.3 Contact Force Prediction: Extrapolation

The estimation of the contact force $F_{\min}$ for unseen parameters in $\mathbf{q}$ relies on the prediction accuracy of the simulation $\hat{\mathcal{T}}$, used to calculate $F^{\dagger}$. Figure 6.21 compares the predicted values of $F^{\dagger}_{\min}$ against the values calculated directly from the measurements. The dynamic models $\mathcal{M}$ are trained on the 10% most stable points. The lower the value of $F^{\dagger}_{\min}$, the further the operating conditions are situated from the conditions included in the training data set. The values calculated by the trajectories generated by the black-box model $\mathcal{M}_{\mathrm{BB}}$ are clearly less resilient to be evaluated outside the training region. This was expected because Fig. 6.20 already illustrated that the corresponding trajectory simulations $\hat{\mathcal{T}}$ situated near to Q are of low quality. Note that for predictions $F^{\dagger}_{\min} < 0$, we obtain $F_{\min} = 0$ according to (6.12), implying that bouncing behavior is incorrectly assumed for these operating conditions.

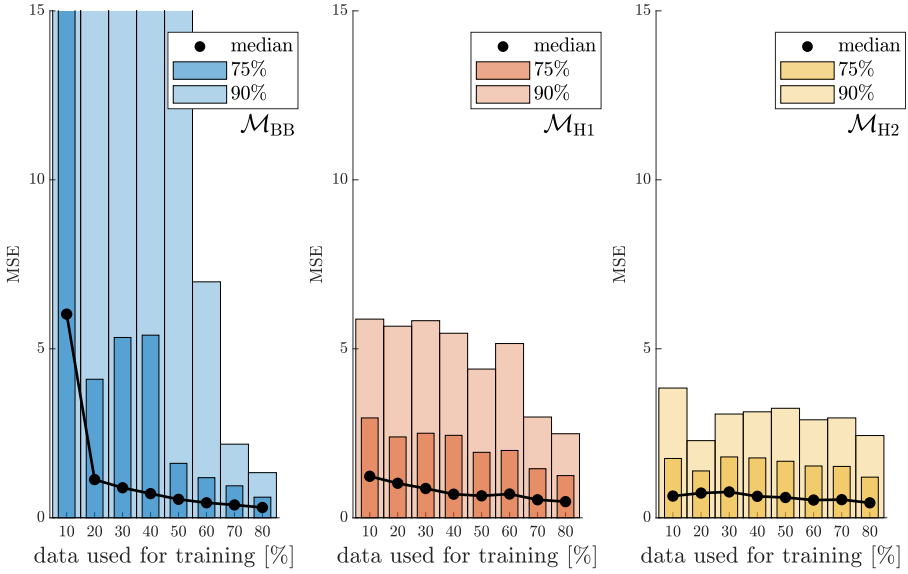


Figure 6.20: Prediction accuracy of $\hat{\mathcal{T}}$ for $\mathcal{I}_{\text{test}} = \{\mathcal{I}_9, \mathcal{I}_{10}\}$, associated with the 20% nominal condition most near to the bifurcation boundary Q. The training data are gradually increased (i.e., $\mathcal{I}_{\text{train}} = \mathcal{I}_1$ for 10%, $\mathcal{I}_{\text{train}} = \{\mathcal{I}_1, \mathcal{I}_2\}$ for 20% ...). Each experiment is repeated 10 times to cope with the variance ingrained in the models.

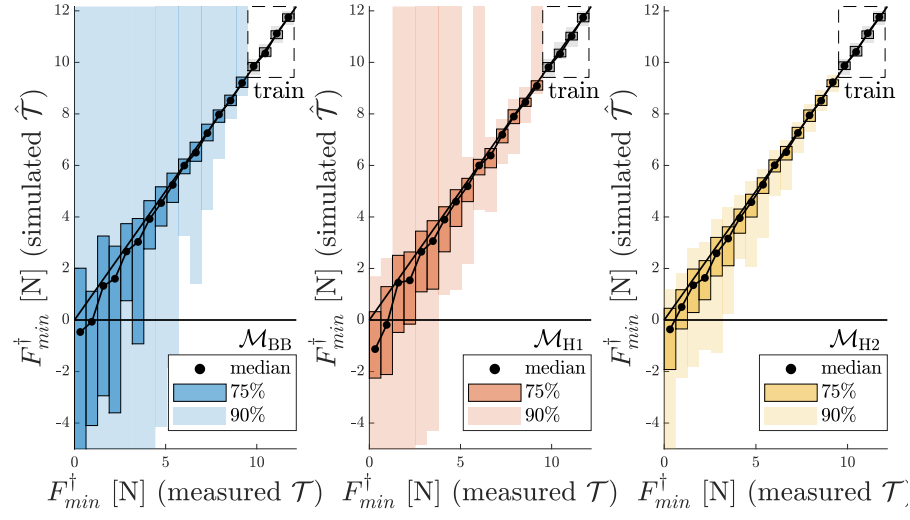


Figure 6.21: Comparison between the interface force metric $F^{\dagger}_{\text{min}}$ obtained directly from the measured trajectories $\mathcal{T}$ and from the simulated trajectories $\hat{\mathcal{T}}$ in $\mathcal{I}_{\text{test}} = \{\mathcal{I}_2, ..., \mathcal{I}_{10}\}$. The model $\mathcal{M}$ used to generate $\hat{\mathcal{T}}$ was trained on $\mathcal{I}_{\text{train}} = \mathcal{I}_1$.

