Optimization Techniques for Data-Based Control and Machine Learning

Von der Fakultät für Elektrotechnik, Informatik und Mathematik der Universität Paderborn zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation von

Katharina Bieker

Paderborn, 2023

Autorin
Katharina Bieker

Gutachter:innen
Jun.-Prof. Dr. Sebastian Peitz
Prof. Dr. Sina Ober-Blöbaum
Prof. Dr. Stefan Klus

Tag der mündlichen Prüfung
20.04.2023

Danksagung

An dieser Stelle möchte ich einigen Menschen danken, die mir durch ihre Unterstützung das Erstellen und Schreiben dieser Arbeit ermöglicht und erleichtert haben.

Zuallerest möchte ich Jun.-Prof. Dr. Sebastian Peitz für seine Unterstützung und die Betreuung während der letzten Jahre danken. Die Zusammenarbeit hat mir immer viel Spaß gemacht und durch seine optimistische Art wusste er mich auch bei Rückschlägen stets zu motivieren. Außerdem danke ich Prof. Dr. Michael Dellnitz für die Möglichkeit, diese Arbeit an seinem Lehrstuhl zu erstellen.

Neben Sebastian Peitz möchte ich auch Prof. Dr. Sina Ober-Blöbaum und Prof. Dr. Stefan Klus dafür danken, dass sie sich die Zeit genommen haben, meine Arbeit zu begutachten und zu bewerten.

Des Weiteren gilt ein großes Dankeschön meinen ehemaligen Arbeitskollegen während der letzten Jahre: Manuel Berkemeier, Bennet Gebken, Raphael Gerlach, Sören von der Gracht, Emina Hadzialic, Benjamin Jurgelucks, Marianne Kalle, Lukas Lanza, Karin Mora, Feliks Nüske, Veronika Schulze, Konstantin Sonntag, Olga Weiß und Adrian Ziessler. Vielen Dank für die interessanten und aufschlussreichen fachlichen Diskussionen, aber auch für die vielen Kaffee- und Mittagspausen, die mir den ein oder anderen Tag verschönert haben. Ganz besonders möchte ich mich bei denjenigen bedanken, die jeweils Teile dieser Arbeit Korrektur gelesen haben.

Außerdem bin ich dankbar für die finanzielle Unterstützung durch das BMBF im Rahmen des Forschungsprojektes "IBOSS – Information-Based Optimization of Surgery Schedules" und durch das Land NRW sowie der EU im Rahmen des Projektes "SET CPS – Simultanes Entwickeln und Testen von Cyber Physical Systems (CPS) am Anwendungsbeispiel eines elektrisch angetriebenen autonomen Fahrzeugs". Zusätzlich möchte ich dem PC^2 meinen Dank ausrichten, dafür, dass ich einen Teil meiner Berechnungen auf dem OCuLUS Cluster durchführen konnte.

Ein besonderer Dank gilt meiner Familie, die mich auf meinem Weg immer unterstützt hat und bei der ich immer Rückhalt finde. Insbesondere danke ich meinen Eltern dafür, dass sie mich dazu ermuntert haben, das zu machen, was ich gerne mache, und mir das Studium ermöglicht haben. Auch danke ich meinen beiden Brüdern, Manuel und Daniel, dafür, dass sie immer ein offenes Ohr für mich haben. Auch meinen Freunden in Paderborn, Iserlohn und an den ganzen anderen Orten, an die es sie verstreut hat, möchte ich dafür danken, dass es mit ihnen auch eine Welt außerhalb der Mathematik gibt. Zu guter Letzt möchte ich Marcel dafür danken, dass er immer für mich da ist und die letzten Monate viel Geduld mit mir hatte, trotz einigen hundert Kilometern Distanz.

Abstract

The control of complex dynamical systems is of great importance in many different engineering applications, for example, in autonomous driving or for the control of combustion processes in power plants. There, the so-called *model predictive control* is often used due to the simple implementation and high control quality. Thereby, the control input is optimized based on a suitable surrogate model that can predict the behavior of the underlying dynamical system with sufficient accuracy over a given time horizon. Especially in recent years, data-based methods and *machine learning* are increasingly used to build surrogate models, for instance, *neural networks*. This thesis addresses various aspects of data-based surrogate models, in particular neural networks, and their use in the context of model predictive control.

In the first part of this thesis, the potential of neural networks for the prediction and control of complex dynamical systems is investigated. For this purpose, a surrogate model based on a *recurrent neural network* is presented and used to control a fluid dynamical system. On the one hand, it becomes clear that using data-based methods for controlling complex systems is quite promising. On the other hand, there are still difficulties in the implementation of such methods and open questions from a theoretical point of view, which are pointed out and discussed. For example, modeling systems with a time-dependent control input is much more complex than modeling autonomous models without additional control input.

This issue is addressed in the second part of this thesis. There, an approach is presented in which continuous control problems are modified to allow for a simpler implementation of surrogate models for model predictive control using many existing data-based methods. To this end, the control input is discretized so that only autonomous surrogate models are required for given control inputs. Error bounds are derived, and the method is validated on several examples with different surrogate models. In addition, the extent to which the presented approach reduces the amount of data needed for training compared to a single model with control input is discussed.

Regardless of whether models with or without control input are considered, they should generalize well, i.e., the prediction quality on the *training data* should not be (significantly) better than that on independent *test data*. If the model is fitted too closely to the training data, this is called *overfitting*. To avoid this problem, so-called *regularization* procedures are often used. This typically involves adding a weighted regularization term, e.g., the ℓ_1-norm, to the error function. The additional regularization term ensures the generation of models which are as sparse as possible and, therefore, typically less prone to overfitting. By varying the regularization parameter, different solutions are obtained, which together form the so-called *regularization path*. In the third part, this regularization problem is considered as a (nonsmooth) *multiobjective optimization problem*, i.e., the error function and the regularization term are optimized simultaneously. In the case where the ℓ_1-norm is used as the regularization term, based on results from the field of multiobjective optimization, the structure of the so-called *Pareto critical set*, which is a superset of the regularization path, is analyzed. Furthermore, a continuation method for com-

puting the critical set is presented, which is specifically tailored to this problem and takes into account the nonsmoothness of the ℓ_1-norm. Afterwards, the proposed method is used to show, based on the training of neural networks, that relevant solutions that generalize well are not included in the regularization path and, thus, cannot be computed with the regularization procedures used in classical approaches. Furthermore, it is discussed to what extent the presented algorithm scales for larger problem instances. Finally, an outlook on the extension to more general regularization problems is given, where the regularization term is only piecewise smooth. More precisely, the structure of the Pareto critical set for this case is analyzed to provide the basis for developing new continuation methods for this class of problems.

Zusammenfassung

Die Steuerung und Regelung komplexer dynamischer Systeme hat in vielen verschiedenen ingenieurwissenschaftlichen Anwendungen eine große Bedeutung, beispielsweise im Bereich des autonomen Fahrens oder bei der Regelung von Verbrennungsprozessen in Kraftwerken. Hier kommt auf Grund der einfachen Umsetzbarkeit und der hohen Regelgüte häufig die sogenannte *modellprädiktive Regelung* zum Einsatz. Dabei wird der Kontrolleingang basierend auf einem geeigneten Ersatzmodell, welches das Verhalten des zugrundeliegenden dynamischen Systems über einen gegebenen Zeithorizont ausreichend genau vorhersagen kann, optimiert. Insbesondere in den letzten Jahren werden zur Bildung der Ersatzmodelle vermehrt datenbasierte Verfahren und *maschinelles Lernen* eingesetzt, wie zum Beispiel *neuronale Netze*. In dieser Arbeit werden verschiedene Aspekte datenbasierter Ersatzmodelle, insbesondere neuronaler Netze, und deren Einsatz im Rahmen der modellprädiktiven Regelung adressiert.

Im ersten Teil der Arbeit wird zunächst das Potential von neuronalen Netzen zur Vorhersage und Regelung komplexer dynamischer Systeme untersucht. Dazu wird ein Ersatzmodell basierend auf *rekurrenten neuronalen Netzen* vorgestellt und zur Regelung eines fluiddynamischen Systems verwendet. Dabei wird zum einen deutlich, dass der Einsatz datenbasierter Methoden für die Regelung komplexer Systeme überaus vielversprechend ist, zum anderen werden aber auch Schwierigkeiten bei der praktischen Umsetzung solcher Verfahren und offene Fragestellungen aus theoretischer Sicht aufgezeigt und diskutiert. So ist beispielsweise die Modellierung von Systemen mit zeitabhängigem Kontrolleingang wesentlich komplexer als die Modellierung autonomer Modelle ohne zusätzlichen Kontrolleingang.

Dieses Problem wird im zweiten Teil der Arbeit adressiert. Dort wird ein Ansatz vorgestellt, bei dem kontinuierliche Kontrollprobleme so abgeändert werden, dass die Ersatzmodellbildung für die modellprädiktive Regelung einfacher und mit vielen verschiedenen bestehenden datenbasierten Verfahren umgesetzt werden kann. Dazu wird der Kontrolleingang diskretisiert, sodass nur autonome Ersatzmodelle für vorgegebene Kontrolleingänge benötigt werden. Es werden Fehlerschranken hergeleitet und die Methode an verschiedenen Beispielen mit unterschiedlichen Ersatzmodellen validiert. Zudem wird diskutiert, inwieweit der vorgestellte Ansatz die Menge der für das Training benötigten Daten gegenüber einem einzelnen Modell mit Kontrolleingang reduziert.

Unabhängig davon, ob Modelle mit oder ohne Kontrolleingang betrachtet werden, sollten diese gut generalisieren, d.h. die Vorhersagegüte auf den *Trainingsdaten* sollte nicht (wesentlich) besser sein als die auf unabhängigen *Testdaten*. Ist das Modell zu sehr auf die Trainingsdaten angepasst, spricht man von *Overfitting*. Um dieses Problem zu vermeiden, werden häufig sogenannte *Regularisierungsverfahren* eingesetzt. Dabei wird typischerweise auf die Fehlerfunktion ein gewichteter Regularisierungsterm, z.B. die ℓ_1-Norm, addiert. Dies sorgt dafür, dass möglichst dünn besetzte Modelle generiert werden, welche normalerweise weniger zu Overfitting neigen. Durch Variieren des Regularisierungsparameters erhält man verschiedene Lösungen, die zusammengenommen den sogenannten *Regularisierungspfad* bilden. Im dritten Teil

wird dieses Regularisierungsproblem als (nichtglattes) *Mehrzieloptimierungsproblem* aufgefasst, d.h. die Fehlerfunktion und der Regularisierungsterm werden gleichzeitig optimiert. Für den Fall, dass die ℓ_1-Norm als Regularisierungsterm verwendet wird, wird, basierend auf Resultaten aus dem Bereich der Mehrzieloptimierung, die Struktur der sogenannten *Pareto-kritischen Menge* analysiert, welche eine Obermenge des Regularisierungspfades ist. Außerdem wird ein Fortsetzungsverfahren zur Berechnung der kritischen Menge vorgestellt, das speziell auf dieses Problem zugeschnitten ist und die Nichtglattheit der ℓ_1-Norm berücksichtigt. Dieses wird anschließend genutzt, um anhand des Trainings neuronaler Netze zu zeigen, dass wesentliche Lösungen, die gut generalisieren, nicht im Regularisierungspfad enthalten sind und somit mit den Regularisierungsverfahren, welche klassischer Weise genutzt werden, nicht berechnet werden können. Außerdem wird diskutiert, inwieweit der vorgestellte Algorithmus für größere Probleminstanzen skaliert. Zum Abschluss wird ein Ausblick zu der Erweiterung auf allgemeinere Regularisierungsprobleme gegeben, bei denen der Regularisierungsterm lediglich stückweise glatt ist. Genauer gesagt, wird die Struktur der Pareto-kritischen Menge für diesen Fall analysiert, um die Grundlage für die Entwicklung neuer Fortsetzungsmethoden für diese Problemklasse zu schaffen.

Contents

1 | Introduction

Control problems appear in a wide variety of areas in our everyday lives as well as in technical applications. For instance, if we try to carry a full cup of coffee from A to B, we compensate for the movement of our body that leads to the coffee sloshing in the cup by making specific movements with the arm and hand to prevent the coffee from spilling over. When driving an autonomous car, it has to follow a predefined trajectory by controlling the angle of the steering axle and the velocity. Mathematically, such tasks are modeled as *optimal control problems*. To this end, the system dynamics is typically represented by a differential equation. For instance, the movement of the car can be modeled by an *ordinary differential equation* (ODE). To measure the performance of a certain control function $\mathbf{u} : [t_0, t_f] \to \mathbb{R}^{n_u}$, a *cost functional* $\mathcal{J}$ is defined. In the case of the autonomous car, this is typically the distance of the vehicle to the predefined trajectory. In total, optimal control problems are of the following form:

$$\begin{aligned} &\min_{\mathbf{u}} \mathcal{J}(\mathbf{y}, \mathbf{u}), \\ &\text{s.t.} \quad \dot{\mathbf{y}}(t) = g(t, \mathbf{y}(t), \mathbf{u}(t)), \quad \text{for } t \in [t_0, t_f], \\ &\qquad \mathbf{y}(t_0) = y_0, \end{aligned}$$

where $\mathbf{y} : [t_0, t_f] \to \mathbb{R}^{n_y}$ is the function of the system state. Problems of this kind can be solved with different numerical techniques, but they all have in common that they cannot react to changes in the physical system in real-time. If we consider the example from above of a human balancing a coffee cup using involuntary movements, they usually directly react to unforeseen changes, such as lightly bumping an unseen obstacle with the arm. Rather than planning the entire journey from A to B in advance, we decide in real-time how to move our hand to prevent the coffee from spilling over, allowing us to react better to unforeseen obstacles.

A similar methodology is used to solve control problems in practical applications. Based on observations or measurements of the current system state, a control input is determined and applied to the real system. Such approaches are called *feedback control* and account for noise and model errors. One of the most intuitive and also very well-working approaches is *model predictive control* (MPC) [GP17]. The basic idea of MPC is to divide the problem into smaller subproblems of shorter time horizons, i.e., instead of finding an optimal control for the whole task, an optimal control for the next few seconds is sought. Afterwards, the computed control input (or, more precisely, the first part of it) is applied to the real system. Then, the

initial state for the next subproblem is updated by observing the state of the real system, introducing a feedback loop. To apply MPC in real-time, the optimization over the short time horizon has to be done fast in comparison to the time scale of the system that should be controlled. Hence, MPC was initially applied to chemical processes modeled by linear systems, as these, on the one hand, typically exhibit slow dynamics and, on the other hand, can be optimized efficiently [CR80; Ric+78]. Today, MPC is a state-of-the-art tool in the industry. It is used in many different engineering disciplines, such as chemical process engineering, electrical engineering, aerospace engineering, or automotive engineering [For+15; Lee11; Sch+21]. The fact that the optimization can be done in real-time for these complex systems is mainly due to improved computational resources. Besides the practical applications showing the capability of MPC, there is also a fundamental mathematical theory that proves error bounds, stability, and robustness of MPC [GP17].

Traditionally, the model in MPC is based on differential equations. In recent years, however, data-based methods have gained increased attention in the control community [BK19; Sch+21], and are also used to identify the system dynamics and build surrogate models for MPC. The basic idea is that it is not necessary to have system knowledge, and it is enough to observe and collect data from the real system. This is specially amplified by the fact that through digitization and the possibilities to store ever larger amounts of data, it is very easy to access vast data sets to build such models. Machine learning is not only playing an increasingly important role in the area of control problems, but also found its way into almost every industrial and scientific field. For instance, methods from machine learning are applied in autonomous driving [BS21; Kir+22], in the development of smart homes [BCR09], in the health care sector [BK18], or in molecular and materials science [But+18]. Breakthroughs as IBM's Watson, which beat two champions in Jeopardy in 2011 [Fer12], and AlphaGo by DeepMind which became the first computer program to beat professional human Go players in 2016 [Sil+16; Sil+17], are certainly major motivators, in particular, as these works were extensively discussed in media. One of the best known model classes for machine learning are *(artificial) neural networks* (NNs) [GBC16]. These are inspired by the structure of human brain cells and are considered a key technology for recent successes in image and speech recognition, as well as artificial intelligence in general. For instance, NNs are at the core of AlphaGo. Neural networks are also used in the field of system identification. Typically, *recurrent neural networks* are used there, including *long short-term memory* (LSTM) networks [HS97], and *echo state networks* (ESN) [Jae10; JH04]. Moreover, to build surrogate models, regression-based frameworks for the identification of nonlinear dynamics [BPK16] or approximations of the *Koopman operator* [Bru+21; Koo31] are often used as well.

In this work, we consider different aspects of using machine learning models in MPC and address the *training* of these models, i.e., the fitting of the model to a given data set. Essentially, the thesis can be divided into two parts. In the first part, we study how data-based models can be used within the MPC framework. To this end, we begin by studying a flow control problem that we want to solve using MPC and an NN architecture. The results are promising but also leave open questions. For

example, the expensive data assimilation in the controlled case and the problem of *overfitting* the model to the data. This is the motivation for the remaining part of the thesis. Thus, subsequently, we analyze how autonomous surrogate models can be used to model the controlled system. The idea is to simplify the model building process and reduce the amount of data required to compute accurate models. In the second part of the thesis, we use tools from multiobjective optimization to analyze the training of NNs. More specifically, we consider regularized optimization problems that are often considered in machine learning training to avoid overfitting and analyze whether the perspective of multiobjective optimization helps us to gain deeper insights into these problems and find better solutions for the NNs. In the following, we discuss the contributions of this work in more detail.

Contributions and structure of the thesis

This thesis is settled mainly in the three areas of model predictive control, machine learning, and multiobjective optimization as summarized in Figure 1.1. More specifically, the work is related to the two intersection areas between machine learning and model predictive control and between machine learning and multiobjective optimization. Hence, Chapter 2, which introduces the theoretical basics needed within this thesis, includes an introduction to optimal control in general and MPC as a particular solution method to optimal control problems in practical applications (Section 2.1). Furthermore, an overview of machine learning with a specific focus on neural networks is given, and the Koopman operator and its numerical approximation via *extended dynamic mode decomposition* (eDMD) is introduced (Section 2.2). Finally, an introduction to multiobjective optimization, where also the case of nonsmooth functions is considered, is provided (Section 2.3).

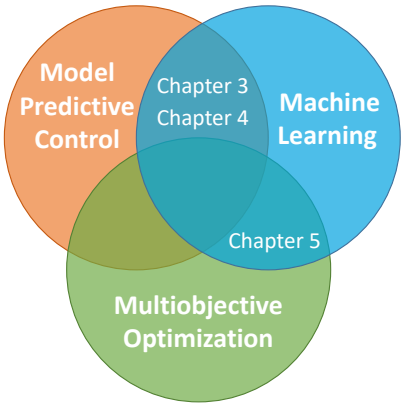


Figure 1.1: Relevance of the topics model predictive control, machine learning, and multiobjective optimization in the thesis.

First, Chapter 3 shows an example of using an NN as a surrogate model in the context of MPC. There, we are concerned with a flow control example, more specifically, the *fluidic pinball* [Den+18; Noa+16]. Flow control problems are usually very hard to solve since the system dynamics are described by nonlinear *partial differential equations* (PDEs) that are often expensive and time-consuming to solve numerically. The fluidic pinball is an academic example that exhibits chaotic behavior (in certain regimes) and is still relatively simple to simulate. It was explicitly constructed to benchmark new control algorithms. The NN architecture used to build a surrogate model for this system was already used before in MPC to control a robot cutting food [LKS15] and to control mode-locked lasers [BBK18]. The approach is based on sensor data, i.e., not the full flow field is observed but only some forces

induced by the flow of the fluid are available. Besides the fact that it is usually impossible to measure a full flow field in practical applications, it seems reasonable to reduce the problem dimension this way since humans and animals also (re-)act only based on partial observations. Returning to the example of carrying a cup of coffee, we cannot measure the movement of the coffee in the cup but only see the surface or only take the amplitude of the coffee into account to determine our movement. The numerical results derived in that chapter are very promising, and an outlook on an extension using online learning is given. This example shows that it is, in principle, possible to control complex dynamical systems utilizing NNs as surrogate models based on sensor data and extend the results obtained so far. However, many open questions and issues remain, which are discussed in Section 3.3. Some of the detected issues motivate the other two main chapters of this thesis.

One of the essential observations in Chapter 3 is that it is unclear how much data is needed to train the NN and how the training data should be sampled concerning the time-dependent control input. This question already arises when learning autonomous dynamical systems but becomes more complicated when adding a control input to the system. On the one hand, this is simply caused by the increased dimension of the system and hence of the input space of the surrogate model. A larger dimension typically requires models with more parameters, i.e., more degrees of freedom, resulting in more data needed to train the model. On the other hand, the system dynamics are usually no longer restricted to a low-dimensional manifold and exhibit a more complicated behavior. Furthermore, the ability to steer the system by applying different control inputs during data generation makes the sampling process more flexible and complex. Another issue dealing with controlled systems is that some techniques often used to build surrogate models for dynamical systems cannot directly be applied to control-dependent systems, for instance, approaches based on the Koopman operator. To overcome this issue, a framework is presented based on the restriction of the control set U to a finite-dimensional subset $V = \{u^1, \dots, u^m\} \subseteq U$ in Chapter 4. The idea is that autonomous surrogate models can be built for each of these finitely many control inputs V. Based on these various autonomous surrogate models, the real system can be approximated. A similar idea is followed in [PK19; POR20] for surrogate models approximating the Koopman operator. If a *full model* is utilized, i.e., a surrogate model which gets the control input as additional input, the optimization problem resulting from MPC is continuous. In contrast, the optimization problem induced by the autonomous models is discrete, as it is optimized over control inputs in V. Since such problems are much harder to solve, different solution strategies, besides solving the optimization problem directly, are introduced in Section 4.1. These utilize relaxation techniques based on the idea of the sum-up-rounding algorithm presented in [SBD12]. There, the algorithm was used to solve mixed-integer optimal control problems. Moreover, some of the theoretical results from [SBD12] are also utilized to derive error bounds for the different solution strategies in Section 4.2. Finally, numerical experiments for different dynamical systems are presented in Section 4.3, demonstrating the numerical applicability of the approach. Furthermore, in Section 4.4, additional experiments show that the approach is probably advantageous concerning the required amount of data.