6.5.4 Estimation of the Transition Boundary Q

The high accuracy of estimated minimal contact force F_{min} for the conditions in the neighborhood of the bifurcation boundary Q indicates the high potential of the (soft) hybrid modeling techniques when used to estimate Q. Figure 6.22 details the derived values of $\hat{Q}$ by means of a grid search in the parameter space $\mathbf{q}$. The estimation of $\hat{Q}$ based on the black-box models $\mathcal{M}_{\text{BB}}$ deteriorates once further removed from the training data. In contrast, although the (soft) hybrid architectures $\mathcal{M}_{\text{H1}}$ and $\mathcal{M}_{\text{H2}}$ are trained by only a limited number of samples, the physical laws included in the model provide sufficient generalization, leading to an accurate estimation of Q. A general visualization of the median results for $\hat{Q}$ is shown in Fig. 6.23. Again, these results indicate the enhanced exploration capabilities due to the physics-inspired laws in the (soft) hybrid modeling architectures.

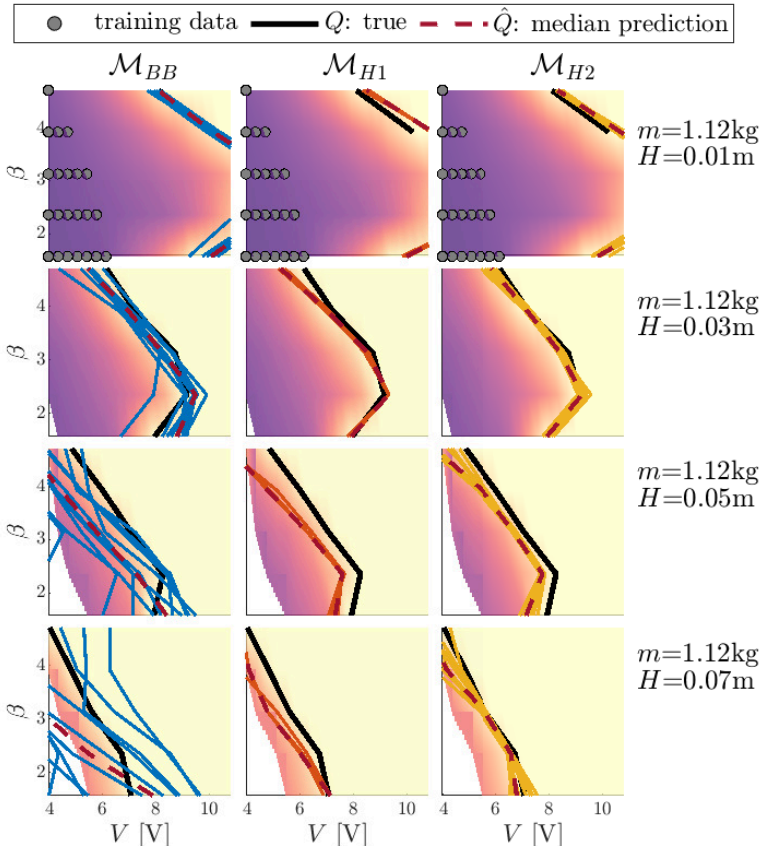


Figure 6.22: Estimation of the bifurcation boundary Q based on the estimated values of F_{min}. The experiment is repeated 10 times by retraining the model on a scarce data set $\mathcal{I}_{\text{train}} = \mathcal{I}_1$.

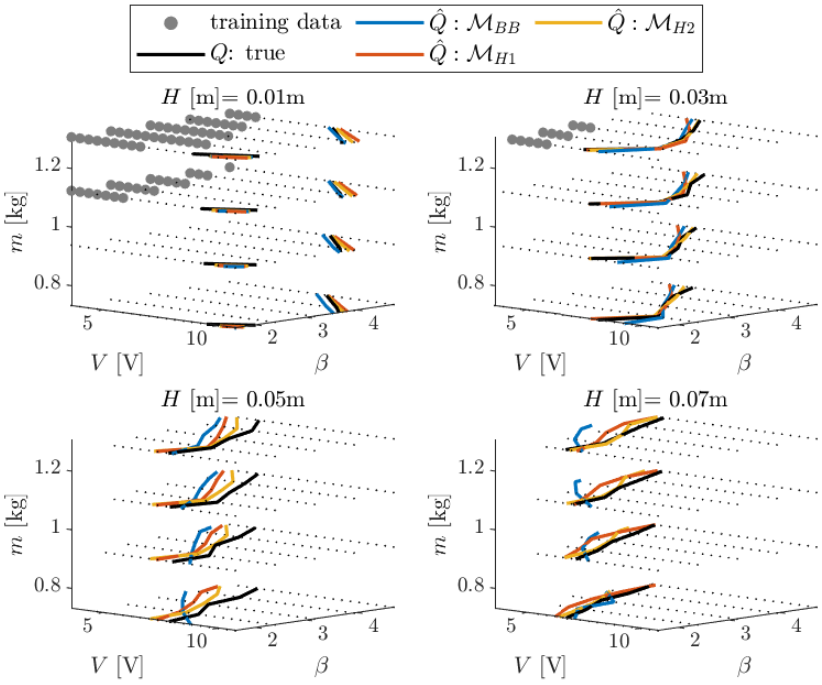


Figure 6.23: Full-grid visualization of the median estimation of the bifurcation boundaries $\hat{Q}$, trained by $\mathcal{I}_{\text{train}} = \mathcal{I}_1$.

Figure 6.24 summarizes the overall estimation accuracy of the bifurcation boundary $\hat{Q}$ for various amounts of training data used to learn the dynamics in $\mathcal{M}$. In addition, the exploration based on direct extrapolation in the parameter space of $\mathbf{q}$, as discussed in Section 6.3.3, is denoted as baseline. The presented approach, which explores the parameter space by simulating the unseen system settings, clearly outperforms the baseline reference if only a small amount of (sorted) data is available. Moreover, these results indicate the enhanced exploration performance obtained by including the partially known physics into the hybrid modeling architecture.

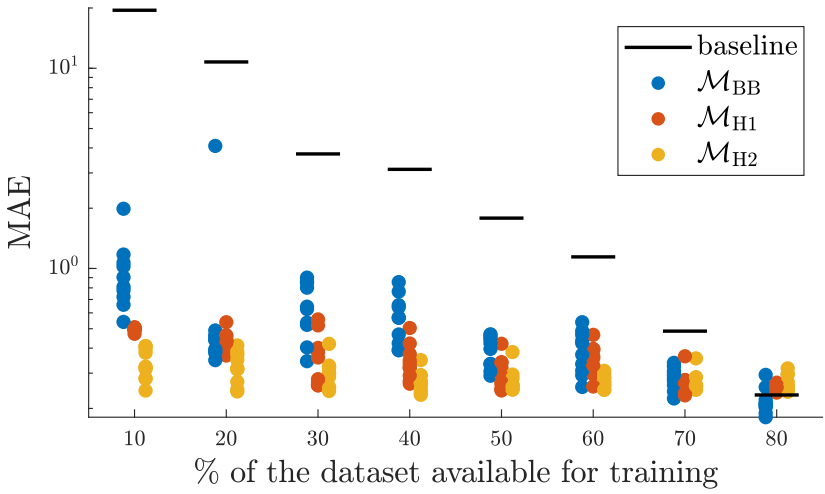


Figure 6.24: Mean absolute error (MAE) between $\hat{Q}$ and Q by gradually increasing the training data (i.e., $\mathcal{I}_{\text{train}} = \mathcal{I}_1$ for 10%, $\mathcal{I}_{\text{train}} = \{\mathcal{I}_1, \mathcal{I}_2\}$ for 20% ...). Each experiment is repeated 10 times.

6.6 Conclusion

This chapter presents a detailed analysis of the dynamics induced by a cam-follower mechanism. A full factorial design of 1600 different system modifications has been performed to characterize the influence of the system parameters on the possible occurrence of follower jumps. The objective of this research was twofold. First, physics-based neural network models are used to capture the influence of the system parameters on the dynamics by learning directly from time series associated with a limited set of parameter settings. We compared the prediction performance of two hybrid modeling architectures, characterized by a close combination of physics-inspired and neural layers, against their black-box counterpart. The enhanced generalization obtained by providing some partially known physics was especially pronounced when the models are evaluated for parameter settings far beyond the region for which they have seen training data. Second, these enhanced extrapolation capabilities have successfully been used to identify the critical set of parameters for which detachment between cam and follower occurs, by solely learning the dynamics from a small dataset of nominal operating conditions. While the acquired results mainly focus on the cam-follower mechanism, the presented methodologies can be of interest for many parameter optimization challenges in a vast amount of mechatronic applications.

References

[1] H. A. Rothbart. Cam Design Handbook. McGraw-Hill handbooks. McGraw-Hill, 2004.

[2] N. Nayak, P. A. Lakshminarayanan, M. K. Gajendra Babu, and A. D. Dani. Predictions of cam follower wear in diesel engines. Wear, 260(1-2):181–192, 2006.

[3] B. A. Paden, S. T. Snyder, B. E. Paden, and M. R. Ricci. Modeling and control of an electromagnetic variable valve actuation system. IEEE/ASME Transactions on Mechatronics, 20(6):2654–2665, 2015.

[4] T. Lenzi, M. Cempini, L. J. Hargrove, and T. A. Kuiken. Design, development, and validation of a lightweight nonbackdrivable robotic ankle prosthesis. IEEE/ASME Transactions on Mechatronics, 24(2):471–482, 2019.

[5] T. Xing, Y. Xu, and J. Ruan. Two-dimensional piston pump: Principle, design, and testing for aviation fuel pumps. Chinese Journal of Aeronautics, 2019.

[6] E. Ottaviano, D. Mundo, G. A. Danieli, and M. Ceccarelli. Numerical and experimental analysis of non-circular gears and cam-follower systems as function generators. Mechanism and Machine Theory, 43(8):996–1008, 2008.

[7] J. Xiang, Z. Cai, Y. Zhang, and W. Wang. A micro-cam actuated linear peristaltic pump for microfluidic applications. Sensors and Actuators A: Physical, 251:20–25, 2016.