Besides the fact that learning surrogate models for dynamical systems with control input is usually more challenging than training models for autonomous systems, another aspect observed in Chapter 3 is the phenomenon of overfitting. Meaning, the model is fitted too closely to the training data and is not able to generalize to unseen data outside the training set. For instance, think about noisy data, where the model should not include the information of the noise but only of the underlying system and hence should not be fitted in detail to the noisy training data. To overcome this issue, different techniques exist. One option, which is often utilized, is to add a regularization term to the loss function, i.e., problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + \lambda g(x)$$

are considered, where f is the original objective and g is the regularization term weighted with some $\lambda \in [0, \infty)$. Typical choices for g are different norms, for instance, the ℓ_1-norm or the (squared) ℓ_2-norm. In Chapter 5, this optimization problem is considered as a *multiobjective optimization problem* (MOP), i.e., f and g are interpreted as separate objectives that have to be minimized simultaneously:

$$\min_{x \in \mathbb{R}^n} \begin{pmatrix} f(x) \\ g(x) \end{pmatrix}.$$

In contrast to single objective optimization, in multiobjective optimization, it is not clear what optimal solutions to these problems are since there is no total order of the $\mathbb{R}^2$ (or more general $\mathbb{R}^k$). Solutions to these problems are defined as the optimal compromises, i.e., points where the objective value of the first objective can only be increased if the objective value of the other function is decreased or the other way around. The penalty approach coincides with the so-called *weighted-sum method* from multiobjective optimization. It is well-known that not all optimal compromises can be computed with this method if not both objectives are convex. Especially in the case of training NNs, the loss function is (highly) nonconvex. Hence, the question arises whether methods from multiobjective optimization can compute other solutions better suited to avoid overfitting or more efficiently than the penalty approach where different penalty parameters have to be tested. In Chapter 5, the idea is to use a *continuation method* to move along the *Pareto critical set* [Hil01]. The Pareto critical set consists of points that fulfill a first-order optimality condition, the so-called *KKT condition*. Essentially, continuation methods are based on the fact that the Pareto critical set satisfies a smoothness condition if the objectives are twice continuously differentiable and sufficiently regular. Since the ℓ_1-norm is nonsmooth, a new continuation method is developed that is specifically tailored to the ℓ_1-norm as the second objective, cf. Section 5.1. Afterwards, numerical examples, including the training of a small NN for an academic data set, are presented. Since in more practical settings NNs usually have many parameters, the resulting optimization problem is correspondingly high-dimensional. Thus, in Section 5.3, it is discussed why the presented method is not directly applicable to such high-dimensional problems and how it may be adapted. Besides regularization with smooth functions or the ℓ_1-norm, other choices are also implemented in other applications, for instance, the ℓ_∞-norm or total variation. To address regularization problems of these types,

in Section 5.4, a general analysis of the Pareto critical set is given, which may help to develop more general continuation methods.

At the end of this thesis, Chapter 6 summarizes the achieved results, and a detailed outlook on future work is given.

Large parts of this thesis have already been published in various scientific publications to which the author has made substantial contributions. This is indicated in more detail at the beginning of the respective chapters. Furthermore, the main contributions of this thesis (linked to the respective preceding publications) are summed up here:

- A proof of concept in Chapter 3: The presented NN architecture can be used to learn the essential dynamic behavior of the fluidic pinball and to control it via MPC.

  > The content of Chapter 3, except for the discussion in Section 3.3, was published before in [Bie+20], to which the author of this thesis was the main contributor.

- Development of the "QuaSiModO" framework to allow for the use of autonomous surrogate models (Section 4.1), including the proof of error bounds for the open-loop problem (Section 4.2) and the numerical evaluation of the approach for various numerical examples (Section 4.3 and Section 4.4).

  > Chapter 4 is based on [PB23], to which both authors contributed equally. The theoretical results concerning the error bounds and the numerical results presented in this thesis are mainly based on the work of the author of this thesis. Compared to the paper, some minor adjustments have been made: The error bound in Lemma 4.2.10 ((E2.a) in [PB23]) is slightly improved, and the error introduced by sum-up-rounding is stated in more detail. Moreover, the numerical example concerning the Mackey-Glass equation, cf. Section 4.3.2, is slightly adapted.

- Development of a continuation method for a biobjective MOP where one objective is smooth and the other one is the ℓ_1-norm (Section 5.1).

  > Chapter 5 is based on [BGP22], to which the author of this thesis was the main contributor. A brief analysis of the occurrence of kinks in the image of the Pareto set in Remark 5.1.5 and a discussion in Remark 5.1.14 on whether parts of the Pareto set can be missed during continuation have been added.

- Demonstration by two small NN examples that well-suited solutions, which avoid overfitting, are not achieved by the penalty approach but can be computed by the developed continuation method (Sections 5.2.3 and 5.3).

  > The content of Sections 5.2.3 and 5.3 was also published in [BGP22].

- Derivation of conditions for kinks in the Pareto critical set for a biobjective MOP where one objective is at least twice continuously differentiable and the other one is piecewise twice continuously differentiable (Section 5.4).

Section 5.4 summarizes the results already published in [GBP22]. The author of the thesis contributed substantially to this work.

2 | Theoretical Background

In this chapter, the main theoretical concepts used in this thesis are introduced. First, a short introduction to optimal control, in particular to model predictive control, is given in Section 2.1. Afterwards, in Section 2.2, we focus on different data-based techniques for building surrogate models that can be used in this framework. Particular attention is paid to neural networks since the surrogate models used primarily in this work belong to this category. Furthermore, neural networks are the main motivation for considering regularization paths and developing the continuation method tailored to the ℓ_1-norm, which is discussed in the last part of this thesis. As this part is based on theoretical results from multiobjective optimization, finally, the basics of this area are introduced in Section 2.3.

Since this thesis covers many different topics, this chapter introduces only the specific basics needed in this thesis. For more details, the reader is referred to the related literature in the respective sections.

2.1 Optimal Control and Model Predictive Control

This section aims to give a short overview of *optimal control* and *model predictive control* for nonlinear dynamical systems. In Section 2.1.1, a brief introduction to the overall concept of optimal control is given, including an overview of typical solution methods. Afterwards, as model predictive control is used to solve control problems in this thesis, it is introduced in Section 2.1.2. Finally, in Section 2.1.3, different options for the application of data-based methods and machine learning in the area of optimal control, especially in model predictive control, are discussed to give an idea of the current state of this area of research.

2.1.1 Optimal Control

This section is mainly based on [Bin+01; Lib12]. In [Bin+01], an overview of optimal control and possible solution strategies is given, whereas in [Lib12], a detailed introduction to the theoretical foundations of optimal control can be found.

In many practical applications, dynamical systems have to be controlled, i.e., according to a predefined task, the state of the system is steered such that a given cost function that models the control task is minimized. The dynamical systems considered in this thesis can be described by either *ordinary differential equations*

(ODE), *delay differential equations* (DDE), or *partial differential equations* (PDE). Regarding the latter, the system state y depends not only on the time t but also on the location x. In principle, by introducing a discretization in space, these systems can be approximated (to arbitrary accuracy) by high-dimensional ODEs by means of finite differences (FD) or the finite element method (FEM) as it is done, for instance, in the *method of lines* [SC63]. In a system described by a DDE – in contrast to an ODE – the derivative of the state depends not only on the current state but also on past states, often leading to much more complex dynamics than in ODEs. One way to solve DDEs is *Bellmann's method of steps*, where the DDE is transformed stepwise into multiple ODEs with different initial conditions [BZ03]. For this reason, and since we do not need any essential properties of DDEs or PDEs within this thesis, we restrict ourselves to ODEs unless otherwise stated, i.e., we consider the (nonlinear) dynamical system given by

$$\begin{aligned} \dot{\mathbf{y}}(t) &= g(t, \mathbf{y}(t), \mathbf{u}(t)), \quad \text{for } t \in [t_0, t_f], \\ \mathbf{y}(t_0) &= y_0, \end{aligned} \tag{2.1}$$

where $\mathbf{y}(t)$ is the *state* at time t, $\mathbf{u}(t)$ is the *control input* at time t and y_0 is the *initial state* at the *initial time* t_0. More precisely, $\mathbf{y} : [t_0, t_f] \to Y \subseteq \mathbb{R}^{n_y}$ is the state function, where $n_y \in \mathbb{N}$, and we assume that Y is connected and open. Furthermore, $\mathbf{u} : [t_0, t_f] \to U \subseteq \mathbb{R}^{n_u}$, with $n_u \in \mathbb{N}$, is called the control function, and the function $g : [t_0, t_f] \times Y \times U \to Y$ describes the system dynamics on the interval $[t_0, t_f]$ with a certain *final time* $t_f \in \mathbb{R}$. Note that we use bold print for the functions $\mathbf{y}$ and $\mathbf{u}$ to make it easier to distinguish them from elements $y \in Y$ and $u \in U$, i.e., from states and control inputs at certain times. Moreover, it should be noticed that in the area of dynamical systems, the system state of an ODE is usually denoted by x and not by y. However, since we sometimes refer to the location as x in examples with PDEs and use x as an arbitrary variable that is optimized in Chapter 5, the system state is denoted by y (or $\mathbf{y}$ for the state function) in this work to avoid confusion. Throughout the thesis, we always assume the local existence of a unique solution $\mathbf{y}$ of (2.1) for every control $\mathbf{u} : [t_0, t_f] \to U$ that is measurable and bounded. By a solution of (2.1) for such a control $\mathbf{u}$, we mean an absolutely continuous function $\mathbf{y}$ with

$$\mathbf{y}(t) = y_0 + \int_{t_0}^{t_f} g(t, \mathbf{y}(t), \mathbf{u}(t))\, dt.$$

One way to guarantee the existence of a unique solution of (2.1) is to ensure that the following conditions are all satisfied [Lib12, Section 3.3.1]:

- The function g is continuous in the first (t) and third argument (u) and continuously differentiable in the second argument (y).
- The partial derivative with respect to the state $\frac{\partial g}{\partial y}$ is continuous in t and u.
- The control function $\mathbf{u}$ is measurable and bounded, e.g., $\mathbf{u}$ may be piecewise continuous.

Note that, in this thesis, we usually consider systems that are not directly dependent on the time t but only indirectly through the control input $\mathbf{u}(t)$, i.e., we consider

systems given by $\dot{\mathbf{y}}(t) = g(\mathbf{y}(t), \mathbf{u}(t))$. This is not a strong limitation since many systems do not depend directly on time and ODEs where g depends on the time can be transformed into systems not depending directly on t by state augmentation, i.e., we consider the expanded state $\hat{\mathbf{y}} := (\mathbf{y}, t)$ with the system equations

$$\dot{\hat{\mathbf{y}}} = \begin{pmatrix} \dot{\mathbf{y}} \\ \dot{t} \end{pmatrix} = \begin{pmatrix} g(t, \mathbf{y}(t), \mathbf{u}(t)) \\ 1 \end{pmatrix} =: \hat{g}(\hat{\mathbf{y}}(t), \mathbf{u}(t)). \tag{2.2}$$

In order to still ensure the existence of a solution, in this case, g has to be continuously differentiable in t (or satisfy an appropriate Lipschitz condition) instead of just being continuous as stated above [Lib12, Section 3.3.1].

The actual control task is modeled via a cost functional $\mathcal{J}$ depending on the control function $\mathbf{u}$ and the resulting state function $\mathbf{y}$. A common form of such a functional is the *Bolza form* which is given by

$$\mathcal{J}(\mathbf{y}, \mathbf{u}) = \int_{t_0}^{t_f} C_L(t, \mathbf{y}(t), \mathbf{u}(t))\, dt + C_M(t_f, \mathbf{y}(t_f)), \tag{2.3}$$

where $C_L : [t_0, t_f] \times Y \times U \to \mathbb{R}$ is called the *Lagrange term* and $C_M : Y \to \mathbb{R}$ is referred to as the *Mayer term*. These two types of control terms can each be transformed into the other type by state transformations [Lib12, Section 3.3.2]. A typical control task, which is mainly considered in this thesis, is *trajectory tracking*, where an output trajectory of the system should be forced to follow a predefined *reference trajectory*. In this case, the Mayer term is omitted, and the Lagrange term is given by

$$C_L(t, \mathbf{y}(t), \mathbf{u}(t)) = \left\| h_{\mathsf{traj}}(\mathbf{y}(t)) - \mathbf{z}_{\mathsf{ref}}(t) \right\|^2,$$

where $h_{\mathsf{traj}} : Y \to Z \subseteq \mathbb{R}^{n_z}$ models the (observed) output trajectory of the system, $\mathbf{z}_{\mathsf{ref}} : [t_0, t_f] \to Z$ is the given reference trajectory, and $\|\cdot\|$ is some norm, typically the ℓ_2-norm. If h_{traj} is the identity (and $n_z = n_y$, $Z = Y$), the full state $\mathbf{y}$ of the system should follow a predefined trajectory. For instance, if the ODE describes the motion of a car, i.e., the state $y \in \mathbb{R}^2$ is the position of the car in the plane, the control task might be to let the car follow a predefined trajectory on the street. Another example could be the flow of a fluid around a cylinder, where the control task is usually not to control the whole state, i.e., the velocity and pressure of the fluid on a given spatial grid, but to steer the forces acting on the cylinder caused by the flow, cf. Section 3.2. In this case, h_{traj} maps the full flow field to these forces. Combining (2.1), where we omit the direct dependency on the time t, and (2.3) leads to the *optimal control problem*

$$\begin{aligned} \min_{\mathbf{u} \in \mathcal{U}} \mathcal{J}(\mathbf{y}, \mathbf{u}) &= \int_{t_0}^{t_f} C_L(t, \mathbf{y}(t), \mathbf{u}(t))\, dt + C_M(t_f, \mathbf{y}(t_f)), \\ \text{s.t.} \quad \dot{\mathbf{y}}(t) &= g(\mathbf{y}(t), \mathbf{u}(t)), \quad \text{for } t \in [t_0, t_f], \\ \mathbf{y}(t_0) &= y_0, \end{aligned} \tag{OCP}$$

where $\mathcal{U}$ is the set of measurable and bounded controls $\mathbf{u} : [t_0, t_f] \to U$. Note that we only optimize over $\mathbf{u} \in \mathcal{U}$ and consider the state function $\mathbf{y}$ not as an

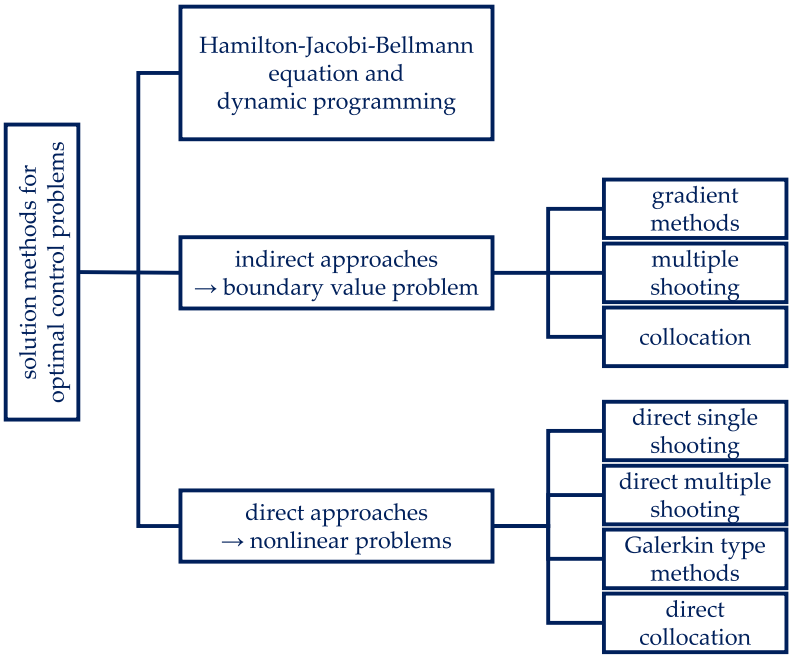


Figure 2.1: Classification of solution methods in optimal control according to [Bin+01].

optimization variable since it is uniquely determined via the system equations by the control function $\mathbf{u}$. Depending on the application, it might be necessary to consider additional constraints, e.g., a final point constraint of the form $c_f(\mathbf{y}(t_f)) = 0$ or path constraints given by $c_p(t, \mathbf{y}(t), \mathbf{u}(t)) \leq 0$. However, additional constraints are not considered in this thesis.

Necessary optimality conditions of a control $\mathbf{u}^*$ and the corresponding trajectory $\mathbf{y}^*$ can be achieved by means of calculus of variations and the maximum principle, cf. [Lib12, Chapter 4].

Solution methods Starting in the 1950s with the dynamic programming approaches presented by Bellmann [Bel10], various numerical solution methods were developed to solve optimal control problems. A detailed overview of existing methods and their classification can be found, for instance, in [Bin+01] or [Rao09] and is summarized in Figure 2.1. Besides *dynamic programming* and similar approaches that are also based on solving the *Hamilton-Jacobi-Carathéodory-Bellman* equation, cf. [Bel10], the solution methods are typically assigned to two major classes: *indirect* and *direct* methods.

Indirect methods refer to methods that aim at deriving the first-order optimality conditions from calculus of variation. This leads to a boundary-value problem which can be solved numerically via discretization. Hence, these approaches are also known as *optimize-then-discretize*. Numerical methods used to solve the discretized problem are, for instance, *gradient methods*, *multiple shooting*, or *collocation*. Although the calculated solutions are generally very accurate, the indirect approach requires a lot of knowledge and experience in optimal control to derive the necessary optimality

conditions and the boundary value problem in such a way that they can be solved numerically efficiently.

Direct methods avoid this issue by first discretizing the system resulting in a high-dimensional optimization problem which can be solved with state-of-the-art optimization techniques. Hence, these approaches are sometimes also referred to as *discretize-then-optimize*. Although direct methods usually deliver more inaccurate solutions than indirect methods [SB92], they have proven to perform more efficiently in practical large-scale applications [Bin+01]. To discretize the system and solve the underlying optimal control problem, there exist again various numerical methods, for instance, *direct single shooting*, *direct multiple shooting*, *collocation*, or *Galerkin-type methods*. An overview of direct solution methods can be found, for instance, in [Pyt99]. Here, direct single shooting, e.g., see [Kra85; Kra94], is briefly explained as this method is used within this thesis to solve the open-loop control problem in MPC, cf. Section 2.1.2.

The main idea of direct single shooting is to introduce a discrete approximation of the control $\mathbf{u}$ and take the state $\mathbf{y}$ as a dependent variable of this representation since it is determined by the system equations (2.1). Via numerical integration methods, the trajectory for a fixed discretized control can be computed, representing the function that maps the discrete representation to the resulting system state. Usually, a discrete approximation $\tilde{\mathbf{u}}$ of the control variable $\mathbf{u}$ is derived by taking constant values $u_i \in U$ over the time horizons $[t_i, t_{i+1}]$, where the fixed time grid $t_0 < t_1 < \cdots < t_{n_t} = t_f$, $n_t \in \mathbb{N}$, is given, i.e., we introduce the approximated control by

$$\tilde{\mathbf{u}}(u_{0:n_t-1}, t) = u_i \quad \text{for } t \in [t_i, t_{i+1}) \text{ and } i \in \{0, \dots, n_t - 1\},$$

where $u_{0:n_t-1} = (u_i)_{i=0,\dots,n_t-1} \in U^{n_t}$ denotes the discrete representation of the control. This results in the nonlinear optimization problem

$$\min_{u_{0:n_t-1} \in U^{n_t}} \int_{t_0}^{t_f} C_L(t, \mathbf{y}(u_{0:n_t-1}, t), \tilde{\mathbf{u}}(u_{0:n_t-1}, t))\, dt + C_M(t_f, \mathbf{y}(u_{0:n_t-1}, t_f)),$$

where the dependent state $\mathbf{y}(u_{0:n_t-1}, t)$ is defined by the system equations (2.1) with control input $\tilde{\mathbf{u}}(u_{0:n_t-1}, t)$. During the optimization, a state-of-the-art integration method, e.g., a *Runge-Kutta* method, can be used to derive an approximation of the system state in each iteration. For an introduction to integration methods, see, for example, [AP98]. The resulting (finite-dimensional) nonlinear optimization problem can be solved with standard optimization techniques, e.g., *sequential quadratic programming* (SQP), *interior point*, or *trust-region* methods [NW06].

Discrete control problem As during computation, i.e., during the numerical integration in the direct shooting method, only discrete time steps can be realized, we mostly consider the system and the optimal control problem in discrete time. Note that in general, the time grid for the discretization of the state $\mathbf{y}$ and the control $\mathbf{u}$ considered in the direct single shooting method does not have to be the same, as the latter may be chosen coarser to reduce the dimension n_t of the optimization

problem. Nevertheless, to simplify the notation, we assume the same time grid for both. More precisely, an equidistant time grid $t_0 < t_1 < \cdots < t_{n_t} = t_f$, $n_t \in \mathbb{N}$, with $t_{i+1} - t_i = \Delta t$ for all $i \in \{0, \dots, n_t - 1\}$, is considered. By assuming a constant control over multiple time steps, the coarser grid for the control in the direct single shooting can be achieved. The equidistant time grid allows for introducing the *time-T-map* $\Phi : Y \times U \to Y$ that represents the evolution of the dynamical system during a fixed time Δt, i.e., for $y_i = \mathbf{y}(t_i)$ and $i \in \{0, \dots, n_t - 1\}$, it holds

$$y_{i+1} = \Phi(y_i, u_i),$$

where

$$\Phi(y_i, u_i) = y_i + \int_{t_i}^{t_{i+1}} g(\mathbf{y}(t), u_i)\, dt.$$

For the discrete version of the optimal control problem, the objective has to be adapted as well. In particular, the integral over the Lagrange term has to be approximated, for instance, by means of quadrature rules. More efficiently, state augmentation of the state $\mathbf{y}$ and C_l can be used to integrate the objective at the same time as the system, cf. [GP17, Section 11.4]. However, since we are looking at the trajectory tracking error in this thesis, it is also reasonable to determine the error at the discrete time steps and simply sum them up, resulting in the discrete control problem

$$\begin{aligned} \min_{u_{0:n_t-1} \in U^{n_t}} J(y_{1:n_t}, u_{0:n_t-1}) &= \sum_{i=0}^{n_t-1} P(t_{i+1}, y_{i+1}, u_i), \\ \text{s.t.} \quad y_{i+1} &= \Phi(y_i, u_i), \quad \text{for } i \in \{0, 1, \dots, n_t - 1\}, \end{aligned} \tag{2.4}$$

where $y_{1:n_t} = (y_i)_{i=1,\dots,n_t} \in Y^{n_t}$ and $u_{0:n_t-1} = (u_i)_{i=0,\dots,n_t-1} \in U^{n_t}$ are the discretized state and control. The objective $P : [t_0, t_f] \times Y \times U \to \mathbb{R}$ may represent the tracking error, i.e., $P(t_{i+1}, y_{i+1}, u_i) = \|h_{\mathsf{traj}}(y_{i+1}) - z_{\mathsf{ref}}(t_{i+1})\|^2$. Furthermore, it is a common choice to add a penalty term for the control u_i, i.e., the objective is given by $P(t_{i+1}, y_{i+1}, u_i) = \|h_{\mathsf{traj}}(y_{i+1}) - z_{\mathsf{ref}}(t_{i+1})\|^2 + \lambda \|u_i\|$ with $\lambda > 0$. This is often motivated by the need to avoid high energy consumption, which is usually associated with large control inputs, but also by the fact that the control problem is often easier to solve numerically when adding a penalty term [GP17].

2.1.2 Model Predictive Control

Until now, we considered so-called *open-loop* solution strategies, i.e., strategies where the optimal control problem is solved over the entire time interval $[t_0, t_f]$ without feedback from the system. Mathematically these methods are able to solve the control problem exactly (up to the discretization error, which might sum up over time). However, in practical applications, perturbations of the system state may occur, or the differential equations representing the real system might be inaccurate, which cannot be taken into account by these approaches. Therefore, it is useful to combine the open-loop strategies with a feedback loop, resulting in *model predictive control* (MPC). A comprehensive introduction to MPC can be found in [GP17].