[8] H. Abderazek, A. Yildiz, and S. Mirjalili. Comparison of recent optimization algorithms for design optimization of a cam-follower mechanism. Knowledge-Based Systems, 191:105237, 2020.

[9] H.-S. Yan, M.-C. Tsai, and M.H. Hsu. An experimental study of the effects of cam speeds on cam-follower systems. Mechanism and Machine Theory, 31(4):397–412, 1996.

[10] R. Alzate, M. Di Bernardo, U. Montanaro, and S. Santini. Experimental and numerical verification of bifurcations and chaos in cam-follower impacting systems. Nonlinear Dynamics, 50(3):409, 2007.

[11] G. Osorio, M. di Bernardo, and S. Santini. Corner-impact bifurcations: a novel class of discontinuity-induced bifurcations in cam-follower systems. SIAM Journal on Applied Dynamical Systems, 7(1):18–38, 2008.

[12] S. Sundar, J. T. Dreyer, and R. Singh. Estimation of impact damping parameters for a cam–follower system based on measurements and analytical model. Mechanical Systems and Signal Processing, 81:294–307, 2016.

[13] P. Flores, R. Leine, and C. Glocker. Application of the nonsmooth dynamics approach to model and analysis of the contact-impact events in cam-follower systems. Nonlinear Dynamics, 69(4):2117–2133, 2012.

[14] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi. Applications of machine learning to machine fault diagnosis: A review and roadmap. Mechanical Systems and Signal Processing, 138:106587, 2020.

[15] C. Sobie, C. Freitas, and M. Nicolai. Simulation-driven machine learning: Bearing fault classification. Mechanical Systems and Signal Processing, 99:403–419, 2018.

[16] S. Kim. Moment of inertia and friction torque coefficient identification in a servo drive system. IEEE Transactions on Industrial Electronics, 66(1):60–70, 2019.

[17] J. Montonen, N. Nevaranta, T. Lindh, J. Alho, P. Immonen, and O. Pyrhönen. Experimental identification and parameter estimation of the mechanical driveline of a hybrid bus. IEEE Transactions on Industrial Electronics, 65(7):5921–5930, 2018.

[18] J. Schoukens and L. Ljung. Nonlinear system identification: A user-oriented road map. IEEE Control Systems Magazine, 39(6):28–99, 2019.

[19] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. arXiv preprint arXiv:1801.01236, 2018.

[20] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. Ineternational Conference on Learning Representations, 2019., 2019.

[21] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In Advances in Neural Information Processing Systems, pages 15353–15363, 2019.

[22] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In Proceedings of the 2019 SIAM Int. Conf. on Data Mining, pages 558–566. SIAM, 2019.

[23] N. Mohajerin and S. L. Waslander. Multistep prediction of dynamic systems with recurrent neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2019.

[24] A. Ajay, M. Bauza, J. Wu, N. Fazeli, J. B. Tenenbaum, A. Rodriguez, and L. P. Kaelbling. Combining physical simulators and object-based networks for control. arXiv preprint arXiv:1904.06580, 2019.

[25] A. Punjani and P. Abbeel. Deep learning helicopter dynamics models. In IEEE International Conference on Robotics and Automation, pages 3223–3230. IEEE, 2015.

[26] W. De Groote, E. Kikken, E. Hostens, S. Van Hoecke, and G. Crevecoeur. Neural network augmented physics models for systems with partially unknown dynamics: Application to slider-crank mechanism. IEEE/ASME Transactions on Mechatronics, 2021.

[27] A. Rahideh, A. H. Bajodah, and M. Shaheed. Real time adaptive nonlinear model inversion control of a twin rotor mimo system using neural networks. Engineering Applications of Artificial Intelligence, 25(6):1289–1297, 2012.

[28] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989.

[29] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[30] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. arXiv preprint arXiv:1710.05941, 2017.

[31] F. Chollet et al. Keras. `https://keras.io`, 2015.

[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symp. on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016.

[33] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Chapter 7

Conclusion and Future Perspectives

This dissertation presents an extensive analysis of hybrid modeling approaches, encompassing a close combination of physics-inspired and black-box neural network components, to predict the nonlinear dynamics in mechatronic applications. To develop the next generation of mechatronic applications, these models need to be accurate, robust and explainable. Furthermore, the prediction performances should be validated on experimental data originating from mechatronic applications to demonstrate their practical impact. These properties are extensively studied in this dissertation. Various modeling approaches are proposed and applied to different mechatronic applications. Below, we conclude the research results, methods and insights from the different chapters.

Chapter 1 introduced the emerging data-flows appearing in a vast amount of industrial sectors. The interconnection between either the machines on plant level and the communication between the components on machine level enables a new generation of high-performing systems. In particular, this evolution is apparent in the increasing complexity of mechatronic applications, which combine various aspects of mechanical, electrical, control and computer science engineering. Digital twins, which can be considered as the real-time counterpart of the system, can help to align all these complex interactions. Nevertheless, this requires the development of accurate and reliable digital models that can mimic the system behavior for varying operating conditions.

Chapter 2 detailed the traditional modeling approaches which typically consider either a physics-inspired or a data-driven methodology. Next, the premise of proposing hybrid modeling approaches was presented using an

explanatory case. This demonstrated why the central approach within this thesis, hybrid models, is endeavored to learn the system dynamics from collected data.

Chapter 3 illustrated the burden associated to developing a physics-inspired model of an intricate mechatronic application such as a wet-friction clutch. We demonstrated that it is almost inevitable that the system is plagued by unknown interactions that are not incorporated in the physics-inspired model. We addressed this problem by introducing a probability distribution associated to the uncertain model parameters. This way, we could explain the model mismatches of the shifting time for different operating conditions. The probability distributions were identified via an inverse identification strategy that propagates the uncertainty through the model in a computationally efficient manner. More precisely, high-order statistical moments of the output distribution were calculated via a generalized polynomial chaos framework to significantly decrease the required amount of model evaluations.

In Chapter 4 we approached the shortcomings of physics-inspired models of mechatronic applications by a more fundamental approach. Instead of recognizing that some interactions of a slider-crank mechanism could not be modeled, we complemented the partially known dynamics with a neural network model. The so-called neural-network-augmented-physics (NNAP) model incorporates both physics-inspired and neural layers. We showed that the neural and physical parameters could be simultaneously updated solely by using state measurements and control values. This way, we obtained a reliable hybrid model that accurately simulates the system behavior. Furthermore, we showed that by extracting the neural network after convergence, we could retrieve new insights about the force interactions acting on the mechanism.

Chapter 5 approached the combination of neural network models and physical expert knowledge from another perspective. Instead of starting from an almost complete physics model, we considered the situation for which we only have limited insights about the system. Therefore, a data-driven approach was considered in which we tried to monitor follower jumps in a cam-follower system by using an LSTM neural network model. We showed that by providing physics-inspired input features to the model, we could significantly improve the prediction performances. Next, we implemented a SHAP model to quantify the contribution of the features to the model output. This explanation model could objectively determine the beneficial influences of the provided physics on the model predictions.

In Chapter 6 we assessed the extrapolation capabilities of the hybrid

modeling architectures with respect to their data-driven counterpart. We showed that for sparse datasets, i.e. hazardous phenomena do (luckily) not occur often in mechatronic systems, the inclusion of the known system dynamics clearly brings additional robustness to the model architecture. We learned the cam-follower dynamics from a limited set of system parameters and observed accurate predictions for design and control settings far beyond the machine parameters used to train the hybrid model. This way, the set of critical parameters settings that lead up to hazardous jump phenomena was identified starting from data of nominal operating conditions.

7.1 Conclusion

To conclude this dissertation, we summarize the main findings for each of the research challenges described in Chapter 1.

- **Prediction performances**
  A major disadvantage of physics-inspired dynamic models is that the expert knowledge is typically not sufficient to model all interactions within the mechatronic system. In Chapter 3 we addressed this problem by assigning a probability distribution to uncertain parameters. We validated this approach on a wet friction cluch system, resulting in the stochastic output distributions shown in Fig. 3.14. This procedure became computationally feasible due to the proposed efficient gPC higher-order moment matching framework that reduced the required amount of function evaluations with an order of magnitude, as illustrated in Fig. 3.17. A more profound approach to obtain accurate predictions was obtained by augmenting the partially known dynamics with a neural network (NNAP), as presented in Chapter 4. The prediction results shown in Fig. 4.13 illustrate the hybrid model being able to capture all interactions present in the slider-crank mechanism. In Chapter 5 we studied the possibility to provide physics-inspired input features to an LSTM neural network to accurately predict follower jumps in cam-follower systems. The obtained results indicated that the physics-inspired features alleviate the modeling effort of the recurrent neural network, resulting into enhanced generalization capabilities, as was shown in Fig. 5.15 and Fig. 5.16.

- **Robustness**
  A key focus of this dissertation was placed on the robustness, which is often a restraining factor to implement neural network architectures in industrial machines. We showed in Fig. 4.13 that the NNAP model remained stable when predicting the slider-crank behavior for higher oper-

ational speeds than used to train the model. The experiments performed on the cam-follower system in Chapter 5 and 6 were highly suited to assess the prediction performances for design modifications (e.g., cam shape) or control parameters (e.g., voltage). Figure 5.16 illustrated that the physics-inspired features of the LSTM model enable enhanced generalization capabilities, which are especially more pronounced when the data is scarce or not uniformly distributed along the parameter space. Figure 6.20 illustrated that the hybrid modeling architectures perform significantly better compared to their data-driven counterpart when evaluated for system parameters that are situated far beyond the training dataset. This increased robustness could be exploited to assess the system behavior on unseen design and control parameters. This enabled the identification of the set of critical parameter values that led up to the unwanted detachment behavior, as illustrated in Fig. 6.23 and Fig. 6.24.