The basic idea of MPC is to solve the open-loop problem on a finite, smaller time horizon of p time steps, i.e., over the so-called *prediction horizon* t_p, using a model of the real system. Afterwards, only the first optimized control u_0 is applied to the real system, and based on the resulting state y_1 of the real system, the optimization for the next p time steps is performed. Hence, in MPC, the following optimization problem, which is quite similar to (2.4), has to be solved for each time step t_k:

$$\begin{aligned} \min_{u_{k:k+p-1} \in U^p} J_k(\tilde{y}_{k+1:k+p}, u_{k:k+p-1}) &= \sum_{i=k}^{k+p-1} P(t_{i+1}, \tilde{y}_{i+1}, u_i), \\ \text{s.t.} \quad \tilde{y}_{i+1} &= \Phi^r(\tilde{y}_i, u_i), \quad \text{for } i \in \{k, \dots, k+p-1\}, \end{aligned} \tag{MPC$_k$}$$

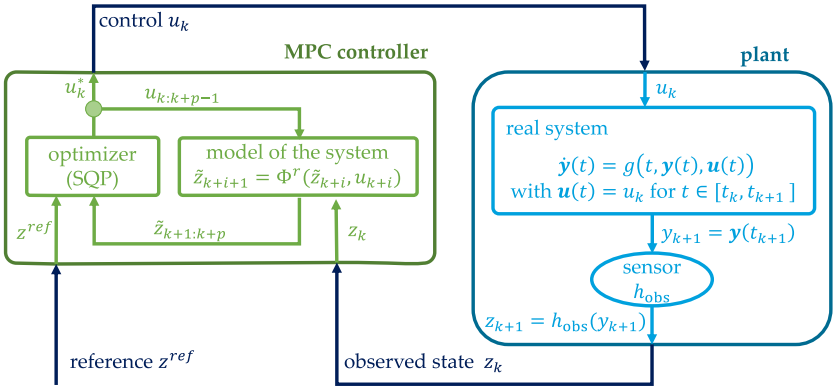
where $u_{k:k+p-1}$ and $\tilde{y}_{k+1:k+p}$ denote the vector of the corresponding control inputs and system states, respectively, i.e.,

$$u_{k:k+p-1} = (u_i)_{i=k,\dots,k+p-1} \quad \text{and} \quad \tilde{y}_{k+1:k+p} = (\tilde{y}_i)_{i=k+1,\dots,k+p}.$$

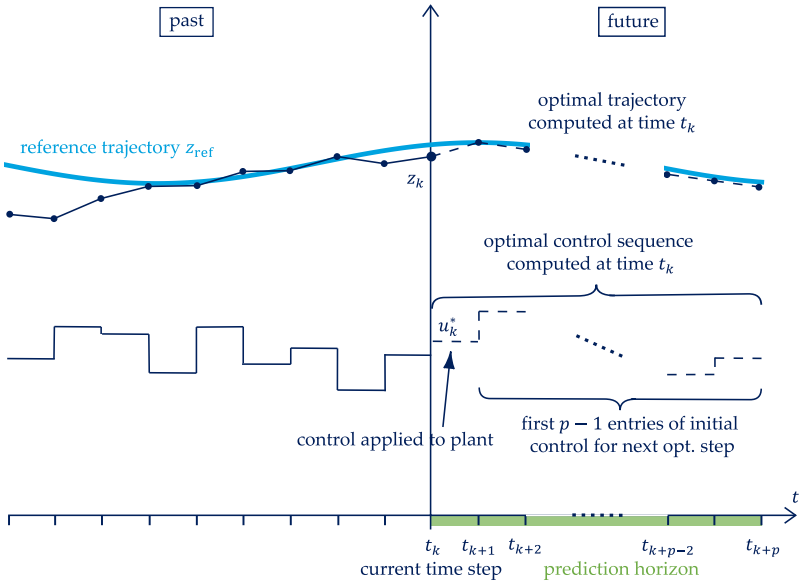
Similar to the previous section, we use this type of notation to denote vectors of system variables which depend on the time steps throughout the thesis. Moreover, to account for the inaccuracy of the model, the approximation of the real system state y_i at time t_i by the model is denoted by $\tilde{y}_i$, and Φ^r represents the equations of the model of the discrete system dynamics approximating the dynamics of the real system described by Φ, i.e., $\Phi^r(y_i, u_i) \approx \Phi(y_i, u_i) = y_{i+1}$. The model might be given, for instance, by differential equations describing the system and solved via numerical integration as in the previous section. Moreover, P is the discrete objective as above, cf. (2.4), and the initial value is given by the true state of the system, i.e., $\tilde{y}_k = y_k$.

A visualization of the concept of MPC is presented in Figure 2.2. As the time horizon over which is optimized is moved forward by one time step after applying the first control input u_k to the system, this method is sometimes also referred to as *moving horizon* or *receding horizon control*. The most important advances of MPC are the straightforward implementation and the possibility to easily include constraints on the control or the system state. Although linear models are faster to optimize, MPC allows for using nonlinear models, which can be beneficial in complex applications. Furthermore, stability results and robustness against the model error of the closed-loop problem can be proven, see [GP17].

The most crucial part of MPC to allow for real-time applicability is undoubtedly the model of the system since it should be accurate enough to achieve good control performance on the one hand and enable a computationally fast optimization on the other hand, i.e., it should be quick to evaluate and may provide gradient and Hessian information easily. In the setting described above, it is assumed that the optimization step at time t_k can be performed in zero time. This is obviously unrealistic but eases the notation and the theoretical considerations. By approximating the initial condition $\tilde{y}_k$ already at time t_{k-1}, i.e., $\tilde{y}_k = \Phi^r(y_{k-1}, u_{k-1})$, the optimization step for solving (MPC$_k$) can and has to be performed during one time step, i.e., it has to be solved in time Δt such that the computed control u_k can be applied to the system at time t_k. Hence, MPC was first introduced for linear



(a)



(b)

Figure 2.2: Visualization of the basic concepts of MPC. In (a), the interaction between the *plant*, which consists of the real system represented by an ODE and the sensor, and the MPC controller, consisting of the optimizer and the model of the system, is shown. In (b), the concept of the shifting horizon with respect to time is visualized, and the shift method for the initialization is outlined.

systems in the area of chemical processes as these typically exhibit slow dynamics [CR80; Ric+78]. Nevertheless, due to more powerful computing technologies and advances in model building, nowadays, MPC is successfully applied in many different areas and is a state-of-the-art tool in industry. For an overview of different MPC approaches and industrial applications, see, for instance, the surveys [For+15; Lee11; Sch+21]. Besides the modeling via physical equations, data-based methods and machine learning models have recently received high attention [BK19; Sch+21]. This is further discussed in Section 2.1.3, and some of those modeling techniques are presented in Section 2.2. Apart from appropriate modeling, another way to allow for real-time application is to apply not only one but multiple control steps to the real system after optimization and to use the feedback not in every time step but only after the *control horizon* t_c. Thus, the optimization can be performed during the time t_c instead of in only one time step of size Δt.

Besides the fact that the duration of the evaluation of the model must not exceed the specified time limit, in many practical applications, the full state y is not available in every time step. Often it is even impossible to measure it with the help of sensors at all. For instance, in a flow control problem, it is unrealistic to measure the velocity and pressure of the fluid on a fine spatial grid. Instead, the velocity may only be accessed at single sensors, or it is only possible to measure certain forces induced by the flow, cf. Section 3.2. Hence, it is necessary that the model only acts on the given sensor data that is available. To this end, we introduce the *observable* $h_{\mathsf{obs}} : Y \to Z \subseteq \mathbb{R}^{n_z}$, $n_z \in \mathbb{N}$, mapping a state y to the observed state z available via the sensor data and assume that the surrogate model is given by $\Phi^r : Z \times U \to Z$ with

$$z_{i+1} \approx \tilde{z}_{i+1} = \Phi^r(z_i, u_i), \quad \text{for } i \in \{0, 1, \dots, n_t - 1\},$$

where $z_i = h_{\mathsf{obs}}(y_i)$ for all $i \in \{0, \dots, n_t\}$. Obviously, an observable h_{obs} has to meet certain requirements to remain capable of fulfilling the control task. First of all, it is necessary that the objective function can still be evaluated based on the given data to rate the control performance, i.e., we assume that there exists P_{obs} with

$$P(t, y, u) = P_{\mathsf{obs}}(t, h_{\mathsf{obs}}(y), u) \quad \forall t \in [t_0, t_f], y \in Y \text{ and } u \in U. \tag{2.5}$$

For practical applications, this is usually not a limitation, as the performance of the system has to be assessable based on the sensor data. Otherwise, it would not be possible to rate the control performance. In the case of trajectory tracking, we may simply assume that $h_{\mathsf{traj}} = h_{\mathsf{obs}}$. The assumption stated in (2.5) allows us to reformulate (MPC_k) as follows:

$$\begin{aligned} \min_{u_{k:k+p-1} \in U^p} J_{\mathsf{obs},k}(\tilde{z}_{k+1:k+p}, u_{k:k+p-1}) = \sum_{i=k}^{k+p-1} P_{\mathsf{obs}}(t_{i+1}, \tilde{z}_{i+1}, u_i), \\ \text{s.t.} \quad \tilde{z}_{i+1} = \Phi^r(\tilde{z}_i, u_i), \quad \text{for } i \in \{k, \dots, k+p-1\}, \end{aligned} \tag{$\mathrm{MPC}_k^{\mathsf{obs}}$}$$

where the initial condition is given by the measurement of the real system at time t_k, i.e., $\tilde{z}_k = z_k = h_{\mathsf{obs}}(y_k)$. Furthermore, the observations should be chosen in such a way that it is – at least theoretically – possible to predict the effect of the control

input on the observed values, i.e., the observation should be able to capture the essential dynamics of the system. This assumption is strongly related to the concept of *observability* in control theory, where the full state should be reconstructible from the observed state [GK71; Kal60a]. A possibility to realize this property with fewer sensor measurements is to use *delay coordinates*, meaning that not only the current state is observed but a time series of states [Lju98; PA16]. This concept is also used in this thesis, cf. Section 3.1. From the perspective of dynamical system theory, this is motivated by Takens' embedding theorem [Tak81], which states that the attractor of a dynamical system can be reconstructed with the help of delay coordinates. In this thesis, the focus does not lie on the choice of the observable h_{obs}. Therefore, we leave it at this informal explanation and assume that h_{obs} is appropriately chosen, i.e., the objective can be evaluated based on the observed state z, cf. (2.5), and it is possible to predict the influence of the control input u based on the observed state of the previous time step, which may be realized by means of delay coordinates.

Another important point concerns the initialization of the control variable in the optimization step. A very intuitive and numerically reasonable way is the *shift method*, meaning that the optimized open-loop control of the last time step is shifted by one. Hence, only a value for the previous initial control input is needed, which can simply be obtained by copying the second to last control input. To make this more precise, we assume that $u^*_{k:k+p-1} = (u^*_k, u^*_{k+1}, \dots, u^*_{k+p-1})$ is the optimal solution of (MPC_k) or ($\mathrm{MPC}_k^{\mathsf{obs}}$), respectively. For the initialization of the control in the next time step, i.e., at time t_{k+1}, we then take

$$u_{k+1:k+p} = (u_{k+1}, \dots, u_{k+p}) = (u^*_{k+1}, u^*_{k+2} \dots, u^*_{k+p-2}, u^*_{k+p-1}, u^*_{k+p-1}).$$

For more advanced initialization techniques, see [GP17, Section 12.5].

2.1.3 Data-Based Methods and Control

In this section, we want to give a brief overview of how data-based methods can be incorporated into optimal control and, more specifically, into MPC. Employing data-based methods is an active field of research, recently fueled by the potential provided by machine learning methods. For a more comprehensive introduction to this field, we refer to [BK19, Chapter 10] and [DBN17, Chapter 2].

From the previous section, arguably, the most intuitive application of data-based methods in optimal control is the task of *system identification*, i.e., the design of models representing the system dynamics. These models can then be used as the model Φ^r in the MPC framework, cf. Section 2.1.2. To accomplish the task of system identification, various approaches have been devised, ranging from more traditional methods to modern machine learning techniques. For instance, there are methods based on *reduced order modeling*, like *proper orthogonal decomposition* (POD) [KV99; MOO13]. In recent years, surrogate models based on the *Koopman operator*, e.g., the approximation via *extended dynamic mode decomposition* (eDMD) [AKM18; KM18a; PK19; POR20; ŠIM21], became quite popular. Furthermore, classical machine learning approaches have just entered the field as well. One of the most famous approaches in this area is probably the use of neural networks.

For instance, *feed-forward neural networks* (FNN) [DER95; SI18] and *recurrent neural networks* (RNN) [BBK18; Bie+20], including *long short-term memory* (LSTM) [Igl+18] and *echo state networks* (ESN) [Arm+19; Jor+18], were used as models in MPC. Indeed, despite MPC, data-based methods for system identification to develop controllers have a long-lasting tradition. One of the first approaches in this direction is probably the work by Kalman [Kal60b; WB95], where the state is estimated from measurements of sensors and the control. More details on data-based surrogate modeling can be found in Section 2.2, as some of the mentioned techniques are used later in this thesis to build surrogate models in the MPC framework.

Another way to use machine learning to solve control problems is to directly learn control laws, called *machine learning control* (MLC) [DBN17]. This might be beneficial as an appropriate control law can often be delivered easier, whereas the dynamics of the system may be hard to capture. One approach that falls into this category is *reinforcement learning* [SB18]. The basic idea of reinforcement learning is that an appropriate control law is learned by the interaction with the system over time by using a *value function* that somehow describes the success or failure of the current control [BK19; Hes+18; KBP13; LVV12]. Another approach is followed in [DBN17]. There, genetic algorithms are used to find an optimal control law based on a predefined class of possible controllers. For the optimization with the genetic algorithm, different control laws are tested in a (systematic) trial-and-error fashion. This way, for instance, the parameters of more traditional approaches, such as a *proportional-integral-derivative* (PID) controller, can be derived or a neural network can be evolved to be used as a controller. A drawback of this approach is that relatively many control laws need to be tested before a suitable one is found, which might be – depending on the system and the control task – computationally expensive and unacceptable.

A natural approach to further improve the outlined methods is to integrate already existing physical or chemical system knowledge to combine the best of both worlds. For instance, in MLC, this can be done by choosing an appropriate class of control laws. In system identification, the expertise might be used to reduce the amount of data needed to train the model, thus fastening the training. Another possibility is to replace parts of the model with known system equations or to ensure the preservation of symmetries or the energy in the system during learning, see, for instance, [DF21; OO23; Rid+21]. By using system knowledge, in general, the accuracy and the generalization of the induced model can often be improved. A comprehensive introduction to this field, called *physics-informed machine learning*, is given in [Kar+21].

2.2 Data-Based Surrogate Modeling

The main focus of this thesis lies on data-based surrogate modeling. Due to the enormous progress in the field of machine learning and, probably more importantly, the availability of a large amount of data, data-based methods have become very popular in various industrial and scientific areas over the last decades. For instance, as discussed briefly in the previous section, there are many possibilities to utilize

data-based techniques to solve optimal control problems. In this section, we first outline the basic concepts of machine learning in Section 2.2.1. Afterwards, in Section 2.2.2, an introduction to (artificial) neural networks is given since they represent one of the most popular model classes in machine learning and serve as the main motivation for the second part of this thesis, where we consider regularization paths. Finally, the approximation of the Koopman operator via *extended dynamic mode decomposition* (eDMD) is explained in Section 2.2.3 since this technique is used several times in this work to create surrogate models. Note that data-based methods, as the approximation of the Koopman operator, are not necessarily classified as a machine learning approach. Nevertheless, within this thesis, we do not distinguish between data-based methods and machine learning and use the terms interchangeably as it makes no significant difference for our purposes.

2.2.1 The Basics of Machine Learning

In this section, we want to clarify what we mean when we say that we "*learn* (something) *from data*" and introduce the basic concepts of machine learning, i.e., we introduce the general learning problem considered in this thesis. For a more comprehensive introduction to this topic, the reader is referred to [AML12; Bis06].

The underlying goal in machine learning is to approximate a *target function*

$$t : \mathcal{A} \to \mathcal{B},$$

which maps instances a from the *input space* $\mathcal{A}$ to a result $b = t(a)$ in the *output space* $\mathcal{B}$. For example, the aim might be to find a model for the time-T-map $\Phi : Y \times U \to Y$ of a dynamical system with control input. In this case, the input space would be given by $\mathcal{A} = Y \times U \subseteq \mathbb{R}^{n_y} \times \mathbb{R}^{n_u}$ and the output space by $\mathcal{B} = Y \subseteq \mathbb{R}^{n_y}$. To learn the target function t, i.e., to find a good approximation thereof, a set of possible *hypotheses* $\mathcal{H}$ is considered and, based on a given data set

$$D = ((a^1, b^1), \dots, (a^N, b^N)) \subseteq \mathcal{D} := \mathcal{A} \times \mathcal{B},$$

the hypothesis which fits best (as specified below) is chosen. Typically, the set of possible hypotheses is implicitly given by a parameterized function

$$m : \mathcal{X} \times \mathcal{A} \to \mathcal{B},$$

where $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the *parameter space*. An instance $m_x : \mathcal{A} \to \mathcal{B}$, $m_x(a) = m(x, a)$ for a fixed parameter $x \in \mathcal{X}$ is referred to as a *model*, and the set of all possible models or the function m is often referred to as the *model class*. Note that we use $x \in \mathcal{X}$ for the parameters of the model, which is rather uncommon in the area of machine learning. However, our viewpoint is more from the optimization perspective, where the variable with respect to which we can optimize is usually referred to as x. Now, the aim is to utilize the information given by the data set D to find parameters such that the resulting model approximates t well. This process is usually referred to as *training*.

In order to find a hypothesis that approximates the target function t well, a measure for the quality of a hypothesis is needed, i.e., some measure to determine the distance between a hypothesis $h \in \mathcal{H}$ and the target t. How this measure should be chosen depends crucially on the considered task. Typically, the mean value over the pointwise errors of the data points in D is considered, i.e.,

$$L_D : \mathcal{X} \to [0, \infty), \quad L_D(x) = \frac{1}{|D|} \sum_{(a,b) \in D} d(b, m(x, a))$$

where $d : \mathcal{A} \times \mathcal{B} \to [0, \infty)$ is some pointwise error measure. The function L_D is usually referred to as the *loss function* or simply the *loss*. Thus, the training can be considered as an optimization problem in the parameter space $\mathcal{X}$ for the loss function defined by a chosen model class m and the fixed data set D. Considering the example of learning the time-T-map of a controlled system, the data points would be of the form $(a^i, b^i) = ((y_i, u_i), y_{i+1})$ with $y_{i+1} = \Phi(y_i, u_i)$ and the pointwise error might be defined as the *least-squares error*, i.e.,

$$d : Y \times Y \to [0, \infty), \quad d(y, \tilde{y}) = \|y - \tilde{y}\|_2^2 .$$

Note that instead of approximating a (deterministic) target function, due to the presence of noise in real-world data, one often aims to learn a probability distribution as a target. The deterministic case can be seen as a special case of this more general setting. However, within this thesis, we restrict ourselves to the deterministic case since the problems we consider are not influenced by noise.

Different types of learning Typically, learning problems are grouped into three classes, namely, *supervised*, *unsupervised* and *reinforcement learning*. The learning problem described so far is the one that arises in supervised learning. The aim is to learn the target function t based on given *labeled* data, i.e., we have data points (a^i, b^i) where b^i denotes the correct outcome of $t(a^i)$. In this thesis, we merely consider this type of learning problem. Depending on the output space $\mathcal{B}$, these are typically further divided into different groups. The best known are *regression* and *classification* tasks.

If we assign a continuous value to the instances, i.e., if the target t is a function that maps to $\mathbb{R}^m$, $m \in \mathbb{N}$, we speak about regression. An example of a regression task is to learn a model of the time-T-map Φ. As explained above, the least-squares error is a typical loss function for this purpose.

In classification tasks, $\mathcal{B}$ consists of a finite number of discrete categories, usually labels or integers representing certain categories. For instance, one of the simplest benchmark data sets for classification is the Iris data set [Fis36]. The task is to classify the input data, which consists of four real-valued inputs (length and width of the sepals and petals of iris flowers), i.e., $\mathcal{A} \subseteq \mathbb{R}^4$, into three categories $\mathcal{B} = \{\text{“setosa”}, \text{“versicolor”}, \text{“virginica”}\}$. In order to make it easier to handle the set numerically, the labels are transformed into vectors of size $|\mathcal{B}|$ representing whether the instance belongs to one of the classes or not. For instance, instead of the data point $(a, \text{“setosa”})$, we would consider $(a, (1, 0, 0))$. More precisely, we replace the

output set $\mathcal{B}$ by $\hat{\mathcal{B}} = \{b \in [0,1]^{|\mathcal{B}|} : \sum_{c=1}^{|\mathcal{B}|} b_c = 1\}$. This way, the output of the model can be interpreted as a probability vector whose components indicate to which class an input likely belongs. Furthermore, this allows for applying the *cross-entropy loss*, which is frequently used as a loss function in classification tasks. It is given by

$$L_D(x) = -\frac{1}{|D|} \sum_{(a,b)\in D} \sum_{c=1}^{|\mathcal{B}|} b_c \log(m(x,a)_c), \tag{2.6}$$

where the subscript c indicates the c-th entry of the corresponding vector.

Until now, we assumed that the whole data set D is known beforehand, i.e., during the entire training process. Two slightly different settings are *active learning* [Set12] and *online learning* [Hoi+21]. Online learning means that new data points are created "online" while the model is already used for prediction, i.e., not all data points are available at the same time, but more and more information comes up during the training, which is done in parallel to the application of the current model. In active learning, the labels of the data points are not known beforehand but can be queried during the training in a targeted manner. This setting is usually considered when it is expensive to create the labels of the data points to reduce the cost of the generation and maximize the informativeness of the data simultaneously.

In other situations, the correct result for a particular input may not be accessible or cannot be used directly for training. Recall the example from the introduction, where a person wants to transport a cup of coffee from A to B. One possibility to solve the control problem would be to learn a model of the time-T-map Φ, i.e., to model the learning task as a supervised learning problem and choose the appropriate control based on the learned model. But we are probably unable to access detailed information on the movement of the coffee inside the cup to prove our model outcome and adapt it appropriately. Another possibility is that the person tries out different movements, i.e., different control inputs, and learns which movements are helpful and which are not, based on some kind of performance measure, e.g., the maximal height of the coffee in the cup, i.e., the control law is learned directly. In a similar way, reinforcement learning is often applied to control tasks, see, for instance, [BK19; Hes+18; KBP13; LVV12]. Usually, the cost function J, cf. (2.4), can be used as a feedback function to measure the control performance (and to optimize the model parameters with respect to it). Another typical application of reinforcement learning is the development of artificial players, e.g., for the game of Go. For a supervised learning setting, the perfect move – or at least a good move that a professional player would make – has to be known. Since this information is usually hard to access, reinforcement learning is a good way to go, as it is easier to assign a score to a move at a particular stage of the game. Thus, reinforcement learning is also the basis for the famous AlphaGo by DeepMind, which was able to beat professional human Go players [Sil+16; Sil+17].