- **Explainability**
  The limited interpretability of data-driven models often pushes mechatronics engineers towards the use of physics-inspired system models. Therefore, we elaborated on the information that could be retrieved from the (hybrid) neural network models, aiming for improved explainability. In Chapter 4 we intentionally excluded the interaction between the slider and spring in the physics model. The force interactions were replaced by a randomly initialized neural network model. Without having explicit data of the spring force, we could, by propagating the prediction errors of the system states, learn the overall system dynamics. Afterwards, once the NNAP model converged, we could retrieve new insights from the neural network such as the spring law and additional friction phenomena, as was shown in Fig. 4.18. The NNAP model was trained by simultaneously updating the neural and physical parameters. This results into a high-dimensional optimization problem that might get stuck in local minima, leading to non-unique solutions. Therefore, we also did a sensitivity study in Section 4.7 to identify the set of parameters that can be simultaneously optimized with the neural network. This way, we could validate that the retrieved information of the neural network represents the actual unmodeled phenomena (that were not included in the physics-inspired model). In Chapter 5 we discussed the beneficial influence of physics-inspired input features on the prediction performances of detachment phenomena in cam-follower systems. To increase the interpretability of the model, we implemented the SHAP framework on this recurrent model to enable an explanation model that quantifies the contribution of all features. As discussed in Fig. 5.23, this approach enabled to quantify how much the LSTM prioritizes the physics-inspired

features above the raw measurements.

- **Experimental Validation**
  This dissertation included the construction of various mechatronic setups and the collection of experimental datasets. These significant efforts were needed to validate the methodologies on experimental mechatronic setups and specifically to apprehend their nonlinear system dynamics. We considered a wet friction clutch, slider-crank and a modular cam-follower mechanism (see Fig. 1.7). The effectiveness of the presented methodologies on the laboratory setups showed that these approaches go beyond just being a theoretical framework and can cope with imperfections such as sensor noise and environmental disturbances. This demonstrated robustness of the algorithms can be of mayor importance to initiate further research on applying (physics-inspired) machine learning on mechatronic applications.

7.2 Future Perspectives

The presented research, in which we combined physics-inspired and data-driven approaches, showed promising results to model the nonlinear behavior of mechatronic systems. Consequently, this research can be considered as a valuable contribution in the development of the next generation mechatronic applications. Nevertheless, there are still some open research questions that need further exploration. Below, we provide some suggestions of future research topics, starting from a fundamental perspective towards direct applications of hybrid modeling approaches on mechatronic systems.

7.2.1 Fundamental Research

- The presented hybrid state-space approaches are applied on nonlinear systems characterized by a low-dimensional system state. For more complex mechatronic applications, there are typically multiple unknown interactions that can be considered as coupled subsystems acting on different time scales. Takens' embedding theorem already stated the required conditions to construct a dynamic systems, starting from a sequence of measurements [1]. However, in practice, when the system is highly nonlinear, neural networks can get stuck in local minima during the optimization process [2]. "Multiple shooting" methods can offer a solution to avoid these local minima when learning the system dynamics [3]. Next to convergence problems, the approach can be plagued by identifiable issues leading to non-unique solutions. In Chapter 4 we

demonstrated a local sensitivity analysis to identify the identifiable physical parameters that can be simultaneously optimized with the neural network. For more complex systems, more elaborate approaches need to be developed that can determine both the structural and practical identifiability of high-dimensional optimization problems [4–6].

- The convergence of the neural network leads to an accurate overall model that can be used to predict the system behavior of mechatronic applications. A major advantage arises when the information in the neural network can be extracted to gain new insights in the unmodeled phenomena, as illustrated in Chapter 4. For more complex systems the extraction of information relying on multidimensional inputs becomes less trivial. Therefore, further research should place emphasis on automating the interpretability of the information captured in the black-box components. Research is needed to try other data-driven modeling architectures that split up linear and nonlinear terms in the regression model [7]. Alternatively, the presented formalisms can be combined with symbolic regression techniques that provide analytical expressions of the fitted phenomena, leading to more interpretable representations [8, 9].

- The hybrid state space models studied in this thesis always considered a fully measured state. In practice, the measurements do often not encompass the full system state. Moreover, if certain dynamic sub-processes are unknown, we should augment the system state with an unknown amount of unmeasured state variables. In practice, an infinite amount of solutions exist since any valid coordinate transformation yields another state-space realization [10]. Consequently, we should aim for the minimal state-space representation, which describes the system by a minimal amount of state variables, to obtain maximum interpret ability of the system dynamics. Although the minimal state dimensions can be easily determined from the impulse responses or input-output data of the system for linear systems, the situation is far more complex when the dynamics should be captured by a nonlinear model [11].

- In this dissertation we studied the behavior of time-invariant systems, meaning that the system properties do not vary trough time. However, in practice, subtle systems changes (e.g., temperature, friction) and external disturbances can influence the system behavior. Adaptive algorithms that can update the physical system parameters in white-box models can be found in the literature [12, 13]. Also, fitted data-driven methods that contain a limited amount of parameters such as linear or polynomial models can be updated online [14, 15]. However, problems

arise when high-dimensional parameter models such as neural networks should be updated based on online data streams. The use of incrementally available information obtained from non-stationary data distributions can lead to catastrophic forgetting or interference in the neural network models [16]. Therefore, research should be devoted to find balanced adaptive update algorithms for hybrid modeling approaches that can keep up with the system changes without compromising on the generalization and robustness of the model.

7.2.2 Improving Mechatronic Applications

As described in Chapter 1 the use of accurate dynamic system models has already many applications in the design, control and sensing of mechatronic systems. Consequently, the operational performances of future mechatronic applications can be further improved by replacing the traditional physics-inspired or data-driven models with hybrid approaches.

- **Design**: Model-based design approaches typically require an accurate (physics) model that outputs the performances for given design parameters. These methods become challenging if the systems are too complex to be fully described by physical laws. In Chapter 6 we showed that the influence of design parameters can be captured by a hybrid model where the unmodeled interactions were replaced by a neural network. In addition, we demonstrated that the model remained reliable when evaluated for completely unseen system designs. Consequently, this can be an incentive to incorporate these hybrid architectures in model-based optimization strategies that numerically determine the optimal design parameters.

- **Control**: The high accuracy and reliability of the hybrid modeling approaches generate opportunities to use these in online control algorithms. Currently, the practical implementation of neural network system models in model-based control algorithms mainly concerns systems that exhibit relatively high time constants such as chemical processes or HVAC systems [17, 18]. Nevertheless, the continuous advances in dedicated hardware designs for neural network architectures can potentially provide sufficient computing power in the near future [19]. Therefore, future research should focus on incorporating these hybrid architectures in model-based control frameworks to increase the operational performances of mechatronic applications.

- **Sensing**: The use of accurate dynamic system models in Kalman filtering techniques has a long history of providing estimations of unknown variables in industrial applications [20]. These methodologies

have shown their benefits in identifying the unknown system parameters (e.g., inertia) in physics-based models [21,22]. In analogy, extended and uncented Kalman filtering techniques have been used to estimate system states while simultaneously updating the neural network model that captures the system behavior [23, 24]. Therefore, in future research, hybrid approaches can be embedded in Kalman filtering frameworks in which unknown physical parameters can be simultaneously updated with the neural parameters to obtain adaptive sensing methodologies that can be used to estimate the unknown variables in mechatronic applications.

References

[1] F. Takens. Detecting strange attractors in turbulence. In Dynamical systems and turbulence, Warwick 1980, pages 366–381. Springer, 1981.

[2] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. arXiv preprint arXiv:1712.09913, 2017.

[3] A. H. Ribeiro, K. Tiels, J. Umenberger, T. B. Schön, and L. A. Aguirre. On the smoothness of nonlinear system identification. Automatica, 121:109158, 2020.

[4] K. Z. Yao, B. M. Shaw, B. Kou, K. B. McAuley, and D.W. Bacon. Modeling ethylene/butene copolymerization with multi-site catalysts: parameter estimability and experimental design. Polymer Reaction Engineering, 11(3):563–588, 2003.

[5] A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. Bioinformatics, 25(15):1923–1929, 2009.

[6] H. Miao, X. Xia, A. S. Perelson, and H. Wu. On identifiability of nonlinear ode models and applications in viral dynamics. SIAM review, 53(1):3–39, 2011.

[7] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon. Identification of nonlinear systems using polynomial nonlinear state space models. Automatica, 46(4):647–656, 2010.

[8] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the national academy of sciences, page 201517384, 2016.

[9] S. Khatiry Goharoodi, K. Dekemele, M. Loccufier, L. Dupre, and G. Crevecoeur. Evolutionary-based sparse regression for the experimental identification of duffing oscillator. Mathematical Problems in Engineering, 2020, 2020.

[10] R. L. Williams and D. A. Lawrence. Linear state-space control systems. John Wiley & Sons, 2007.

[11] B. De Schutter. Minimal state-space realization in linear system theory: an overview. Journal of computational and applied mathematics, 121(1-2):331–354, 2000.

[12] D. Papageorgiou, M. Blanke, H. H. Niemann, and J. H. Richter. Robust backlash estimation for industrial drive-train systems: Theory and validation. IEEE Transactions on Control Systems Technology, 27(5):1847–1861, 2018.

[13] K. A. J. Verbert, R. Tóth, and R. Babuška. Adaptive friction compensation: a globally stable approach. IEEE/ASME Transactions on Mechatronics, 21(1):351–363, 2015.

[14] N. Nevaranta, J. Parkkinen, T. Lindh, M. Niemelä, O. Pyrhönen, and J. Pyrhönen. Online estimation of linear tooth belt drive system parameters. IEEE Transactions on Industrial Electronics, 62(11):7214–7223, 2015.

[15] P. Yadmellat and M. R. Kermani. Adaptive modeling of a magnetorheological clutch. IEEE/ASME Transactions on Mechatronics, 19(5):1716–1723, 2013.

[16] G. I Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. Neural Networks, 113:54–71, 2019.

[17] A. l. Afram, F. Janabi-Sharifi, A. S. Fung, and K. Raahemifar. Artificial neural network (ann) based model predictive control (mpc) and optimization of hvac systems: A state of the art review and case study of a residential hvac system. Energy and Buildings, 141:96–113, 2017.

[18] D. L. Yu and J. B. Gomm. Implementation of neural network predictive control to a multivariable chemical reactor. Control engineering practice, 11(11):1315–1323, 2003.

[19] E. Wang, J. J. Davis, R. Zhao, H.-C. Ng, X. Niu, W. Luk, P. Y.K. Cheung, and G. A. Constantinides. Deep neural network approximation for cus-

tom hardware: Where we've been, where we're going. ACM Computing Surveys (CSUR), 52(2):1–39, 2019.