In contrast to supervised and reinforcement learning, unsupervised learning can be seen as learning without feedback. There, the task is to find patterns or structural results in given unlabeled data $(a^1, \dots, a^N)$. A typical task that falls into this

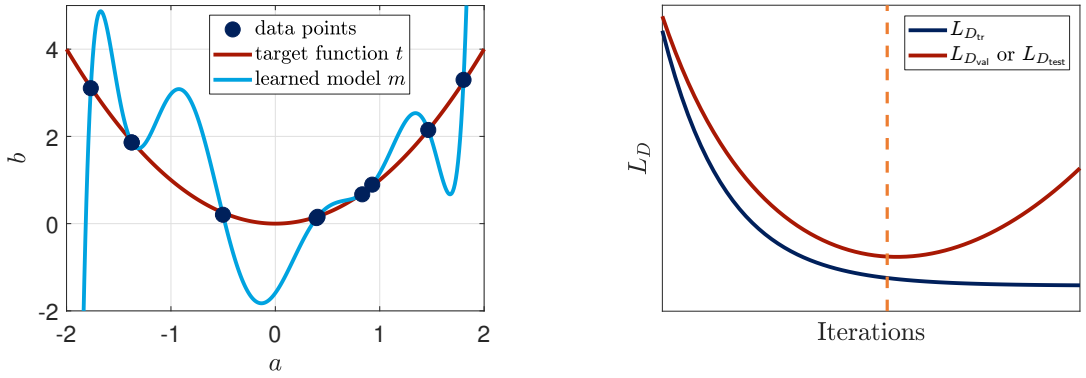
category is clustering. Unsupervised learning can be a stand-alone technique or used as a preprocessing step in supervised learning.

Generalization to points outside the data set From a theoretical perspective, one of the first questions when considering the introduced learning setting is whether it is really possible to approximate the function t outside the given data set D or to give an error bound. The answer is obviously "no" if we assume that there is no information about the target function t available, which is the underlying assumption. Nevertheless, if we consider the problem from a probabilistic viewpoint, it is possible to show certain bounds on the (probabilistic) error if we assume that the data is sampled independently according to a probability distribution. For a detailed discussion, we refer to [AML12]. Here, we leave it that abstract and introduce a concept allowing us to estimate the performance of a trained model outside the given data set D in practice instead.

Training, test and validation set To evaluate the model performance outside the data used for the training, an independent data set can be used. Therefore, we distinguish between the *training set* $D_{\mathsf{tr}} \subseteq \mathcal{D}$ and the *test set* $D_{\mathsf{test}} \subseteq \mathcal{D}$ which have to be sampled independently and only the training data set D_{tr} is allowed to be used during the training. Indeed, it is possible to show that if the test set is large enough, the error on the test set gives a good approximation of the so-called generalization error, i.e., (loosely speaking) the error of the model outside the training data set D_{tr}. If we assume that we only have the data set D available and have to divide it into D_{tr} and D_{test}, the trade-off between training performance, which usually gets better by increasing the size of the training set, and the ability to estimate the generalization error, which needs a large test set, has to be taken into account.

In addition to the training and test set, sometimes a third (independent) data set is introduced, the so-called *validation set* $D_{\mathsf{val}} \subseteq \mathcal{D}$. It is used to estimate the ability of the model to generalize to unseen data during the training, i.e., it is used similar to the test set but may influence the training. Typically, a validation data set is used to select a model from multiple trained models. For instance, a linear and a nonlinear model are trained and one wants to decide which of them generalizes better to unseen data. Therefore, the error on the validation data set is calculated. As the validation set is used during training or for selecting a model, it cannot be used to estimate the generalization performance, i.e., the validation data set has to be collected independently of the test set.

Overfitting and regularization Another way to use the validation set, typically used in neural network training where the training is done iteratively (by applying a gradient descent method to solve the optimization problem), is to observe the error on the validation set and stop the training as soon as the validation error $L_{D_{\mathsf{val}}}$ increases. This strategy is usually referred to as *early stopping*. The idea behind it is that an increasing validation error indicates that the model performance outside the training data set gets worse, although the training error $L_{D_{\mathsf{tr}}}$ still decreases. A typical learning curve showing this behavior is presented in Figure 2.3b. If the error



(a) (b)

Figure 2.3: (a) Example of overfitting for the target function $t : \mathbb{R} \to \mathbb{R}, \quad t(a) = a^2$. (b) Example curves of training and test error during iterative training. The point where overfitting starts is marked by the orange dashed line.

on the training data set is small and the error on the validation (or test) set is huge, the model is fitted too close to the training data. This phenomenon is also known as *overfitting*.

To overcome this issue, different techniques are used. Besides early stopping, a common idea is adding a *regularization term* (also called *penalty term*) to the loss-function, i.e., instead of minimizing $L_{D_{\text{train}}}$,

$$\min_{x \in \mathcal{X}} L_{D_{\text{train}}}(x) + \lambda \Omega(x)$$

is solved, where $\Omega : \mathcal{X} \to \mathbb{R}$ and $\lambda \in [0, \infty)$. Obviously, the choice of Ω crucially influences the result, and the (optimal) choice of Ω depends (again) on the concrete learning problem. Typically, Ω is a positive function, for instance, a norm. The basic idea behind this regularization approach is to get the simplest model that still fits the data since these models usually generalize better, cf. [AML12]. What is meant by a "simple" model is described by the regularization function Ω. For instance, sparse models can be seen as simple models. To measure the sparsity of a model, one would count the nonzero entries of the parameter x, i.e., calculate the ℓ_0-norm $\|x\|_0$ (whereas $\|\cdot\|_0$ is of course not a real norm). Since the ℓ_0-norm is hard to handle during optimization, typically, approximations thereof are used, e.g., the ℓ_1-norm. For linear models, i.e., $m : \mathbb{R}^{n_b \times n_a} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_b}, m(X, a) = Xa$, it was proven that the ℓ_1-norm indeed ensures sparse solutions if the resulting system of equations is over-determined, i.e., $N = |D_{\text{tr}}| < n_a \cdot n_b$ [Don06; RZH04]. Moreover, the solution of the regularized problem depends on the choice of λ. If λ is chosen too small, overfitting is still likely. If, in contrast, λ is too large, the trained model may not be able to approximate the target t well enough since the model complexity is too low, i.e., the model cannot learn from the given data. The issue of choosing λ is also addressed in Chapter 5.

Models There is a variety of different model classes that are used to learn from data, and depending on the specific learning problem the performance may differ.

One of the simplest model classes is the family of linear models, which includes *support vector machines* (SVM) typically used for classification tasks [Bis06]. These are used to illustrate the results regarding the structure of the regularization path in Section 5.4.2. Another large and well-known model class is formed by (artificial) neural networks, which can be further divided into different architectures. These are mainly used in this thesis and are introduced in the next section. Moreover, there are other methods specifically tailored to build surrogate models for dynamical systems. One of which is eDMD, which aims to approximate the Koopman operator. This is presented in Section 2.2.3. Of course, there are many other model classes which, however, are not discussed here.

2.2.2 Neural Networks

Probably the best known class of models in machine learning, not least due to increased media attention, are so-called *artificial neural networks* or simply *neural networks* (NN) [Bis06; GBC16]. Inspired by the structure of human brain cells, these networks are composed of layers of artificial neurons where the neurons of one layer are connected with neurons of other layers. Neural networks are considered a key technology for recent successes in image and speech recognition, as well as artificial intelligence in general. For instance, NNs were at the core of AlphaGo, which was the first artificial intelligence to defeat a professional Go player [Sil+16; Sil+17]. Furthermore, from a theoretical (mathematical) perspective, NNs are an active research area. See, for instance, [Ber+21; E+20] for an overview of current challenges.

The most basic structure of an NN is a *perceptron* or *neuron*, which is given by

$$m : \mathcal{X} \times \mathcal{A} \to \mathcal{B}, ((w, w^0), a) \mapsto b = \sigma\left(w^\top a + w^0\right),$$

where $\mathcal{A} \subseteq \mathbb{R}^{n_I}$, $\mathcal{B} \subseteq \mathbb{R}$, $(w, w^0) \in \mathcal{X} = \mathbb{R}^{n_I+1}$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is a (nonlinear) continuous function, the so-called *activation function*. The trainable parameters are given by $x = (w, w^0)$, where w are called the *weights* and w^0 is called a *bias*. Here, we usually identify x with a vector in $\mathbb{R}^n$, where $n = n_I + 1$ is the number of (trainable) parameters in the perceptron.

Combining multiple perceptrons in parallel and serial leads to a more complex model class, a *neural network*, which can be described by a graph, cf. Figure 2.4. The NN shown in Figure 2.4b is a *feed-forward neural network* (FNN) as the input is propagated forward towards the output without feedback. More precisely, it is a *fully connected neural network*. The function describing this model class is given by $m : \mathcal{X} \times \mathcal{A} \to \mathcal{B}, m(x, a) = b$ with

$$\begin{aligned} h_0 &= a, \\ h_j &= \sigma_{h_j}\left(W_j^\top h_{j-1} + w_j^0\right) \quad \text{for } j \in \{1, \dots, K\}, && (2.7) \\ b &= \sigma_o(\underbrace{W_{K+1}^\top h_K + w_{K+1}^0}_{=h_{K+1}}), && (2.8) \end{aligned}$$

where (2.7) and (2.8) describe the equation of the *hidden* and the *output layer*, respectively. Thus, K denotes the number of hidden layers. The parameters that

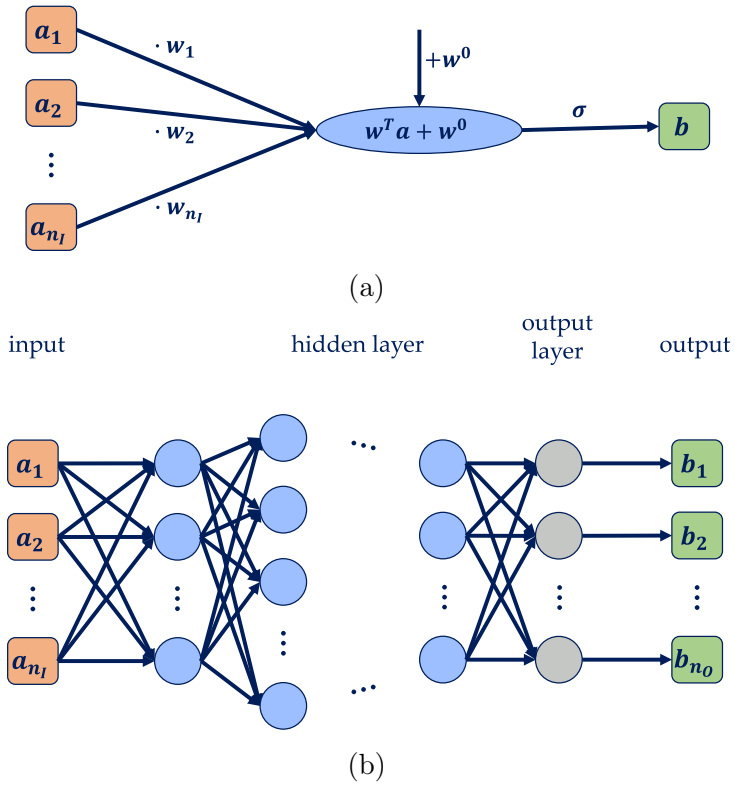


Figure 2.4: Architecture of a perceptron (a) and a fully-connected FNN (b).

can be optimized are the weights $W_j \in \mathbb{R}^{n_{j-1} \times n_j}$ and the biases $w_j^0 \in \mathbb{R}^{n_j}$, where n_j is the number of neurons of the j-th layer. Accordingly, n_0 and n_{K+1} are equal to the input and output dimension, respectively, i.e., $n_0 = n_I$ and $n_{K+1} = n_O$, where $\mathcal{A} \subseteq \mathbb{R}^{n_I}$ and $\mathcal{B} \subseteq \mathbb{R}^{n_O}$. Similar to the perceptron, we summarize the trainable parameters in a single vector $x = ((W_1, w_1^0), \ldots, (W_{K+1}, w_{K+1}^0)) \in \mathbb{R}^n$, where $n = \sum_{j=1}^{K+1} n_{j-1} \cdot n_j + n_j$ is the number of trainable parameters. Moreover, $\sigma_{h_j} : \mathbb{R}^{n_j} \to \mathbb{R}^{n_j}$, $j \in \{1, \ldots, K\}$, and $\sigma_o : \mathbb{R}^{n_{K+1}} \to \mathbb{R}^{n_{K+1}}$ are the activation functions for the hidden layers and the output layer, respectively. As an activation function, in principle, every continuous function can be chosen. For the hidden layers, typically, scalar functions are chosen that are applied elementwise to the input vector. Some common choices for activation functions are summarized in Table 2.1. If the NN consists of more than two hidden layers, the network is typically called a *deep neural network*.

Besides the practical success stories, there are also theoretical reasons to apply NNs and study them further. The most fundamental result is that FNNs can approximate every real-valued continuous function on a compact set $\mathcal{A}$ arbitrary well [HSW89]. Although this result is promising in the first place, it states nothing about the architecture and amount of data needed for a certain (probabilistic) error. Thus, there is still an active research area studying the approximation and generalization properties of NNs. The third aspect that has to be considered is the training of NNs, of which some aspects are also discussed in this thesis.

There are various architectures of NNs, usually specified for certain tasks. For

Table 2.1: Some examples for possible activation functions $\sigma : \mathbb{R}^m \to \mathbb{R}^k$.

	Mathematical Definition	Applied in ...
Tangent hyperbolicus	$\sigma(h) = \tanh(h)$ (applied elementwise)	hidden l.
Softplus	$\sigma(h) = \log(\exp(h) + 1)$ (applied elementwise)	hidden l.
Rectified linear unit	$\sigma(h) = \mathrm{ReLU}(h) = \max(h, 0)$ (applied elementwise)	hidden l.
Logistic function/ Sigmoid function	$\sigma(h) = \frac{1}{1+\exp(-h)}$ (applied elementwise)	hidden l.
Identity	$\sigma(h) = \mathrm{id}(h) = h$	linear output l.
Softmax (layer)	$\sigma(h)_i = \frac{\exp(h)_i}{\sum_{j=1}^{k}(\exp(h)_j)}, i \in \{1, \dots, m\}$	output l. (classification)

instance, *convolutional neural networks* (CNN) are used to classify images. Note that we did not allow feedback within the network so far. The information from the data is only propagated forward through the network, layer per layer. However, there are also NNs that allow for feedback, so-called *recurrent neural networks* (RNN). These are typically used to handle time series data and are addressed later.

Training of NNs

The training of an NN is basically solving an optimization problem where the objective is the (nonconvex) loss function L, which is assumed to be continuously differentiable, i.e., $L \in C^1$. This minimization problem is usually solved by the *steepest descent method* [NW06]. Typically, the gradient is not approximated in each optimization step, but calculated exactly (to machine accuracy) using *algorithmic differentiation* (AD) [GW08b]. In the neural network literature, the resulting algorithm is called *backpropagation*. To further increase the performance of the training, a *stochastic gradient descent* method is usually applied, meaning that that the loss and the gradient are not evaluated on the entire training set D_{tr} but only for single points or subsets.

Backpropagation The backpropagation algorithm is, in principle, the *reverse mode* of AD. The general idea of AD is to differentiate a function given by a concatenation of elementary functions (e.g., $+, -, \dots, \sin, \cos, \dots$), whose derivatives are known, utilizing the chain rule. Consider the concatenation $f = g_3 \circ g_2 \circ g_1$ of three differentiable functions $g_1 : \mathbb{R}^n \to \mathbb{R}^{m_1}$, $g_2 : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$ and $g_3 : \mathbb{R}^{m_2} \to \mathbb{R}$ and assume that their derivatives Dg_i are known. Then, the gradient of f in a point $x \in \mathbb{R}^n$ can be computed by

$$\nabla f(x) = Dg_1(x)^\top Dg_2(g_1(x))^\top \nabla g_3(g_2(g_1(x))).$$

The idea of the reverse mode is to compute in a first step the value of $f(x)$ and store the intermediate results $g_1(x)$ and $g_2(g_1(x))$ (as well as the performed op-

erations g_1,g_2 and g_3). In a second step, $\nabla f(x)$ can be computed by evaluating a row of matrix vector products, i.e., $z = Dg_2(g_1(x))^\top \nabla g_3(g_2(g_1(x))) \in \mathbb{R}^{m_1}$ and $Dg_1(x)^\top z = \nabla f(x) \in \mathbb{R}^n$. Applying this idea recursively to an NN yields a very efficient way to compute the gradient of the loss function with respect to the trainable parameters $x \in \mathcal{X}$. Powerful implementations of the backpropagation algorithm are provided, for instance, by TensorFlow [Mar+15] or PyTorch [Pas+19b]. For a more detailed explanation of backpropagation or AD in general, the reader is referred to [Bis06; GW08b].

Stochastic gradient descent method If the training data set is extremely large (which is often the case in practical applications), evaluating the loss function and its gradient can become very expensive. To overcome this issue, *stochastic gradient descent* (SGD) was established, see, for instance [GBC16]. The idea is that in each iteration, the loss function is evaluated only for a single, randomly chosen data point from the set of data points not used so far for the training. Once all data points have been used for one step in the gradient descent algorithm, the random data point can be drawn from the full data set again. One pass over all data points is called an *epoch*. A pseudo algorithm is given in Algorithm 1. Alternatively, it is also possible to consider more than one data point per iteration, i.e., to use a subset of data points, a so-called *(mini) batch*. This is usually referred to as *mini batch learning* in the literature. If the data set cannot be distributed equally into batches of the specified size, the remaining data points may either be left out for that epoch, cf. Algorithm 1, or constitute a smaller batch. From a theoretical perspective, the first question is

Algorithm 1 Pseudo-algorithm for mini batch learning, including the two special cases of SGD ($b = 1$) and the traditional steepest descent method ($b = |D_{\mathsf{tr}}| = N$).

Input: loss function $L : \mathbb{R}^n \times D \to [0, \infty) \in C^1$, training data D_{tr}, starting value $p^0 \in \mathbb{R}^n$, sequence of learning rates $(\varepsilon_k)_{k \in \mathbb{N}}$, batch size b
Output: critical point of L

Set $k = 0$.
while stopping criterion not met **do**
  Set $D_{\mathsf{cur}} = D_{\mathsf{tr}}$.
  while $D_{\mathsf{cur}} \geq b$ **do**
    Choose random set $B \subseteq D_{\mathsf{cur}}$ with $|B| = b$.
    Set $D_{\mathsf{cur}} = D_{\mathsf{cur}} \setminus B$.
    Calculate $\nabla L_B(p^k)$.
    Update: $p^{k+1} = p^k + \varepsilon_k \nabla L_B(p^k)$.
    Set $k = k + 1$.
  end while
end while

whether we can end up in a (local) minimum with the proposed algorithm, i.e., we are asking for a convergence proof. For the convex case, convergence can be proven [Bot99], based on the idea of stochastic approximation [RM51]. Concerning the convergence to global optima in the NN setting, there is a lot of current research, also for the traditional gradient descent algorithm. Recently, convergence to a global

optimum was shown for certain over-parameterized NNs [ALS19; Du+19].

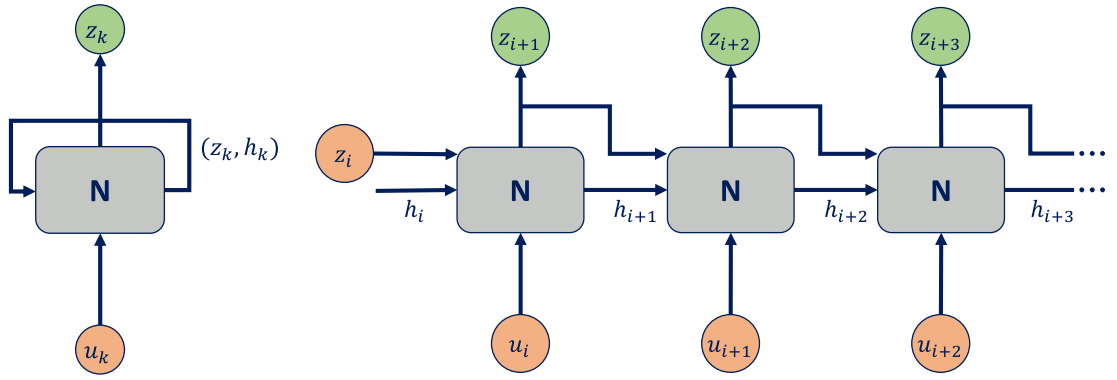
Besides the classical SGD, there are some more sophisticated stochastic gradient descent algorithms. One of the most famous algorithms to name is the *Adam* optimizer [KB14]. It is based on a so-called *momentum*. Essentially, this means that, in addition to the gradient, past iteration steps are taken into account, which usually results in a faster convergence [Pol64; Sut+13]. The Adam optimizer is a state-of-the-art optimizer for NN training. The convergence to critical points of the objective function was recently proven under certain assumptions [BB21; CC21]. (The convergence proof in the original paper [KB14] appeared to be wrong, see, for instance, [RKK18].)

Regularization As discussed in the previous section, regularization techniques can be used to regularize the loss function in order to avoid overfitting. Typical choices for the regularization term for NN training are the ℓ_2- or ℓ_1-norm [GBC16, Chapter 7]. Motivated by the case of linear models, cf. Section 2.2.1, the ℓ_1-norm is used to ensure sparsity for NNs as well. Although there is no proof for this (to the best of the author's knowledge), it is often used in practice and normally yields satisfying results. The regularization with the ℓ_1-norm is considered in the last part of this thesis, cf. Chapter 5.

Recurrent neural networks (RNN)

In contrast to the FNN considered so far, in *recurrent neural networks* (RNN), feedback of information within the network is allowed. Typically, RNNs are used to process sequential data, e.g., time series data of a (discrete) dynamical system. Due to the recurrent structure, RNNs are able to save information from previous time steps and incorporate the long time behavior of the time series data into the prediction for the next time steps. The basic structure of an RNN is shown in Figure 2.5a. To keep the notation a little bit simpler and give an intuitive interpretation, we assume that the aim is to approximate the time-T-map Φ of a dynamical system with control input and use the corresponding notation. Basically the RNN consists of an FNN, denoted by N that gets as an input the current (observed) state z_k and the current control input u_k. (In an uncontrolled setting, the input would be omitted.) The (observed) state for the next time step z_{k+1} is then approximated by N and afterwards fed back, i.e., the FNN is iteratively applied multiple times. In addition, there is usually a hidden state h_k that is also fed back to give the ability to save information on the dynamic behavior of the system in addition to the output state. Indeed, it is not necessary to feed back the output state z_k, but it is commonly done in time series prediction. The architecture of the described RNN may become clearer in the unfolded representation of the RNN, cf. Figure 2.5b. Every time step is predicted by the same FNN N but uses additional information from the previous time steps.

When the RNN is used as a model to predict future time steps, it gets only the current time step y_0 (or the observed state z_0) as an input and the predicted future states y_i, $i \geq 1$, are fed back after each time step. During the training, the time



(a) (b)

Figure 2.5: RNN for time series prediction in (a) and its unfolded version in (b). The grey box represents an FNN denoted by N. The green circles represent the output of the RNN, and the orange ones are the inputs.

series data includes the states of these future time steps. Hence, it is possible to just feed the hidden state back and use the true state at time t as input. This process is called *teacher forcing*.

Backpropagation through time Similar to FNNs, the training is done by employing gradient-based methods. The required gradient of the loss is usually computed via *backpropagation through time* (BPTT), which means that the backpropagation algorithm (or the reverse mode of AD) is just applied to the unfolded RNN.

Especially when the time series to be predicted are comparatively long, i.e., the RNN cell is run many times in a row, RNNs are known to suffer from the vanishing and exploding gradient phenomenon. This is due to the fact that the gradients become smaller or larger cell by cell (depending on whether they are smaller or larger than 1). To overcome this issue, specialized architectures were invented, which have proven to work well for time series prediction, for instance, *long short-term memory neural networks* and *echo state networks*.

Long short-term memory neural networks (LSTM) A special form of an RNN is a so-called *long short-term memory* (LSTM) neural network. These are specifically tailored to (long) sequential data, e.g., time series from dynamical systems, and are designed to avoid vanishing and exploding gradients [HS97].

The architecture of a single LSTM-cell is presented in Figure 2.6. A cell consists of three parts, namely, the *forget gate*, the *input gate* and the *output gate*. The equation of the forget gate is given by

$$f_k = \sigma\left(W_f \begin{pmatrix} z_k \\ u_k \end{pmatrix} + w_f^0\right),$$

where σ is the sigmoid function applied elementwise. The equation of the input gate

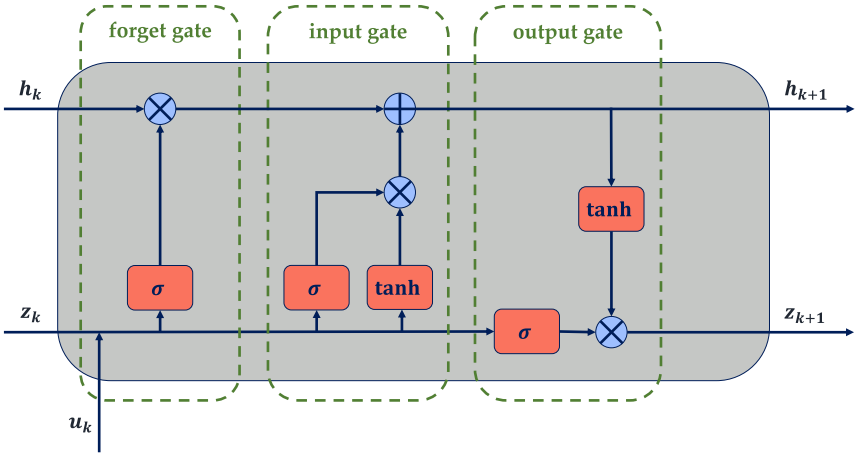


Figure 2.6: Architecture of a single LSTM-cell (for time series prediction).

is given by

$$i_k = \sigma\left(W_{i_1}\begin{pmatrix} z_k \\ u_k \end{pmatrix} + w_{i_1}^0\right) \odot \tanh\left(W_{i_2}\begin{pmatrix} z_k \\ u_k \end{pmatrix} + w_{i_2}^0\right),$$

where $\odot$ denotes the *Hadamard product* for matrices and vectors, i.e., the elementwise product. Combining these two parts and the internal hidden state h_k of the previous cell (referring to the previous time step for time series prediction), h_{k+1} is computed by

$$h_{k+1} = f_k \odot h_k + i_k.$$

Finally, the output of the LSTM cell is computed as

$$z_{k+1} = \sigma\left(W_o\begin{pmatrix} z_k \\ u_k \end{pmatrix} + w_o^0\right) \odot \tanh(h_{k+1}).$$

The matrices $W_f, W_{i_1}, W_{i_2}, W_o \in \mathbb{R}^{n_h \times n_z + n_u}$ and the vectors $w_f^0, w_{i_1}^0, w_{i_2}^0, w_o^0 \in \mathbb{R}^{n_h}$ are the parameters of the network and are optimized during the training, where n_h is the number of hidden neurons and n_u and n_z are the dimension of the control input and the system state, respectively. The initial values h_0 and z_0 are usually chosen to be zero in a standard LSTM. If a time series should be predicted, the initial state y_0 and its observed state z_0 are usually known and can also be used as the initial value in the LSTM.

In [HS97], a specialized form of BPTT was proposed for the training of LSTMs. The idea is to back-propagate the error not completely through the network, i.e., the gradients are only computed partially. However, nowadays, it is common to train LSTMs with the standard BPTT algorithm. Moreover, there are multiple adaptions to the original architecture, see, for instance, [Gre+17].

Concerning time series prediction for chaotic systems, there are various publications. For instance, the LSTM approach was successfully applied to forecast chaotic systems in a reduced order space in [Vla+18].

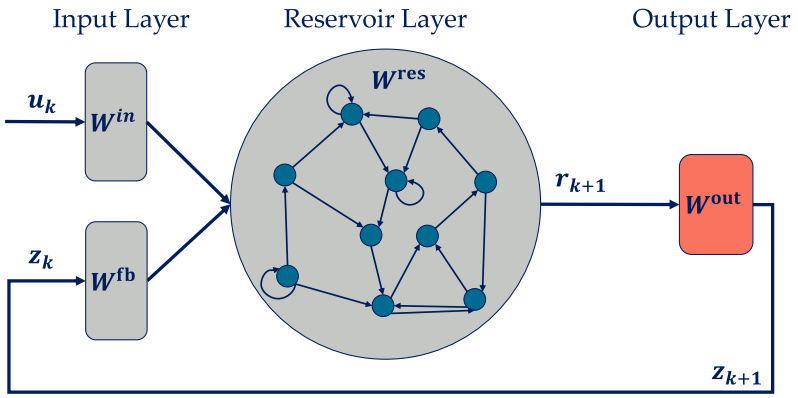


Figure 2.7: Scheme of an ESN used for time series prediction.

Echo state networks (ESN) *Echo state networks* (ESN) are another specialization of RNNs which are frequently used for time series prediction. They belong to the field of *reservoir computing*. The main idea for ESNs is to simplify and speed up the training process by limiting the training of the network to a few parameters. More precisely, only the parameters in the linear output layer can be trained, resulting in a system of linear equations that has to be solved. For an overview of ESNs and reservoir computing in general, we refer the reader to [LJ09] and for more details to [Jae02; Jae10; JH04].

Essentially, an ESN is an RNN with a linear output layer, as shown in Figure 2.7. The basic idea is to map the incoming, current state into a higher dimensional space via a sparsely connected, randomly initialized NN, called the *reservoir*. The parameters in the reservoir are not trained, i.e., they are fixed. The output of the reservoir, the so-called *reservoir state* r_k, can be seen as a hidden state that memorizes the long-term behavior of the data and is fed back into the reservoir in each iteration. Formally, the reservoir state r_{k+1} is given by

$$r_{k+1} = \sigma(W^{\mathsf{in}} u_k + W^{\mathsf{res}} r_k + W^{\mathsf{fb}} z_k),$$

where σ is a nonlinear continuous activation function, for instance, tanh. The matrices W^{in}, W^{res} and W^{fb} are randomly generated (sparse) matrices and are fixed, i.e., their entries cannot be optimized during the training. Depending on the application, W^{in} and W^{fb} may be omitted. For instance, if a dynamical system with control input is considered, the control is usually taken as additional time-dependent input. Otherwise, for time series prediction (without control input), the term W^{in} is omitted. Based on the current reservoir state, the output state, i.e., the (observed) state of the system in the next time step predicted by the ESN, is computed via

$$z_{k+1} = W^{\mathsf{out}} r_{k+1},$$

where W^{out} is the trainable output matrix. Note that the only trainable parameters are the ones of this linear output layer. As a surrogate model for time series prediction, the ESN approximates the observed state of the next time step z_{k+1}, which is usually provided to the reservoir via a feedback loop, in addition to the reservoir state r_{k+1}, to predict the next time step.

To train an ESN, a long time series of data $(z_0, \dots, z_N)$ (or $((z_0, u_0), \dots, (z_N, u_N))$ in the controlled case) is created which is mapped via the reservoir to a time series of reservoir states $(r_1, \dots, r_{N+1})$. Note that the (observed) state z_k is used to create the reservoir state for each time step, i.e., we utilize teacher forcing. For the training, only the system of linear equations $Z = W^{\text{out}} R$ with $Z = (z_1, \dots, z_N)$ and $R = (r_1, \dots, r_N)$ has to be solved, which is much faster than applying optimization algorithms such as gradient descent. Since the initial reservoir state highly influences the subsequent states, typically, the first time steps are left out for the training. Note that the described training of an ESN is different from the usual RNN training since the whole time series is used at once and not multiple short time series of fixed length are considered as training data. This way, the reservoir state evolves over the whole time series of training data. Thus, an ESN is able to capture the long time behavior of a system without explicitly using delay coordinates. Still, it does so indirectly through the feedback within the reservoir. Moreover, the prediction by the ESN usually starts directly with the last state of the training data such that the reservoir state already contains (implicitly) the dynamics of the previous time steps.

There are plenty of publications studying the ability of ESNs to predict chaotic systems, see, for instance, [CHS20; Jae10; LHO18; LHW12; MW17; Pat+17; Pat+18]. Furthermore, some works discuss the use of ESNs in an MPC context [Arm+19; Jor+18].

2.2.3 Data-Based Approximation of the Koopman Operator

The so-called *Koopman operator* is often considered in dynamical system theory and was first introduced in the work by Koopman [Koo31]. The basic idea is to study the evolution of *measurements* of the dynamical system state instead of the system state itself, leading to a linear operator whose properties are related to the dynamical system. Finite-dimensional, data-based approximations of the Koopman operator are of special interest since these lead to linear approximations of potentially nonlinear dynamics. For a comprehensive overview, including an introduction to the topic, practical applications, and current research questions, the reader is referred to [Bru+21].

Since the Koopman operator is defined for autonomous dynamical systems, we consider those first, while systems with control input are discussed later in this section. In accordance with Section 2.1, we consider the system given by the ODE $\dot{\mathbf{y}} = g(\mathbf{y}(t))$ and the corresponding time-T-map $\Phi : Y \to Y$ for the discrete time system. Now, we define the Koopman operator for Φ by

$$\mathcal{K} : L^2(Y) \to L^2(Y), \quad \mathcal{K}f = f \circ \Phi,$$

i.e., for a state $y_k \in Y$ and a measurement $f \in L^2(Y)$ it holds

$$\mathcal{K}f(y_k) = f(\Phi(y_k)) = f(y_{k+1}).$$

Here, $L^p(Y)$, $p \in [1, \infty)$, denotes the Banach space of all (equivalence classes of) measurable functions $f : Y \to \mathbb{R}$ with $\int_Y |f(y)|^p \, dy < \infty$. Note that $L^2(Y)$ is even

a Hilbert space. The Koopman operator represents the evaluation of the dynamical system in time in the measurement space. In the literature, the measurements f are also often referred to as *observables* in accordance with the observable h_{obs} usually given by the practical application, cf. Section 2.1.1, which also acts on the full state $y \in Y$. Nevertheless, we distinguish between those two types of functions since the observable h_{obs} is firmly specified by the application (and maps to $\mathbb{R}^{n_z}$, $n_z \in \mathbb{N}$), and the measurements f considered for the Koopman operator can be arbitrary functions in $L^2(Y)$ (mapping to $\mathbb{R}$). To compute a surrogate model for the dynamical system, the idea is to derive a finite-dimensional approximation of the Koopman operator, which is still linear and maps the current state lifted into the space of measurements to an approximation of the lifted state for the next time step.

Extended dynamic mode decomposition One common technique to build such an approximation is *extended dynamic mode decomposition* (eDMD) [KKS16; WKR15]. The basic idea of eDMD is to use a dictionary of functions that spans a subspace $\mathbb{V} \subseteq L^2(Y)$ of the space of measurements and to approximate the Koopman operator within this subspace. Therefore, data of the dynamical system is needed, i.e., pairs (y_i, y_{i+1}) with $y_{i+1} = \Phi(y_i)$ have to be collected. With the help of the basis functions, the data is mapped into the finite-dimensional subspace, in which a system of linear equations, whose solution is the approximation of the Koopman operator, can be solved. eDMD is an extension of the dynamic mode decomposition (DMD) [Sch10; Tu+14] (where only linear measurements are considered).

Formally, we assume that multiple data points are given and arrange these in two matrices,

$$D = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix} \quad \text{and} \quad \hat{D} = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_N \end{bmatrix},$$

where $\hat{y}_i = \Phi(y_i)$. The data can be collected as a consecutive time series, i.e., $\hat{y}_i = y_{i+1} = \Phi(y_i)$, or D can be randomly sampled in the state space Y.

As we want to consider the approximation in the measurement space, we need a basis that builds the subspace $\mathbb{V}$. To this end, we assume that we have a dictionary of linearly independent measurement functions $\{\psi_1, \ldots, \psi_M\}$ building the basis of $\mathbb{V}$. Now, we transfer the collected data into the (usually higher dimensional) measurement (sub)space and get

$$\begin{aligned} \Psi_D &= \begin{bmatrix} \Psi(y_1) & \Psi(y_2) & \cdots & \Psi(y_N) \end{bmatrix} \in \mathbb{R}^{M \times N} \quad \text{and} \\ \Psi_{\hat{D}} &= \begin{bmatrix} \Psi(\hat{y}_1) & \Psi(\hat{y}_2) & \cdots & \Psi(\hat{y}_N) \end{bmatrix} \in \mathbb{R}^{M \times N}, \end{aligned}$$

where $\Psi : \mathbb{R}^{n_y} \to \mathbb{R}^M$ collects all the measurements ψ_i, $i \in \{1, \ldots, M\}$, in one vector-valued function.

The idea is now to compute an approximate solution $K \in \mathbb{R}^{M \times M}$ of

$$\Psi_{\hat{D}} \approx K \Psi_D.$$

The least-squares solution of this system can be computed by

$$K^\top = \Psi_{\hat{D}} \Psi_D^+ = \left(\Psi_{\hat{D}} \Psi_D^\top\right)\left(\Psi_D \Psi_D^\top\right)^{-1},$$

where $^+$ denotes the Moore–Penrose inverse. Hence, the matrix K gives us now an approximation for the discrete time update of the lifted state $\Psi(y)$. By adding the identity to the basis functions, which is possible if the system states are restricted to a compact set Y, we thus get a linear approximation of the real system dynamics.

As discussed in Section 2.1.2, in practical applications, we can usually just observe the system state by an observable h_{obs}. In this case, the presented calculations can be applied to the observed state $z = h_{\mathsf{obs}}(y)$ instead of the full state y. Nevertheless, the observable h_{obs} obviously changes the state of measurements to $\{\psi_1 \circ h_{\mathsf{obs}}, \dots, \psi_M \circ h_{\mathsf{obs}}\}$. Depending on the observable h_{obs}, this might inhibit the ability to discover the whole dynamics. As already shortly discussed in Section 2.1.2, using delay coordinates as measurements might help to overcome this issue. In the context of the Koopman operator, this is also discussed in [Bru+17; KM18a].

In [KM18b], it was shown that the approximation of the Koopman operator computed by eDMD converges to the projection of the Koopman operator to the subspace $\mathbb{V}$ in the infinite data limit. The prediction error in the finite data case was considered in [LT20] and [Nüs+21] for DMD and eDMD, respectively.

Generator Indeed, the Koopman operator depends on the choice of the time step Δt for the time discretization of the time-T-map Φ. Defining the Koopman operator for arbitrary time steps Δt leads to a family of operators that build a C_0-semigroup. The generator of this semi-group is given by

$$\mathcal{L}f = \lim_{t \to 0} \frac{f \circ \Phi^t - f}{t}.$$

An approximation of the Koopman generator can also be derived from data and may be used to build a Koopman-based surrogate model, see, for instance, [Klu+20].

Koopman operator with control input There are various approaches to use surrogate models based on the Koopman operator within a control setting. In contrast to the NN architectures described before, the Koopman operator framework does not allow adding an additional input directly. The first intuitive idea to overcome this issue is to consider the dynamical system for the augmented state $\hat{y} = (y, u)$ instead (similar to Equation (2.2)). This way DMD and eDMD were extended to DMDc (for linear systems) [PBK16] and eDMDc [KM18a]. Since these methods derive a linear surrogate model, methods from linear MPC can be utilized, which is usually very efficient, cf. [KM18a]. The main question in this framework is how to choose the subspace of measurements ψ_i. One can choose functions acting solely on the state, solely on the control input, or on the product space, see [PBK18] for a discussion. Moreover, in [PBK18], an alternative variant of the Koopman operator was introduced that is called *Koopman with inputs and control* (KIC). It takes into account that there is (usually) no evolution in time of the control that has to (or can) be predicted by the Koopman operator and thus allows for differing input and output spaces.

In a nonlinear controlled system, the system state usually depends nonlinearly on the control input. Therefore, for the approximation of the Koopman operator, it

is necessary to include measurements that act on the system state and the control. (Although this is not proven, this seems reasonable, cf. [Bru+21]). This leads to a much larger number of measurement functions compared to the autonomous case (without control) and, thus, typically, to a larger amount of data that is required to cover the dynamics in the lifted space, cf. [Nüs+21]. To overcome these larger data requirements, another approach was followed by [Klu+20; Nüs+21; PK19; POR20]. There, the idea is to consider a finite subset $\{u^1, \dots, u^m\} \subseteq U$ of control values and define for each control an autonomous system for which the Koopman operator is defined. Via interpolation between the single autonomous systems, a control affine surrogate model is created. If the underlying system is control affine, as well, error bounds can be derived since the interpolation does not cause an additional error, cf. [Nüs+21; POR20]. A similar approach is followed in Chapter 4 for arbitrary surrogate models.

2.3 Multiobjective Optimization

In this thesis, we use multiobjective optimization to get a better insight into regularization problems, cf. Chapter 5. Hence, this section aims at giving a short introduction to this topic. For a more detailed insight into the area of multiobjective optimization, see, for instance, [Ehr05; Mie98].

In contrast to ordinary single-objective optimization, the idea in multiobjective optimization is to optimize a vector-valued function, i.e., we consider the *multiobjective optimization problem* (MOP)

$$\min_{x \in \mathbb{R}^n} f(x), \tag{MOP}$$

where $f = (f_1, \dots, f_k) : \mathbb{R}^n \to \mathbb{R}^k$ with $n, k \in \mathbb{N}$ and $k \geq 2$. We call f_i, for $i \in \{1, \dots, k\}$, the *objective functions* or *objectives*, $\mathbb{R}^n$ the *variable space* and $\mathbb{R}^k$ the *image space*. Here, we consider an unconstrained MOP. However, in principle, it is possible to add constraints and restrict the optimization problem to a feasible set. MOPs occur naturally in many practical applications in industry or economy, where different (usually competing) objectives have to be taken into account. For example, the design of electric control units (ECUs) in the field of autonomous, electric driving must take into account driving safety and comfort as well as the cost and electric range of the vehicle. Since the objectives are conflicting, the solution to such a task is an (optimal) compromise between the different requirements. Obviously, there is usually more than one possible compromise. Hence, the solution to an MOP is, in contrast to a single-objective optimization problem, typically a set and not a single point. The aim of multiobjective optimization is to compute this set, the so-called *Pareto set* named after Vilfredo Pareto [Par06], or at least parts or single points of it. For practical applications, it is usually necessary to choose a concrete solution that should be realized. This process is referred to as *decision making*, cf. [Mie98]. For instance, the decision can be made by an expert, the so-called *decision maker*, who is able to oversee the solutions in the Pareto set and decide which compromise fits best. Decision making can be seen as a whole area itself and is not covered in this thesis.

We start by introducing the mathematical concept of Pareto optimality and deriving necessary optimality conditions in Section 2.3.1. There, we consider also the case where f is nonsmooth, i.e., f is merely locally Lipschitz continuous. Finally, we give an overview of some solution methods in Section 2.3.2.

2.3.1 Pareto Optimality and Criticality

Since there exists no total order on $\mathbb{R}^k$ for $k \geq 2$, from a mathematical point of view, it is not directly clear what we mean by $\min f(x)$ for $f : \mathbb{R}^n \to \mathbb{R}^k$. Intuitively, from a practical perspective, optimal solutions of a problem with conflicting objectives are those where the value of one objective can only be improved when at least one other objective increases at the same time. To formalize this concept, we introduce the following (partial) order on $\mathbb{R}^k$.

Definition 2.3.1. *For $v, w \in \mathbb{R}^k$, we define*

$$v \leq w :\Leftrightarrow v_i \leq w_i \quad \forall i \in \{1, \dots, k\}$$

and analogously $v < w$.

It is easy to verify that the order defined in Definition 2.3.1 is a (strict) partial order. Based on this definition, we can now define what we mean by a solution of (MOP).

Definition 2.3.2. *Consider the minimization problem* (MOP).

(a) We say, a point $x \in \mathbb{R}^n$ dominates *a point $y \in \mathbb{R}^n$ if $f(x) \leq f(y)$ and $f(x) \neq f(y)$, i.e., if $f_i(x) \leq f_i(y) \quad \forall i \in \{1, \dots, k\}$ and $f_j(x) < f_j(y)$ for at least one $j \in \{1, \dots, k\}$.*

(b) A point $x^* \in \mathbb{R}^n$ is called Pareto optimal *if there is no $x \in \mathbb{R}^n$ dominating x^*. The set $\mathcal{P}$ containing all Pareto optimal points is called the* Pareto set *and its image $f(\mathcal{P})$ is referred to as the* Pareto front.

(c) A point $x^* \in \mathbb{R}^n$ is called locally Pareto optimal *if there exists an open set $U \subseteq \mathbb{R}^n$ with $x^* \in U$ such that there is no $x \in U$ dominating x^*.*

Remark 2.3.3. *In the literature Pareto optimal points are sometimes also called* Edgeworth-Pareto optimal *or* (Pareto) efficient *points. In accordance with Definition 2.3.2(a) another common terminology is* nondominated *points. See [Ehr05, Table 2.4] for a more detailed overview.*

Furthermore, it is possible to define a slightly weaker form of Pareto optimality which is also often used in the literature and helps us to formulate some of the upcoming results.

Definition 2.3.4. *Consider the minimization problem* (MOP). *A point $x^* \in \mathbb{R}^n$ is called* weakly Pareto optimal *if there is no $x \in \mathbb{R}^n$ with $f(x) < f(x^*)$, i.e., x^* is*

weakly Pareto optimal if there is no $x \in \mathbb{R}^n$ *with*

$$f_i(x) < f_i(x^*) \quad \forall i \in \{1, \dots, k\},$$

i.e., there is no point $x \in \mathbb{R}^n$ strictly dominating $x^* \in \mathbb{R}^n$. *Analogously, a point* $x^* \in \mathbb{R}^n$ *is called* locally weakly Pareto optimal *if there exists an open set* $U \subseteq \mathbb{R}^n$ *with* $x^* \in U$ *such that there is no* $x \in U$ *with* $f(x) < f(x^*)$.

The definitions are reasonable in the sense that (weakly) Pareto optimality implies locally (weakly) Pareto optimality and every (locally) Pareto optimal point is also (locally) weakly Pareto optimal, i.e.,

$$x \text{ is Pareto opt.} \Rightarrow \left\{ \begin{array}{l} x \text{ is locally Pareto opt.} \\ x \text{ is weakly Pareto opt.} \end{array} \right\} \Rightarrow x \text{ is locally weakly Pareto opt.}$$

Furthermore, concerning the optimal points of the single objectives f_i, $i \in \{1, \dots, k\}$, we can state the following remark.

Remark 2.3.5. *Consider the minimization problem* (MOP). *If* $x^* \in \mathbb{R}^n$ *is a (local) minimum of* f_j *for some* $j \in \{1, \dots, k\}$, *then* x^* *is (locally) weakly Pareto optimal for* (MOP). *Furthermore, if* x^* *is a strict (local) minimum, then it is even (locally) Pareto optimal for* (MOP).