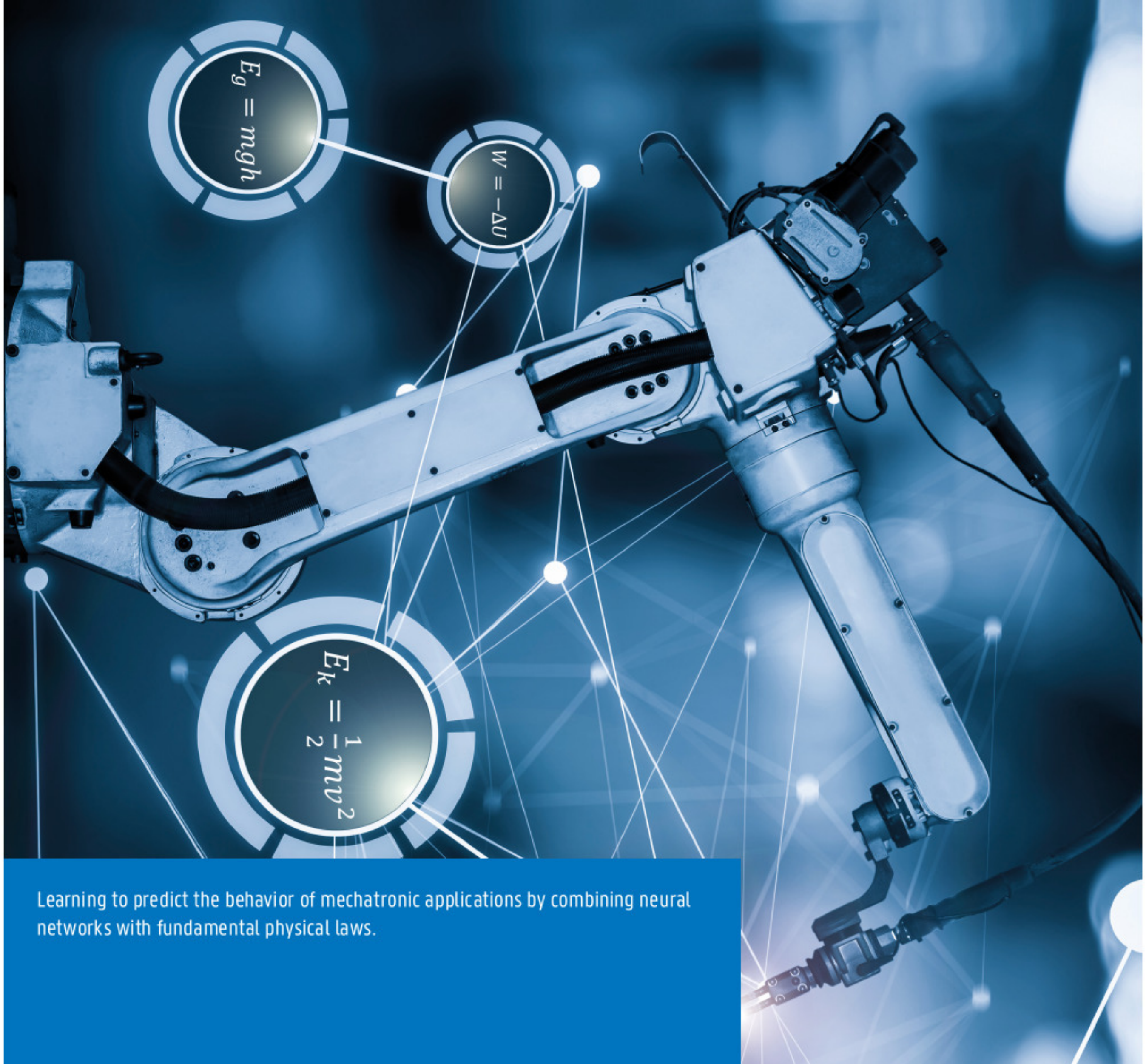
[20] F. Auger, M. Hilairet, J. M. Guerrero, E. Monmasson, T. Orlowska-Kowalska, and S. Katsura. Industrial applications of the kalman filter: A review. IEEE Transactions on Industrial Electronics, 60(12):5458–5471, 2013.

[21] P. Manganiello, M. Ricco, G. Petrone, E. Monmasson, and G.i Spagnuolo. Dual-kalman-filter-based identification and real-time optimization of pv systems. IEEE Transactions on Industrial Electronics, 62(11):7266–7275, 2015.

[22] S. Hong, C. Lee, F. Borrelli, and J. K. Hedrick. A novel approach for vehicle inertial parameter identification using a dual kalman filter. IEEE Transactions on Intelligent Transportation Systems, 16(1):151–161, 2014.

[23] R. Zhan and J. Wan. Neural network-aided adaptive unscented kalman filter for nonlinear state estimation. IEEE Signal Processing Letters, 13(7):445–448, 2006.

[24] S. C. Stubberud, R. N. Lobbia, and M. Owen. An adaptive extended kalman filter using artificial neural networks. In Proceedings of 1995 34th IEEE Conference on Decision and Control, volume 2, pages 1852–1856. IEEE, 1995.



Learning to predict the behavior of mechatronic applications by combining neural networks with fundamental physical laws.