To visualize the concept of Pareto optimality and the different introduced terms, we consider the following example.

Example 2.3.6. *We consider* (MOP) *with different objectives.*

(a) We start with a relatively simple example, where the objectives are given by two parabolas, i.e., $f : \mathbb{R} \to \mathbb{R}^2$ *is given by*

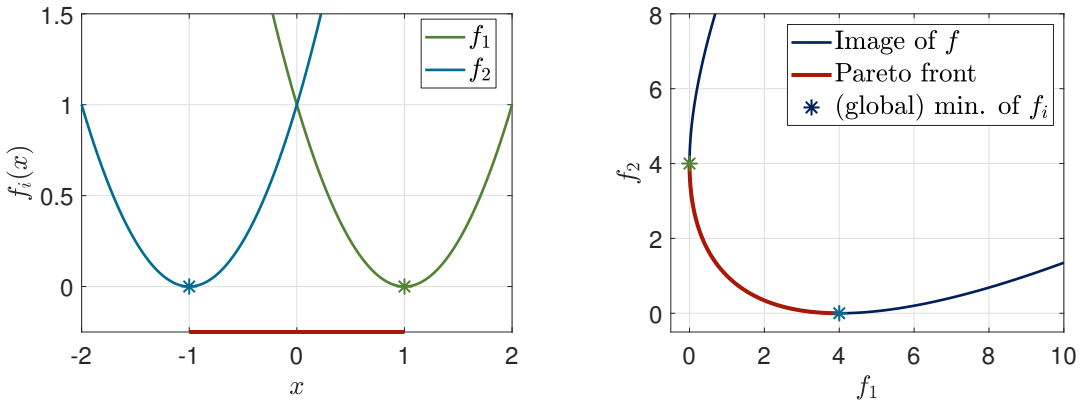
$$f(x) = \begin{pmatrix} (x-1)^2 \\ (x+1)^2 \end{pmatrix}. \tag{2.9}$$

Intuitively, the Pareto optimal solutions are the points on the line between the two (global) minima of the objectives as shown in Figure 2.8. Since the minima are strict, -1 *and* 1 *belong to the Pareto set as well, i.e., the Pareto set is given by* $[-1, 1]$.

(b) A very similar result is obtained when we consider $f : \mathbb{R}^2 \to \mathbb{R}^3$ *given by*

$$f(x) = \begin{pmatrix} (x_1+1)^2 + (x_2+1)^2 \\ x_1^2 + (x_2-1)^2 \\ (x_1-1)^2 + (x_2+1)^2 \end{pmatrix}. \tag{2.10}$$

The graphs of these functions are paraboloids and are shown in Figure 2.9, as well as the resulting Pareto set which is the triangle with the corners $(-1, -1)^\top$, $(0, 1)^\top$ *and* $(1, -1)^\top$.



(a) (b)

Figure 2.8: Results of (MOP) with f given by (2.9), i.e., Example 2.3.6(a). (a) Graphs of f_1 and f_2 and the Pareto set. (b) Image of f as well as the Pareto front. Furthermore, the global minima of the single objectives are marked by stars in both plots.

(c) *To visualize the different concepts of Pareto optimality, we consider the following more complex example. The objective* $f : \mathbb{R} \to \mathbb{R}^2$ *is given by*

$$f(x) = \begin{pmatrix} (x - \frac{3}{2})^2 \\ \frac{8}{9}x^8 + \frac{4}{3}x^6 - \frac{29}{6}x^4 + \frac{19}{9}x^2 + \frac{1}{2} \end{pmatrix}. \tag{2.11}$$

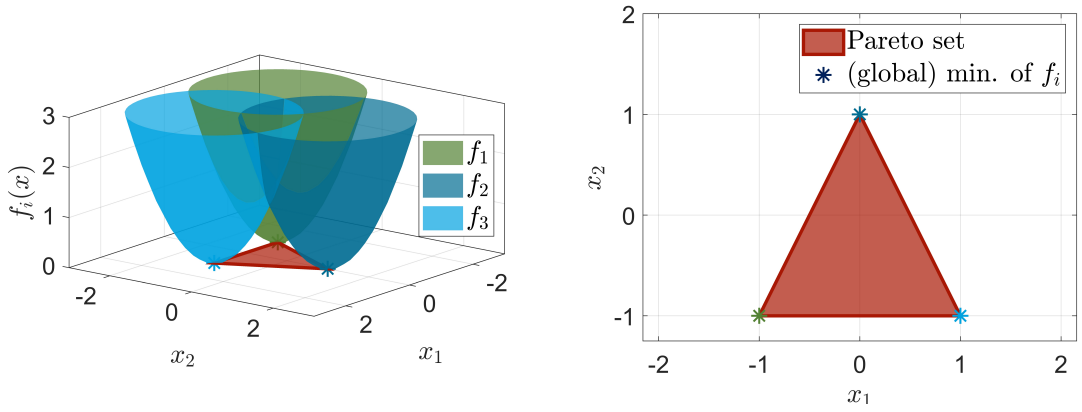
The results are shown in Figure 2.10. Due to the local maxima and the local minimum of f_2*, we observe that the intervals* $[-1, -0.5]$ *and* $[0, 0.5]$ *are locally Pareto optimal. Furthermore, the global minima of* f_2 *are not strict, and hence* -1 *is only weakly Pareto optimal, whereas* 1 *is Pareto optimal. This means, in particular, that* -1 *does not belong to the Pareto set, i.e., the Pareto set is given by the interval* $[1, 1.5]$.

If we assume that all objective functions are at least differentiable, we can derive a necessary condition for Pareto optimality similar to the single-objective case. For the single-objective case, the optimality condition was derived independently by Kuhn and Tucker [KT51], and Karush [Kar39] and are thus usually referred to as *Karush-Kuhn-Tucker* (KKT) *condition*. The condition for the multiobjective case was also published by Kuhn and Tucker in [KT51]. In accordance with the single-objective case, it is often referred to as the KKT condition as well.

Proposition 2.3.7. *Let* $f : \mathbb{R}^n \to \mathbb{R}^k$ *be differentiable and assume that* $x^* \in \mathbb{R}^n$ *is locally weakly Pareto optimal for* (MOP)*. Then, there exists* $\alpha^* \in [0, 1]^k$ *with*

$$\sum_{i=1}^{k} \alpha_i^* \nabla f_i(x^*) = 0 \text{ and } \sum_{i=1}^{k} \alpha_i^* = 1. \tag{KKT}$$

Since Proposition 2.3.7 holds for all locally weakly Pareto optimal points, it is, in particular, a necessary condition for Pareto optimality. If we set $k = 1$ in Proposition 2.3.7, we obtain $\nabla f_1(x^*) = \nabla f(x^*) = 0$ (with $\alpha_1^* = 1$) which is the well-known



(a) (b)

Figure 2.9: Results of Example 2.3.6(b). (a) Graphs of f_1, f_2 and f_3 and the Pareto set. (b) Pareto set in decision space. Furthermore, the global minima of the single objectives are marked by stars in both plots.

necessary condition for optimality in single-objective optimization. In analogy to the single-objective case, we make the following definition.

Definition 2.3.8. *A point $x \in \mathbb{R}^n$ is called Pareto critical if it satisfies (KKT). The corresponding $\alpha_i \in [0,1]$, $i \in \{1,\dots,k\}$ are called* KKT multipliers. *The set of all Pareto critical points is called the* Pareto critical set *and is denoted by $\mathcal{P}_c$.*

Furthermore, sufficient conditions can be derived if the objectives are twice differentiable. However, these conditions are not needed within this thesis, and we refer instead to [Mie98] for a discussion. Nevertheless, in the convex case, the KKT condition is also sufficient, summarized in the following proposition.

Proposition 2.3.9. *Let $f : \mathbb{R}^n \to \mathbb{R}^k$ be differentiable.*

(a) Assume f_i is convex for all $i \in \{1,\dots,k\}$. Then, $x^* \in \mathbb{R}^n$ is weakly Pareto optimal if and only if there exists $\alpha^* \in [0,1]^k$ such that (KKT) holds.

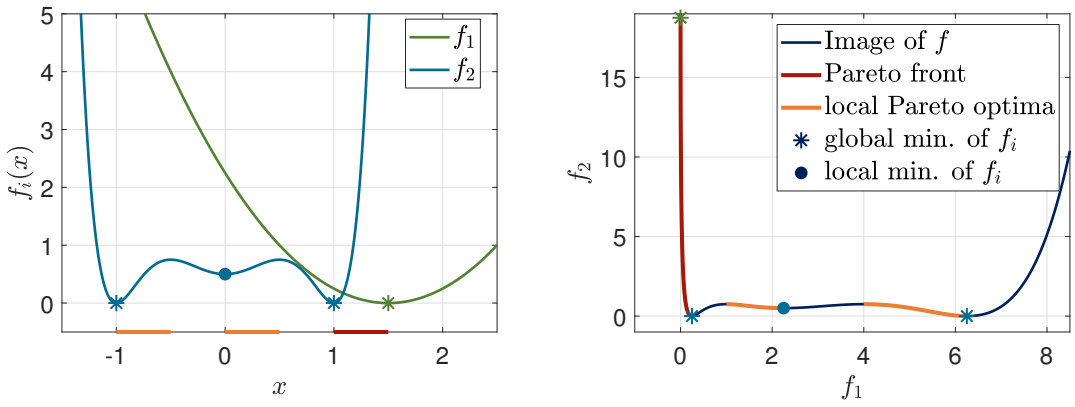
(b) Assume f_i is strictly convex for all $i \in \{1,\dots,k\}$. Then, $x^* \in \mathbb{R}^n$ is Pareto optimal if and only if there exists $\alpha^* \in [0,1]^k$ such that (KKT) holds.

In Chapter 5, we consider the ℓ_1-norm, i.e., a nonsmooth function, as an objective. Thus, we also give a brief overview of how to handle objectives that are merely *locally Lipschitz continuous* (and not differentiable) in the context of multiobjective optimization. For completeness, we give the following definition.

Definition 2.3.10. *Consider $f : \mathbb{R}^n \to \mathbb{R}^k$. We say that f is* Lipschitz continuous *if and only if there is some $L > 0$ such that for all $y_1, y_2 \in \mathbb{R}^n$, it holds*

$$|||f(y_1) - f(y_2)||| \leq L \, \|y_1 - y_2\| \,, \tag{2.12}$$

where $\|\cdot\|$ and $|||\cdot|||$ denote arbitrary norms of the $\mathbb{R}^n$ and $\mathbb{R}^k$, respectively. We say



(a) (b)

Figure 2.10: Results of (MOP) with f given by (2.11), i.e., Example 2.3.6(c). (a) Graphs of f_1 and f_2 and the (local) Pareto set. (b) Image of f as well as the (local) Pareto front. Furthermore, the global minima of the single objectives are marked by stars and the local minimum of f_2 by a dot in both plots.

that f is locally Lipschitz continuous *if and only if for all $x \in \mathbb{R}^n$ there is some $L > 0$ and $\varepsilon > 0$ such that (2.12) holds for all $y_1, y_2 \in B_\varepsilon(x) = \{y \in \mathbb{R}^n : \|x - y\| < \varepsilon\}$.*

We start by introducing a generalization of the concept of gradients commonly used in nonsmooth optimization, the so-called *subdifferential*. For a detailed introduction to nonsmooth functions and their optimization, we refer to [BKM14; Cla90]. Multiobjective optimization for nonsmooth functions is considered, for instance, in [MEK14; Mie98]. Note that if we consider a locally Lipschitz continuous function f, the set of points, where f is not differentiable, is a null set (Rademacher's Theorem, [EG15, Theorem 3.2]). This allows for the following definition of the subdifferential.

Definition 2.3.11. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz continuous, and let $\Omega \subseteq \mathbb{R}^n$ be the set of points where f is not differentiable. Then, the (*Clarke*)* subdifferential *of f in x is given by*

$$\partial f(x) := \mathbf{Conv}(\{g \in \mathbb{R}^n \mid \exists (x_j)_j \in \mathbb{R}^n \setminus \Omega \text{ with } x_j \xrightarrow{j\to\infty} x \text{ and } \nabla f(x_j) \xrightarrow{j\to\infty} g\}),$$

where **Conv** *denotes the convex hull. An element $g \in \partial f(x)$ is called a* subgradient.

Note that $\partial f(x)$ is nonempty, convex and compact [BKM14, Theorem 3.3]. If f is continuously differentiable, this definition coincides with the definition of the gradient in the way that the subdifferential is the set containing the gradient, i.e., if f is continuously differentiable in x, it holds $\partial f(x) = \{\nabla f(x)\}$. In the following introductory example, we calculate the subdifferential of the ℓ_1-norm, which is needed later in Chapter 5.

Example 2.3.12. *The ℓ_1-norm, given by*

$$f_{\ell_1}(x) = \|x\|_1 = \sum_{i=1}^{n} |x_i|, \quad \text{for } x \in \mathbb{R}^n,$$

is non-differentiable in all x with $x_j = 0$ for some $j \in \{1, \ldots, n\}$. It is easy to verify that the subdifferential is given by

$$\partial f_{\ell_1}(x) = \mathbf{Conv}\left(\{g \mid g_j = \operatorname{sgn}(x_j) \text{ if } x_j \neq 0, \quad \text{and} \quad g_j \in \{-1, 1\} \text{ if } x_j = 0\}\right),$$

i.e., the subgradients of f_{ℓ_1} are of the following form:

$$g = \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix} \in \partial f_{\ell_1}(x) \Leftrightarrow \begin{cases} g_j = \operatorname{sgn}(x_j) & \text{if } x_j \neq 0, \\ g_j \in [-1, 1] & \text{if } x_j = 0. \end{cases}$$

For instance, consider the point $\hat{x} = (2, 0, -3, 0)^\top \in \mathbb{R}^4$. Then, the subdifferential is spanned by four subgradients:

$$\partial f_{\ell_1}(\hat{x}) = \mathbf{Conv}\left(\left\{\begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}\right\}\right).$$

Analogous to the smooth case, a necessary optimality condition for MOPs containing nonsmooth objective functions can be obtained based on the subdifferentials of the single objectives, see, for instance, [Mie98], or [MEK14].

Proposition 2.3.13 (Nonsmooth KKT condition)**.** *Assume that $f : \mathbb{R}^n \to \mathbb{R}^k$ is locally Lipschitz continuous and x^* is locally weakly Pareto optimal. Then*

$$0 \in \mathbf{Conv}\left(\bigcup_{i=1}^{k} \partial f_i(x^*)\right) \subseteq \mathbb{R}^n, \tag{2.13}$$

i.e., it exists $\alpha^* \in [0, 1]^k$ with $\sum_{i=1}^{k} \alpha_i^* = 1$ and $g_i \in \partial f_i(x^*)$ for all $i \in \{1, \ldots, k\}$ such that

$$0 = \sum_{i=1}^{k} \alpha_i^* g_i. \tag{2.14}$$

Similar to the smooth case, we make the following definition.

Definition 2.3.14. *We call $x \in \mathbb{R}^n$* Pareto critical *if it satisfies the KKT condition (2.13) and we call the set of all those points the* Pareto critical set *$\mathcal{P}_c$.*

Note that the vector $\alpha \in [0, 1]^k$ now depends on the concrete choice of subgradients, i.e., the individual elements chosen from $\partial f_i(x)$. In accordance with the smooth case, the KKT condition is sufficient in the case of convex functions, cf. [MEK14, Theorem 14].

Proposition 2.3.15. *Let $f : \mathbb{R}^n \to \mathbb{R}^k$ be locally Lipschitz continuous.*

(a) Assume f_i is convex for all $i \in \{1, \dots, k\}$. Then, $x^* \in \mathbb{R}^n$ is weakly Pareto optimal if and only if Equation (2.13) holds.

(b) Assume f_i is strictly convex for all $i \in \{1, \dots, k\}$. Then, $x^* \in \mathbb{R}^n$ is Pareto optimal if and only if Equation (2.13) holds.

We conclude this section with an example in which the Pareto critical set for (MOP) with the ℓ_1-norm as one of two objectives is computed. Since the ℓ_1-norm is often used to ensure sparsity of the solutions, cf. Section 2.2.1, this means that we search for a compromise between an optimal solution of the first objective f_1 and sparse solutions. For more details, see also Chapter 5, where problems of this type are considered.

Example 2.3.16. *We consider Problem (MOP) where $f : \mathbb{R}^2 \to \mathbb{R}^2$ is given by*

$$f(x) = \begin{pmatrix} (x_1 - 1)^2 + (x_2 + 2)^2 \\ \|x\|_1 \end{pmatrix}. \tag{2.15}$$

According to Proposition 2.3.13 a point $x \in \mathbb{R}^2$ is Pareto critical if $\alpha \in [0, 1]$ exists with

$$0 \in \alpha \left\{ \begin{pmatrix} 2(x_1 - 1) \\ 2(x_2 + 2) \end{pmatrix} \right\} + (1 - \alpha)\partial f_2(x). \tag{2.16}$$

By inserting the concrete subdifferential of the ℓ_1-norm, cf. Example 2.3.12, it is easy to see that this condition is not fulfilled if $x_1 < 0$ or $x_2 > 0$ since

$$\begin{aligned} 2\alpha(x_1 - 1) - (1 - \alpha) < 0 \quad \forall x_1 < 0 \quad \text{and} \\ 2\alpha(x_2 + 2) + (1 - \alpha) > 0 \quad \forall x_2 > 0. \end{aligned}$$

Furthermore, if we consider the case where $x_1 > 0$ and $x_2 < 0$, the condition reduces to

$$\alpha \begin{pmatrix} 2(x_1 - 1) \\ 2(x_2 + 2) \end{pmatrix} + (1 - \alpha) \begin{pmatrix} 1 \\ -1 \end{pmatrix} = 0.$$

This is only true if $x_2 = -x_1 - 1$, $(0 <) \; x_1 \leq 1$ and $-2 \leq x_2 < -1$ (with $\alpha = \frac{1}{3-2x_1} = \frac{1}{2x_2+5} \in (\frac{1}{3}, 1]$). Hence, the line between the points $(0, -1)^\top$ and $(1, -2)^\top$ is Pareto critical (including $(1, -2)^\top$ and excluding $(0, -1)^\top$). Now, only the coordinate axes remain, i.e., $x_1 = 0$ or $x_2 = 0$. To be more precise, we have to analyze two cases: $x_2 = 0$ and $x_1 \geq 0$ and the case where $x_1 = 0$ and $x_2 < 0$. First assume that $x_2 = 0$, then (2.16) is only fulfilled if

$$4\alpha + (1 - \alpha)\xi = 0$$

with $\xi \in [-1, 1]$ holds. It is easy to see that this is only true if $\xi \in [-1, 0]$ with $\alpha \in [0, \frac{1}{5}]$. From $\alpha = 0$ it follows directly that $x_1 = 0$, i.e., the point $(0, 0)^\top$ is Pareto critical. (Since $(0, 0)^\top$ is the global minimum of the ℓ_1-norm this was already clear.)

For $x_1 < 0$, we already now that $\alpha = \frac{1}{3-2x_1}$ but this is equivalent to $x_1 = \frac{3\alpha-1}{2\alpha}$, i.e., for $\alpha \in (0, \frac{1}{5}]$ it follows $x_1 < 0$ which is a contradiction. Finally, we consider the case where $x_1 = 0$ and $x_2 < 0$. Here, it has to hold

$$\begin{aligned} -2\alpha + (1-\alpha)\xi = 0 \quad \text{and} \\ 4\alpha(x_2+2) + (1-\alpha) = 0 \end{aligned}$$

with $\xi \in [-1, 1]$. From the second equation it follows again that $\alpha = \frac{1}{2x_2+5}$ (and $x_2 \geq -2$ since $\alpha \in [0, 1]$). Inserting this into the first equation leads to $\xi = \frac{1}{x_2+2}$. As $\xi \in [-1, 1]$, it has to hold $x_2 \geq -1$.

To sum up, the Pareto critical set is given by the line between $(0, 0)^\top$ and $(0, -1)^\top$ and the line between $(0, -1)^\top$ and $(1, -2)^\top$, including the points $(0, 0)^\top$, $(0, -1)^\top$ and $(1, -2)^\top$, as shown in Figure 2.11. Since both objectives are strictly convex, the Pareto critical set coincides with the Pareto set, cf. Proposition 2.3.15.

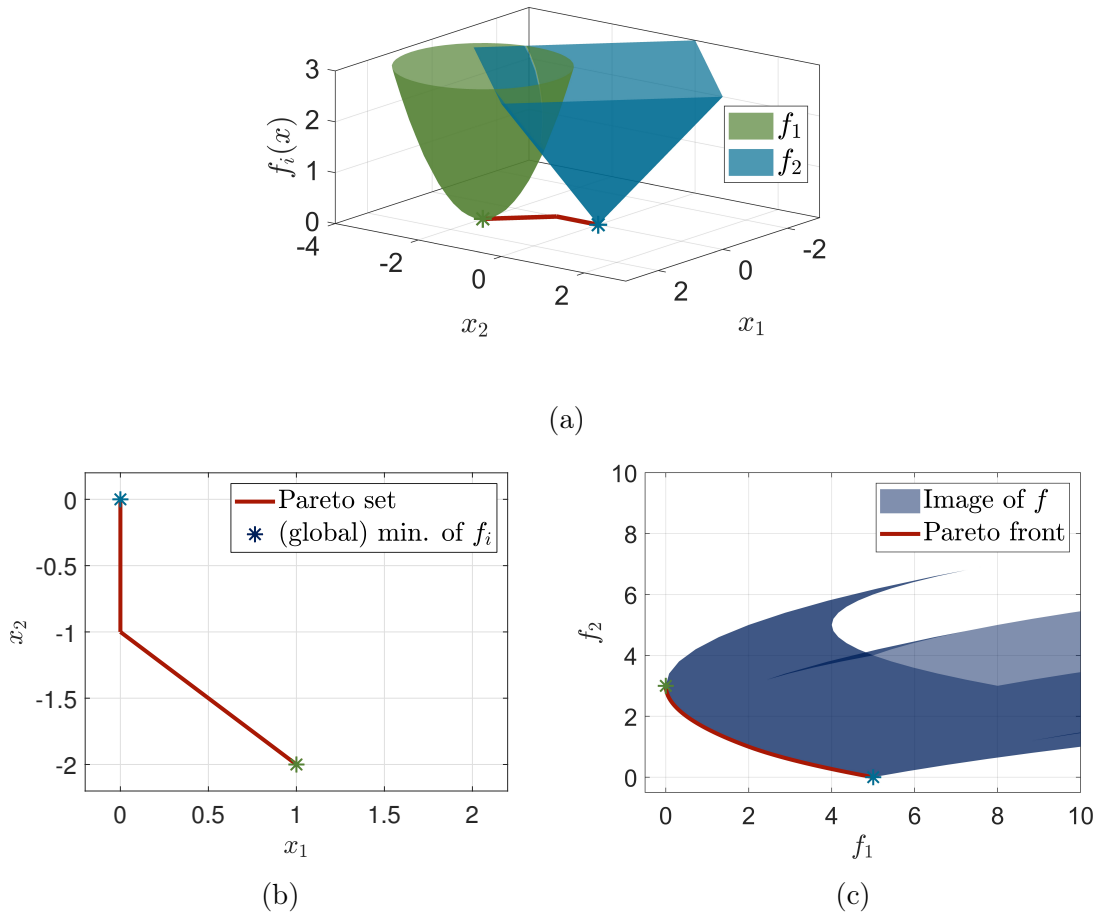


Figure 2.11: Results of (MOP) with f given by (2.15), i.e., Example 2.3.16. (a) Graphs of f_1 and f_2 and the Pareto (critical) set. (b) Pareto (critical) set of (MOP) in decision space. (c) Image of f as well as the Pareto front. Furthermore, the (global) minima of the single objectives are marked by stars in all plots.

Note that in the previous example, there is a kink in the Pareto set where it hits the x_1-axis, i.e., where it hits the nonsmoothness. This piecewise smooth structure usually occurs when considering biobjective MOPs where one objective is the ℓ_1-norm. This structure is further analyzed and exploited in Chapter 5.

2.3.2 Solution Methods

To solve (MOP), different solution methods exist. A very intuitive approach is to transform the MOP into a single-objective optimization problem such that traditional optimization methods can be applied. Two examples of such methods are the *weighted-sum method* and the *ε-constraint method* presented in the subsequent sections. Besides the transformation to a single-objective problem, there also exist methods that solve (MOP) directly, for instance, *multiobjective descent*. The methods named so far all have in common that they compute only one Pareto optimal point at a time. In contrast to that, *continuation methods* and *evolutionary algorithms* aim at computing the entire Pareto critical or Pareto optimal set, respectively. In addition to introducing the methods, we also discuss whether they are limited to smooth problems or can be applied to nonsmooth functions.

Weighted sum

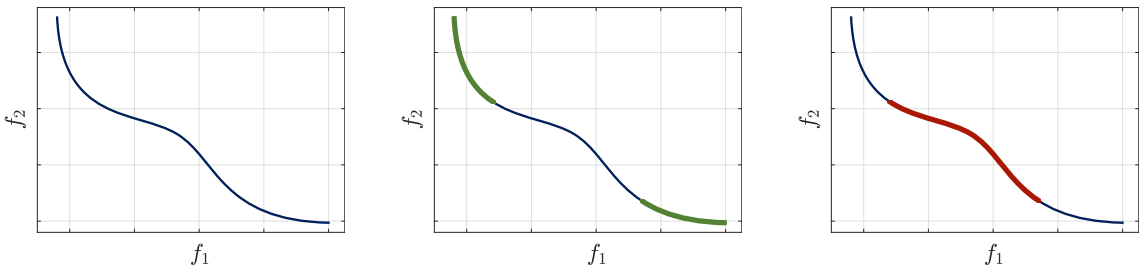
The weighted sum method is probably the simplest approach to transform an MOP into a single-objective optimization problem. As the name suggests, the idea is to weight the single objectives f_i and just sum them up, i.e., instead of (MOP), one solves

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{k} \alpha_i f_i(x) \tag{2.17}$$

with $\alpha \in [0,1]^k$ and $\sum_{i=1}^k \alpha_i = 1$. Problem (2.17) can now be solved with standard optimization methods for single-objective problems, e.g., with the *steepest descent* method or in case of non-differentiable objectives with *bundle methods*. By varying the parameter α different solutions can be obtained which can be proven to be at least weakly Pareto optimal for (MOP) (for all $\alpha \in [0,1]^k$ with $\sum_{i=1}^k \alpha_i = 1$). Unfortunately, not all Pareto optimal solutions can be obtained by the weighted sum method. This is only possible if all objectives f_i are convex, since otherwise, the Pareto front may be *nonconvex*. Formally, we say that the Pareto front is convex if the set

$$f(\mathcal{P}) + \{y \in \mathbb{R}^k : y \geq 0\}$$

is convex. Otherwise, we say that the Pareto front is nonconvex. (For a more formal discussion of the structure of the Pareto front, the reader is referred to [Ehr05].) As illustrated in Figure 2.12, only the convex parts of the Pareto front can be computed by the weighted sum method. Furthermore, note that if f_i is differentiable for all $i \in \{1, \ldots, k\}$ and x^* is a solution of (2.17) with weights $\alpha \in [0,1]^k$, then x^* is Pareto critical with KKT multipliers α.



(a) (b) (c)

Figure 2.12: Visualization of possible solutions of the weighted sum method in the objective space: (a) Pareto set (blue). (b) Solutions computable with the weighted sum method (green). (c) Optimal points not computable with the weighted sum method (red).

ε-Constraint

Another approach is followed by the *ε-constraint* method. Here, the idea is to minimize only one objective f_j, for a $j \in \{1, \dots, k\}$, and introduce constraints that ensure that the other objectives stay smaller than a chosen threshold. More formally, instead of (MOP), one solves

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f_j(x), \\ &\text{s.t. } f_i(x) \leq \varepsilon_i \quad \forall i \in \{1, \dots, k\} \setminus \{j\} \end{aligned} \tag{2.18}$$

with $\varepsilon_i \in \mathbb{R}$ for all $i \in \{1, \dots, k\} \setminus \{j\}$. Similar to the weighted sum method, solutions of (2.18) are at least weakly Pareto optimal and by varying the parameter ε different solutions can be obtained. Furthermore, $x \in \mathbb{R}^n$ is Pareto optimal if and only if x is a solution of (2.18) with $\varepsilon_i = f_i(x), i \in \{1, \dots, k\} \setminus \{j\}$ for all $j \in \{1, \dots, k\}$. Thus, to ensure Pareto optimality, (2.18) has to be solved k-times. In return, however, any Pareto optimal point can be computed by this approach, and convexity is not necessary, in contrast to the weighted sum method.

Multiobjective steepest descent

In contrast to scalarization methods where the MOP is transformed such that single-objective optimization techniques can be used, other approaches aim at adapting algorithms known from single-objective optimization to multiobjective optimization. A very prominent algorithm in this class is the *multiobjective steepest descent method* [FS00], which is briefly explained here. As the name suggests, it is a generalization of the steepest descent method for the single-objective case, and hence based on gradient information, i.e., we assume that f_i is at least continuously differentiable for all $i \in \{1, \dots, k\}$. The basic idea is to find a direction in the decision space such that a descent in all objectives can be realized, i.e., if $x^j \in \mathbb{R}^n$ is the current point, we want to compute a direction $v^j \in \mathbb{R}^n$ such that for $x^{j+1} = x^j + t_j v^j$ with a certain step size $t_j > 0$ it holds

$$f_i(x^{j+1}) < f_i(x^j) \quad \forall i \in \{1, \dots, k\}.$$

In the single-objective case this can be realized by taking the direction of the steepest descent, i.e., $v^j = -\nabla f(x^j)$. For the multiobjective case, in [FS00], the authors prove that a suitable descent direction is given by the optimal solution of

$$\begin{aligned} \min_{(v,\beta)\in\mathbb{R}^{n+1}} & \beta + \frac{1}{2}\|v\|_2^2, \\ \text{s.t.} & \nabla f_i(x^j)^\top v \leq \beta \quad \forall i \in \{1,\dots,k\}. \end{aligned} \tag{2.19}$$

Alternatively, one can derive the same descent direction v^j by solving the dual problem, cf. [FS00]. In this case, the direction is given by $v^j = -\sum_{i=1}^k \bar{\alpha}_i \nabla f_i(x^j)$ where $\bar{\alpha}$ is the optimal solution of

$$\begin{aligned} \min_{\alpha\in[0,1]^k} & \left\| \sum_{i=1}^k \alpha_i \nabla f_i(x^j) \right\|_2^2, \\ \text{s.t.} & \sum_{i=1}^k \alpha_i = 1. \end{aligned} \tag{2.20}$$

As in the single-objective case, the descent direction has to be scaled to ensure a descent (in all objectives), i.e., an appropriate step size t_j has to be calculated. This can be done by generalizing the well-known Armijo step size from the single-objective case [NW06], i.e., the step size is calculated by

$$t_j = \max\left(\left\{ t = \frac{1}{2^l} \;:\; l \in \mathbb{N}, f_i(x^j + tv^j) < f_i(x^j) + tc\nabla f_i(x^j)^\top v^j \;\forall i \in \{1,\dots,k\} \right\}\right) \tag{2.21}$$

with $c \in (0,1)$. If x^j is not critical, the set in (2.21) is not empty and hence the maximum is well defined, cf. [FS00]. In Algorithm 2, a pseudo code of the multiobjective steepest descent method is presented. It is possible to show that all accumulation points of the sequence $(x^j)_{j\in\mathbb{N}}$ generated by this algorithm are Pareto critical, cf. [FS00].

Algorithm 2 Pseudocode for the multiobjective steepest descent method [FS00].

1: Choose start value $x^0 \in \mathbb{R}^n$. Set $j := 0$.
2: **while** x^j is not (approximately) Pareto critical **do**
3: Compute direction v^j by solving (2.20) or (2.19).
4: Compute a step length t_j such that (2.21) holds.
5: Set $x^{j+1} = x^j + t_j v^j$.
6: Set $j = j + 1$.
7: **end while**

To improve the convergence rate, it is possible to include second-order information and derive a Newton direction [FDS09]. Besides gradient-based methods for smooth MOPs, there are also methods for nonsmooth MOPs usually approximating the subdifferentials of the objectives, e.g., the multiobjective proximal bundle method

[MKW15] or the descent method from [GP21] which is based on the approximation of the so-called *ε-subdifferential*. Furthermore, it is also possible to generalize the SGD method presented in Section 2.2.2 to the multiobjective setting, cf. [LV21; MPD18].

Continuation

The methods introduced so far have in common that they only compute a single point. In contrast, continuation methods aim at computing the entire Pareto critical set, or, to be more precise, entire connected components of it. Essentially, they are based on the fact that the Pareto critical set $\mathcal{P}_c$ in combination with the KKT multipliers is a manifold if $f \in C^2$, i.e., if f is twice continuously differentiable, and sufficiently regular. To formalize this, we define

$$H : \mathbb{R}^n \times [0,1]^k \to \mathbb{R}^{n+1}, H(x,\alpha) = \begin{pmatrix} \sum_{i=1}^k \alpha_i \nabla f_i(x) \\ 1 - \sum_{i=1}^k \alpha_i \end{pmatrix}, \tag{2.22}$$

i.e., it holds $x \in \mathcal{P}_c \Leftrightarrow \exists \alpha \in [0,1]^k : H(x,\alpha) = 0$ or in short $\mathcal{P}_c = \mathrm{pr}_x(H^{-1}(0))$, where $\mathrm{pr}_x : \mathbb{R}^{n+k} \to \mathbb{R}^n, \mathrm{pr}_x(x,\alpha) = x$ is the projection onto the decision space. This enables us to formulate the following theorem proven in [Hil01].

Theorem 2.3.17. *Let be $f \in C^2$, define the map H as in (2.22) and define the set $\mathcal{M} := (H_{|\mathbb{R}^n \times (0,1)^k})^{-1}(0)$. Furthermore, we denote the Jacobian of H by DH.*

(a) If $\mathrm{rk}(DH(x,\alpha)) = n+1$ for all $(x,\alpha) \in \mathcal{M}$, then $\mathcal{M}$ is a $(k-1)$-dimensional submanifold of $\mathbb{R}^{n+k}$ and the tangent space in a point $(x,\alpha) \in \mathcal{M}$ is given by $T_{(x,\alpha)}\mathcal{M} = \ker(DH(x,\alpha))$.

(b) If $(x,\alpha) \in \mathcal{M}$ with $\mathrm{rk}(DH(x,\alpha)) = n+1$ exists, then there is an open set $U \subseteq \mathbb{R}^{n+k}$ with $(x,\alpha) \in U$ such that $\mathcal{M} \cap U$ is a $(k-1)$-dimensional submanifold of $\mathbb{R}^{n+k}$ and the tangent space in (x,α) is given by $T_{(x,\alpha)}\mathcal{M} = \ker(DH(x,\alpha))$.

The concrete Jacobian of H in $(x,\alpha) \in \mathbb{R}^n \times [0,1]^k$ is given by

$$DH(x,\alpha) = \begin{pmatrix} \sum_{i=1}^k \alpha_i \nabla^2 f_i(x) & Df(x)^\top \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+k)} \tag{2.23}$$

This can be utilized to derive sufficient conditions which ensure that the requirements of Theorem 2.3.17 are fulfilled, summarized in the following lemma.

Lemma 2.3.18. *Let be $x \in \mathbb{R}^n$.*

(a) Let be $\alpha \in [0,1]^k$ with $\sum_{i=1}^k \alpha_i = 1$. If $\sum_{i=1}^k \alpha_i \nabla^2 f_i(x)$ is regular, then $\mathrm{rk}(DH(x,\alpha)) = n+1$.

(b) If $\nabla^2 f_i(x)$ is positive definite for all $i \in \{1,\dots,k\}$, then $\mathrm{rk}(DH(x,\alpha)) = n+1$ for all $\alpha \in [0,1]^k$ with $\sum_{i=1}^k \alpha_i = 1$.

In particular, Lemma 2.3.18(b) implies that the requirements of Theorem 2.3.17 are satisfied if all objectives are strongly convex, cf. [BV04].

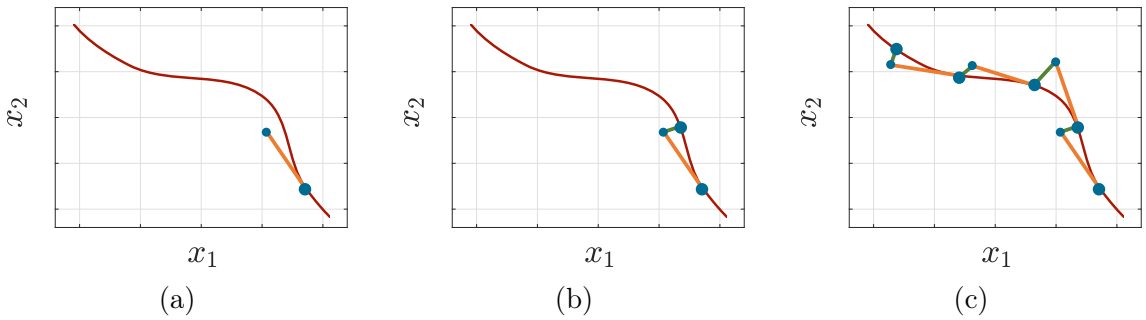


Figure 2.13: Illustration of the basic concepts of the continuation method (in the variable space $\mathbb{R}^n$): (a) Predictor step (orange). (b) Corrector step (green). (c) Multiple predictor and corrector steps resulting in an approximation of the Pareto critical set $\mathcal{P}_c$ (red).

The basic idea of continuation (or *homotopy*) methods is to start with an initial point $(x^j, \alpha^j) \in \mathcal{M}$ and find a point $(x^{j+1}, \alpha^{j+1}) \in \mathcal{M}$ near (x^j, α^j) by "walking along" the manifold. Therefore, two steps are usually followed, the so-called *predictor* and *corrector* step. Hence, these methods are also often referred to as *predictor-corrector* methods. They are not limited to the application of multiobjective optimization but are used to compute general parameterized manifolds. For a introduction to continuation methods in general, the reader is referred to [AG90]. In Figure 2.13, the basic concept of continuation methods is visualized for $n = k = 2$. Note that only the decision space $\mathbb{R}^n$ is plotted and not the whole space $\mathbb{R}^{n+k}$ where the continuation takes place.

In the predictor step, a movement along the tangent space of the manifold is done. Since the tangent space is given by $T_{(x^j,\alpha^j)}\mathcal{M} = \ker(DH(x^j, \alpha^j))$ according to Theorem 2.3.17, we are interested in an orthonormal basis $\{q_1, ..., q_{k-1}\}$ of the null-space of $DH(x^j, \alpha^j)$ since this allows us to derive evenly distributed directions in the case where the manifold is of higher dimension ($k > 2$). Such a basis can be obtained by utilizing a QR-factorization of $DH(x^j, \alpha^j)^\top$, cf. [Hil01].

After the predictor step, a corrector step is needed to get a Pareto critical solution. For instance, one can use the Gauss-Newton method to find a point (near the end point of the predictor step) that fulfills the KKT condition. Alternatively, it is possible to walk along the orthogonal space of the tangent space until the Pareto critical set is reached, cf. [Hil01]. The resulting root-finding problem can be solved, for instance, with Newton's method. To get the entire (current connected component of the) Pareto critical set, the predictor and the corrector step are repeated iteratively.

Besides various ways to realize the corrector step, there are different options to further adapt the continuation method, for instance:

- A crucial aspect is the step size of the predictor, where different strategies can be followed. One possible aim might be to derive an even distribution of the computed points in the decision or image space. In the first case, this can be done by normalizing the predictor. In the second case, based on the

orthonormal basis $\{q_1, \ldots, q_{k-1}\}$ derived in the predictor step, it is possible to combine and scale the directions for the predictor step such that an even distribution in the image space can be realized, cf. [Hil01].

- The endpoint of the predictor serves as the initial value to the corrector step. If many iterations are needed within the corrector step, this initial point might be far away from the Pareto critical set (which indicates a strong curvature). In this case, it is probably beneficial to reduce the step size of the predictor. If, in contrast, the corrector needs only a few iterations, it might be helpful to increase the step size [AG90].
- The predictor and corrector step become more expensive with an increasing dimension of the decision space $\mathbb{R}^n$, especially the computation of the Hessians $\nabla^2 f_i(x)$ becomes costly. One possibility to overcome this issue is to use update strategies such as BFGS, cf. [NW06, Algorithm 6.1], as proposed in [MS17].
- If, on the other hand, the number of objectives k increases, the main issue is that in the predictor step, many directions have to be followed. If $k > 2$, the Pareto critical set is (usually) no longer one-dimensional, and it is hard to handle the point cloud derived during the procedure. An alternative is to use methods that utilize coverings by sets, for instance, by boxes, as done in [SDD05].
- Interactive methods can help to compute just the relevant region of the Pareto critical set and thus reduce the computational costs [MS17; Sch+19].

The above considerations and results are clearly restricted to the case where f is smooth. In Chapter 5, a continuation method is derived for the nonsmooth case where one objective is the ℓ_1-norm.

Evolutionary algorithms

The last class of methods we present here are *evolutionary algorithms*. These are heuristic algorithms used for optimization and search in general, but there are also variants specifically tailored to multiobjective optimization. Basically, this type of algorithm is inspired by the evolution observable in nature. According to scientific theory, only the fittest (or the best adapted) individuals stay alive and are able to reproduce themselves and thus become the dominant species after a time [Dar59; Smi93].

Evolutionary algorithms work in a similar fashion. Starting with an initial *population*, i.e., a set of points in the variable space, the best-fitting points are filtered out based on a given fitness function. Then, these points are used to create new points by *recombination* and *mutation* to build the population of the new *generation*. Recombination means generating new points with the help of two (or more) individuals from the selected group of points with the help of a so-called *recombination operator*, for instance, by linear interpolation between two points. Mutation means that in each iteration, some random points are added to the population to ensure that the population is not steered too much into one direction and optimal solutions are missed. To this end, single individuals are usually adapted randomly. For instance,

new points are sampled randomly in a region around the selected points. In the next iteration, this newly generated population is again evaluated by the fitness function and so on. A pseudocode for a basic evolutionary algorithm is presented in Algorithm 3. For a more detailed introduction to evolutionary algorithms in general, see, for instance, [ES15; Nis97].

Algorithm 3 Pseudocode for a basic evolutionary algorithm.

1: Create randomly an initial *population* $X_0 \subseteq \mathbb{R}^n$. Set $j := 0$.
2: **while** termination criterion is not fulfilled **do**
3: Compute fitness function value of all individuals x in current population X_j.
4: Choose the *fittest* individuals (with lowest fitness function value) from X_j.
5: Create new population $X_{j+1} \subseteq \mathbb{R}^n$ through *recombination* and *mutation*.
6: Set $j = j + 1$.
7: **end while**

In multiobjective optimization, the fitness function is based on the concept of Pareto optimality, i.e., usually, in the selection step, nondominated points are sorted out to build the new population (via recombination and mutation). For the concrete realization of multiobjective evolutionary algorithms, there are many options resulting in various implemented and used algorithms. For an overview, the reader is referred to [Coe06; Zho+11]. One of the most popular multiobjective evolutionary algorithms is *NSGA-II* (nondominated sorting genetic algorithm II) [Deb+02]. Since evolutionary algorithms do not make use of gradient information, they are also applicable if there are objectives that are not differentiable. Furthermore, they are able to handle nonconvex objectives and are easy to implement. In addition, the Pareto optimal set can be computed at once in a single run of the algorithm. Nevertheless, they usually need a lot of function evaluations to derive appropriate approximations of the Pareto set, and there are usually no convergence guarantees.

3 | DeepMPC for Flow Control A Motivating Example

In this chapter, an example that reflects the promising possibilities of data-based methods, or more precisely, neural networks, in the field of complex system control is presented. We use an RNN architecture that is able to learn the time-dependent dynamics of a system with control input and can easily be incorporated into the MPC framework. Due to the combination of a deep neural network and MPC, the approach is referred to as *DeepMPC*. The performance is then evaluated using an example of flow control problems, the so-called *fluidic pinball*.

Flow control problems appear in many different engineering problems with different objectives, for example, to achieve more efficient combustion in cars or power plants, to increase high lift for airplanes, or to avoid wind noise for driving vehicles but also at architectural buildings [BW20]. These problems are described by nonlinear partial differential equations and are often expensive and time-consuming to solve numerically. In addition, in practical applications, there are usually only a few places in the system where sensors can be placed, and measurement data can be collected. Moreover, these sensors are often quite costly. Hence, real-time control based on the full state simulation is usually infeasible. Fortunately, the essential dynamics of these systems are often of low dimension and may be captured by measurements of only a few sensors [Den+19; Man+18]. This enables and motivates the use of surrogate models based on sensor data, learning the relevant parts of the dynamics.

Here, as a concrete example, the fluidic pinball is considered since it was specifically designed to test new (machine learning) control algorithms in the area of active flow control [Den+18; Noa+16]. It consists of a fluid that flows around three cylinders arranged in a triangle. The cylinders can be rotated to control the flow. On the one hand, this problem is relatively fast to solve and implement due to its simple architecture. On the other hand, the control task is challenging since the uncontrolled system possesses complex dynamical behavior, i.e., the system behaves chaotically in certain regimes [Den+18; Den+19]. Thus, although the fluidic pinball is an academic example, it is still a good reference for evaluating approaches to flow control. Hence, various data-based methods were used to study and solve different control tasks in the fluidic pinball setting. For instance, POD was used to approximate the system in [Pas+19a]. In [PK19; POR20], approximations of the Koopman operator and generator for fixed control inputs were computed and embedded into an MPC framework. Furthermore, approaches falling into the category of machine learn-

ing control (MLC) were used, cf. Section 2.1.3, where the idea is to directly learn a control law by utilizing genetic algorithms in a trial-and-error manner [Cor+21; Rai+20; RM21].

The work presented here extends the previously mentioned literature and considers an RNN architecture specifically tailored to time series prediction and can easily be integrated into the optimization step in the MPC framework. DeepMPC with similar RNN architectures was already used successfully in [LKS15] and in [BBK18] for controlling robot motion and laser bifurcation, respectively. The method is based on sensor data and allows for incorporating online learning based on data collected during the control phase. In [Mor+18], a similar approach, also based on NNs, is presented. There, the idea is to use an NN architecture consisting of an encoder and decoder motivated by Koopman operator theory. In contrast to the approach presented here, the surrogate model is based on the full state. Furthermore, the flow around a single cylinder at a considerably smaller Reynolds number ($Re = 50$) is considered as a test case, leading to significantly less complex dynamics.

After the RNN architecture and some training details are presented in Section 3.1, in Section 3.2, the fluidic pinball setting is introduced in more detail, and the results of the application of the DeepMPC approach are presented. In Section 3.3, the approach is discussed further based on the presented results and related to the research questions addressed in this thesis.

This chapter is based on [Bie+20], to which the author was the main contributor.

3.1 Design of the RNN

In the following, we present and discuss an MPC approach that uses an NN as a model. To be more precise, an RNN architecture is used, which is specifically tailored to the forecast of a time series based on past data similar to those presented in [BBK18; LKS15].

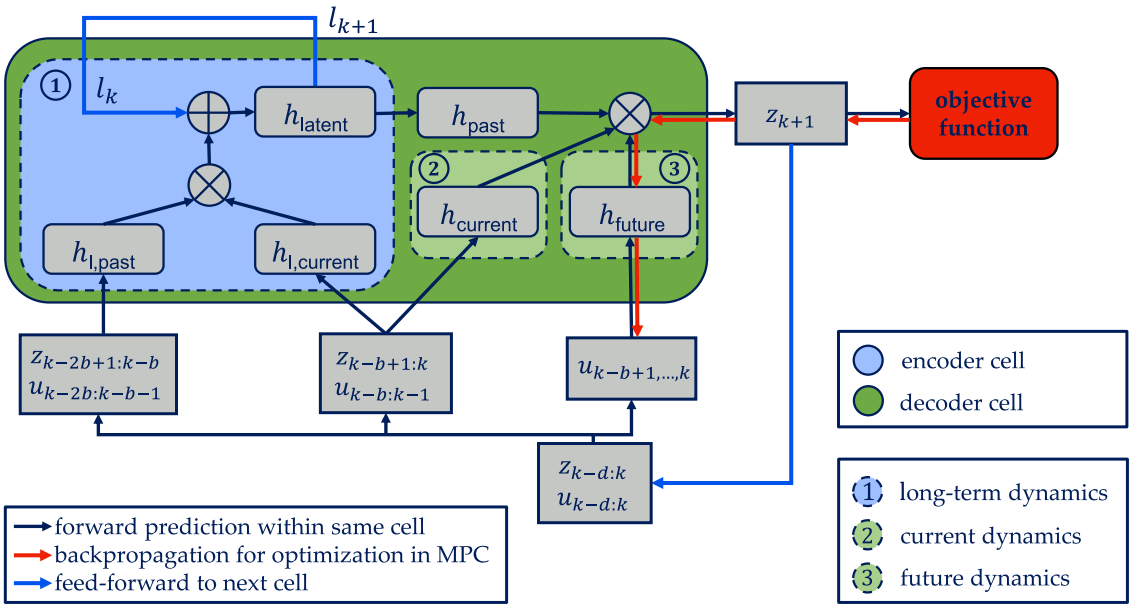
The aim of the RNN is to model the behavior of a dynamical system. More specifically, it is supposed to predict the observed states for the next time step z_{k+1} based on the current observed state $z_k = h_{\mathsf{obs}}(y_k)$ and the control input u_k which is applied to the system. In the case of the fluidic pinball, the observables may be the forces acting on the cylinders, i.e., the lift and the drag, as these can be measured easily in practice. Motivated by the Takens' result, cf. Section 2.1.2, the future state is predicted by using a time series of past observed states and the corresponding control inputs, i.e., the time series

$$(z_{k-d:k}, u_{k-d:k}) = (z_{k-d}, \dots, z_k, u_{k-d}, \dots, u_k)$$

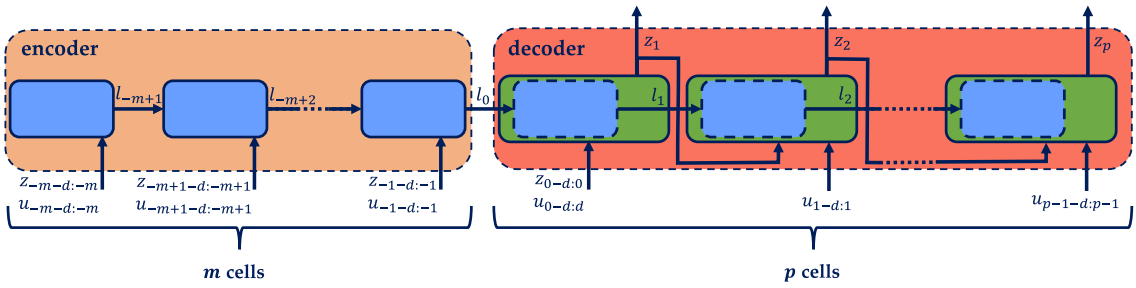
serves as input for the RNN cell that shall predict z_{k+1}:

$$z_{k+1} \approx \tilde{z}_{k+1} = \Phi^r_{\mathsf{RNN}}(z_{k-d:k}, u_{k-d:k}).$$

The delay d is chosen to be an even number, i.e., $d = 2b$ with $b \in \mathbb{N}$. This is due to the RNN architecture, for which the time series is divided into two parts, cf. Figure 3.1a.



(a) RNN cell with details



(b) Unrolled RNN

Figure 3.1: RNN architecture for the DeepMPC framework. In (a), a single RNN cell is shown, and in (b), the unrolled RNN is presented with encoder and decoder.

Since the aim is to predict the observed state not only one but p time steps into the future, an RNN with a feedback loop is used. In order to incorporate the long-term behavior, an artificial hidden state l_k is introduced and fed forward from one cell to another along with the predicted observed state. Therefore, the RNN cell is divided into two parts. The first part predicts the latent, hidden state l_k, whereas the second part predicts the observed state based on the latent state. As this hidden state has to be initialized properly before the first time step is predicted, the RNN is divided into an *encoder* and a *decoder* where an encoder cell predicts the hidden, latent state and the decoder cell consists of the encoder cell and the additional part predicting the observed state. The resulting unfolded RNN is shown in Figure 3.1b. Obviously, the number of decoder cells is given by the number of time steps in the prediction horizon p. The number of encoder cells, denoted by $m \in \mathbb{N}$, has to be chosen experimentally and depends on the underlying dynamics of the system.

Now, we want to detail the structure of a single decoder cell (including an encoder cell), which predicts the future observed state $z_{k+1} = h_{\text{obs}}(y_{k+1})$ based on the time-

delayed data $(z_{k-d:k}, u_{k-d:k})$. The decoder cell can be divided into three functional parts, cf. Figure 3.1a, which are composed of some FNNs represented by the smaller grey boxes in the diagram. The three parts are related to the long-term, the current, and the future dynamics. To this end, the input time series $(z_{k-d:k}, u_{k-d:k})$ is divided into

$$\begin{aligned} a_{\text{past}} &= (z_{k-2b+1:k-b}, u_{k-2b:k-b-1}) \in \mathbb{R}^{N_a}, \\ a_{\text{current}} &= (z_{k-b+1:k}, u_{k-b:k-1}) \in \mathbb{R}^{N_a}, \\ u_{\text{future}} &= (u_{k-b+1:k}) \in \mathbb{R}^{N_u}, \end{aligned}$$

where we identify a_{past}, a_{current} and u_{future} with the corresponding vectors of dimension $N_a = b \cdot (n_z + n_u)$ and $N_u = b \cdot n_u$, respectively. In accordance with Section 2.1, n_z denotes the dimension of the observed state z, and n_u is the dimension of the control input u. Besides the considered delay $d = 2b$, the number of encoder cells m, and the number of time steps in the prediction horizon p, the user has to determine the size of the NN by choosing a dimension of the latent state N_l and the number of parameters in the smaller FNNs (grey boxes). As these smaller networks consist only of one fully connected layer, it suffices to configure the output dimensions. Except for h_{latent}, where the output dimension is given by N_l, the dimension is chosen to be equal for all sub-networks and is denoted by N_h.

In the first part of the cell, we are interested in obtaining a latent state l_{k+1} to represent the long-term dynamics. The purpose of the encoder cell is exactly this, namely to encode the long-term dynamics of the system into a latent state. The concrete equations are given by

$$\begin{aligned} h_{\text{l,past}} &= \text{ReLU}(W_{\text{l,past}} \cdot a_{\text{past}} + w^0_{\text{l,past}}), \\ h_{\text{l,current}} &= \text{ReLU}(W_{\text{l,current}} \cdot a_{\text{current}} + w^0_{\text{l,current}}), \\ l_{k+1} = h_{\text{latent}} &= \text{ReLU}(W_{\text{latent,h}} \cdot (h_{\text{l,past}} \odot h_{\text{l,current}}) + W_{\text{latent,l}} \cdot l_k + w^0_{\text{latent}}), \end{aligned}$$

where $\odot$ denotes the Hadamard product, i.e., the elementwise multiplication, and $\cdot$ the normal matrix-vector product. As activation function, the rectified linear unit ReLU is chosen, cf. Table 2.1. The dimensions of the weight-matrices and bias-vectors are given by $W_{\text{l,past}}, W_{\text{l,current}} \in \mathbb{R}^{N_h \times N_a}$, $W_{\text{latent,h}} \in \mathbb{R}^{N_l \times N_h}$, $W_{\text{latent,l}} \in \mathbb{R}^{N_l \times N_l}$, $w^0_{\text{l,past}}, w^0_{\text{l,current}} \in \mathbb{R}^{N_h}$ and $w^0_{\text{latent}} \in \mathbb{R}^{N_l}$. The second part of the decoder cell incorporates the current dynamics and consists only of

$$h_{\text{current}} = \text{ReLU}(W_{\text{current}} \cdot a_{\text{current}} + w^0_{\text{current}}),$$

where $W_{\text{current}} \in \mathbb{R}^{N_h \times N_a}$ and $w^0_{\text{current}} \in \mathbb{R}^{N_h}$. Based on the time series of the control inputs u_{future}, including the current control u_k, the future dynamics are included in the third part given by

$$h_{\text{future}} = \text{ReLU}(W_{\text{future}} \cdot u_{\text{future}} + w^0_{\text{future}})$$

with $W_{\text{future}} \in \mathbb{R}^{N_h \times N_u}$ and $w^0_{\text{future}} \in \mathbb{R}^{N_h}$. Based on the three mentioned parts, together with

$$h_{\text{past}} = \text{ReLU}(W_{\text{past}} \cdot l_{k+1} + w^0_{\text{past}}),$$

the (approximated) future observed state is then obtained by computing

$$\tilde{z}_{k+1} = W_{\text{out}} \cdot (h_{\text{past}} \odot h_{\text{current}} \odot h_{\text{future}}) + w^0_{\text{out}},$$

where $W_{\text{past}} \in \mathbb{R}^{N_h \times N_l}$, $w^0_{\text{past}} \in \mathbb{R}^{N_h}$, $W_{\text{out}} \in \mathbb{R}^{N_h \times n_z}$ and $w^0_{\text{out}} \in \mathbb{R}^{n_z}$.

Training of the RNN In order to train the RNN, time series data of the real system with varying control inputs is created, i.e., a time series

$$((z_0, u_0), (z_1, u_1), \ldots, (z_N, u_N))$$

with $z_k = h_{\mathsf{obs}}(y_k)$ and $y_{k+1} = \Phi(y_k, u_k)$ is collected, where u_k is chosen uniformly at random in the control space $U \subseteq \mathbb{R}^{n_u}$. If the application requires a continuous control input, this has to be taken into account when creating the data, e.g., by choosing random control inputs for certain time steps and using interpolation techniques to compute continuous control inputs in between. As discussed in Section 2.2.2, RNNs suffer from exploding and vanishing gradients. To avoid these issues, a three-step approach is followed here, which was used in a similar way in [BBK18] and is based on the approach in [LKS15]:

1) Initialization of the parameters via a *conditional restricted Boltzmann machine* (RBM) and *Xavier initialization*
2) Training of the prediction of one time step, i.e., training of one decoder cell
3) Training of the whole RNN

In the first step, a part of the parameters is initialized via a modified conditional RBM which was originally proposed in [THR06]. RBMs can be used to learn the underlying probability distribution of a given data set. In the context of RNN training, they, therefore, can be helpful in finding good initial values for the network parameters [HS06]. Here, we use the RBM to initialize all parameters in layers that are directly connected to an input, like $h_{\text{l,past}}$ or $h_{\text{l,current}}$. Afterwards, the remaining weights are initialized using the normalized Xavier initialization [GB10]. After the initialization phase, we first train the prediction of a single time step, i.e., the network with all m encoder cells, but only one decoder cell is trained. This is done because numerical experiments suggested that the training can be stabilized in this way and suffers less from exploding and vanishing gradients. In the last step, the whole RNN is trained. In step 2) and 3) BPTT in combination with the stochastic gradient algorithm Adam, cf. Section 2.2.2, is used to train the NN.

MPC To fulfill the actual MPC task, besides the prediction of the future (observed) states, the control inputs $u_{k:k+p-1} = (u_i)_{i=k,\ldots,k+p-1}$ have to be optimized in the k-th time step. Similar to the training of the parameters of the NN, this is done via a gradient-based method where the gradient of the objective function with respect to the control inputs $u_{k:k+p-1}$ is computed via backpropagation, cf. Section 2.2.2, marked by the red arrows in Figure 3.1a. Here, we use a BFGS-method, cf. [NW06]. Since the latent state of the RNN is computed by m encoder cells, which need $d = 2b$ time steps as input, an initialization phase is required where $m+2b$ time

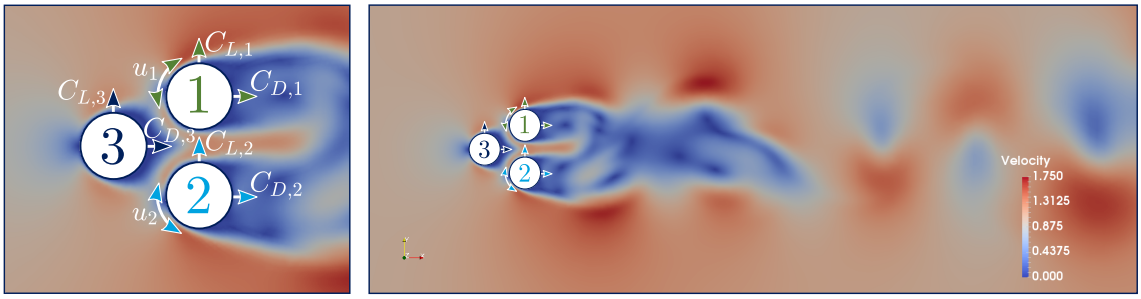


Figure 3.2: The setting of the fluidic pinball with a snapshot of the velocity of the uncontrolled system. On the right side, the whole domain with a colorbar for the norm of the velocity vector is presented. On the left side, a zoom with lift and drag coefficients and the control inputs can be seen.

steps are performed without active MPC control. In practical applications, it might be necessary to use other control strategies during this phase to keep the system in an admissible regime. Here, for simplicity, we keep the control input constant to 0 for the first $m + 2b$ time steps.

Online learning To further improve the model accuracy, additional sensor data can be collected during the operation phase and be used to perform online learning. This data probably represents the relevant region of the dynamical system states better than data collected with random control inputs. Furthermore, the data collection does not require additional time or effort since the data is needed for the MPC anyway. For the optimization, the same optimizer as in the offline training can be used. However, the choice of training parameters (hyper-parameters), e.g., the batch size and the step size of the optimizer (learning rate), are even more crucial as overfitting can occur easily due to the tailored data.

3.2 Application to a Fluid Flow Problem

To motivate the practical relevance of the presented RNN approach, we consider the so-called *fluidic pinball* as a flow control example that was designed to provide a control problem that behaves dynamically complicated, i.e., chaotic in certain regimes, and yet is still comparatively cheap and fast to compute. Thus, new methods to build surrogate models and control approaches can be tested easily, as done in [Den+19; Pas+19a; POR20].

In the setting of the fluidic pinball, a fluid flows around three cylinders arranged in a triangle in a plane as shown in Figure 3.2. All the cylinders have the same radius R, and a fluid enters the domain Ω from the left with constant velocity $y_\infty \in \mathbb{R}$. We denote the position by $x \in \mathbb{R}^2$ in Cartesian coordinates and the velocity of the fluid at time t in x in the horizontal and vertical direction with $\mathbf{y}(x,t) \in \mathbb{R}^2$. To analyze the performance of the presented approach, we study the system at different *Reynolds numbers* $Re = \frac{2R}{\nu} y_\infty$, where ν denotes the kinematic viscosity of the fluid. To be more precise, we consider $Re \in \{100, 140, 200\}$. If we additionally denote the

pressure with $\mathbf{p}$, the behavior of the system can be described by the incompressible 2D-Navier Stokes equations, i.e.,

$$\begin{aligned} \frac{\partial \mathbf{y}(x,t)}{\partial t} + \mathbf{y}(x,t) \cdot \nabla \mathbf{y}(x,t) &= -\nabla \mathbf{p}(x,t) + \frac{1}{Re} \Delta \mathbf{y}(x,t), \\ \nabla \cdot \mathbf{y}(x,t) &= 0, \\ (\mathbf{y}(x,0), \mathbf{p}(x,0)) &= (y_0(x), p_0(x)), \end{aligned}$$

where ∇ refers to the nabla operator, ∂ to the partial derivative and Δ to the Laplace operator. In this uncontrolled case, a no-slip condition (i.e., $\mathbf{y}(x,t) = 0$) has to hold at the boundary of the cylinders. By rotating the cylinders, the flow can be affected and controlled. In this example, we allow a rotation of the two rear cylinders (1 and 2). A time-dependent Dirichlet boundary condition on the surface of the cylinders introduces the control inputs u_j, $j \in \{1, 2\}$, which are the angular velocities of the rotation of the cylinders 1 and 2, respectively.

As the fluid flows around the cylinders, forces acting on the cylinders are induced. The force in the horizontal direction is referred to as the drag (F_D), and the force in the vertical direction is called the lift (F_L). These forces are strongly related to the fluid field and can be easily computed from the full system state or be measured when considering the real system. Therefore, as relevant system quantities that serve as input for the RNN, we observe the lift and drag coefficients given by

$$C_{L,j} = \frac{2F_{L,j}}{\rho y_\infty^2} \quad \text{and} \quad C_{D,j} = \frac{2F_{D,j}}{\rho y_\infty^2} \quad \text{for } j \in \{1, 2, 3\},$$

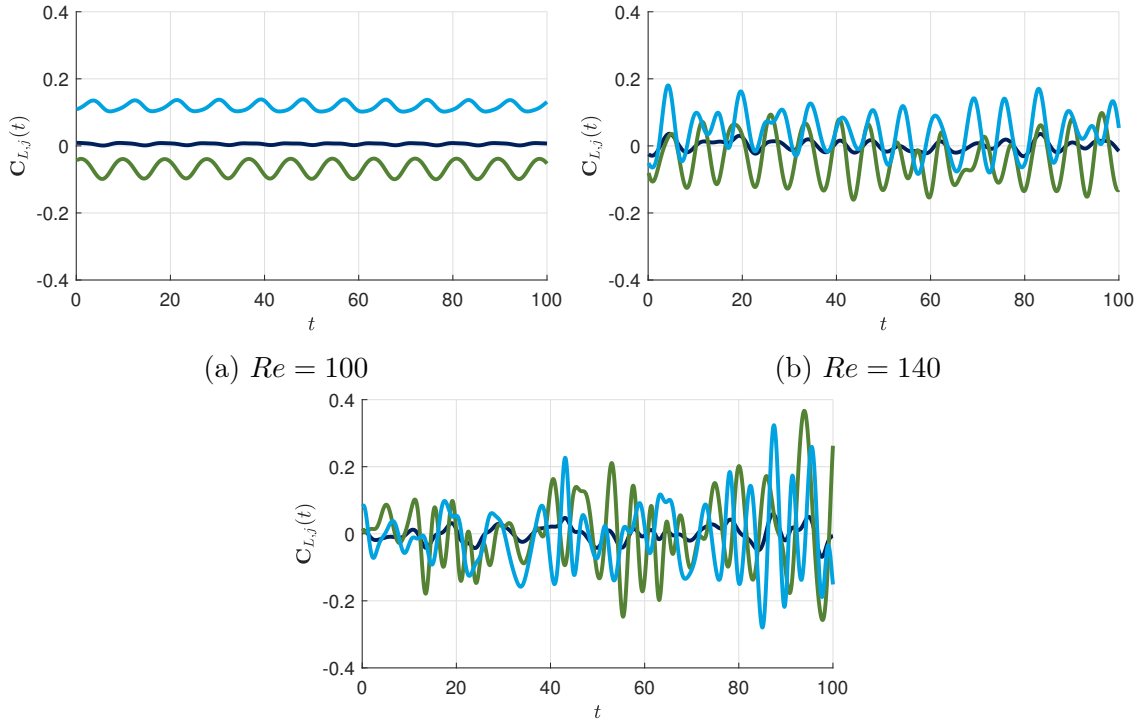
where ρ denotes the density of the fluid. Hence, the observed state is given by

$$z = (C_{L,1}, C_{L,2}, C_{L,3}, C_{D,1}, C_{D,2}, C_{D,3}).$$

To control the flow, we aim to steer the lift coefficients of the cylinders to predefined constant values, which then implicitly leads to a steady state of the flow.

The DeepMPC architecture, i.e., the RNN and the MPC loop, were implemented in Python using Tensorflow [Mar+15]. The simulation of the fluidic pinball was implemented by Sebastian Peitz using the open-source solver OpenFOAM [JJT07].

To observe the behavior for different complexity levels, we consider various Reynolds numbers, i.e., $Re \in \{100, 140, 200\}$. With increasing Reynolds number, the dynamics become more and more complicated, cf. Figure 3.3. At $Re = 100$, the system possesses a periodic solution. Interestingly, the lift coefficients do not evolve symmetrically with respect to the horizontal symmetry axis, and the mean of $|\mathbf{C}_{L,2}(t)|$ (over one period) is approximate twice the mean of $|\mathbf{C}_{L,1}(t)|$. According to [Den+19], the system has three steady solutions at $Re = 100$, where one is symmetric with respect to the horizontal axis and the other two break the symmetry. The symmetric solution is unstable, whereas the asymmetric solutions are stable and identical except for mirroring on the horizontal axis. At $Re = 140$ and $Re = 200$, the system behaves chaotically, and due to the more complex dynamics, it can be expected that the control task is much more complicated than for $Re = 100$. Note that, statistically, the symmetry of the flow is recovered for the higher Reynolds numbers since the



(a) $Re = 100$ (b) $Re = 140$

(c) $Re = 200$

Figure 3.3: Evolution of the lift coefficients at the three cylinders of the uncontrolled fluidic pinball at various Reynolds numbers. In accordance with Figure 3.2, $\mathbf{C}_{L,1}(t)$ is shown in green, $\mathbf{C}_{L,2}(t)$ in cyan and $\mathbf{C}_{L,3}(t)$ in dark blue.

system switches quickly between the oscillation around the first asymmetric steady solution and the other mirror-conjugated steady solution [Den+19].

The control task is to steer the first lift constantly to 1, the second to -1, and the third to 0, i.e., the reference trajectories for the lift coefficients are given by

$$(C_{L,1}^{\mathsf{ref}})_k = 1,\ (C_{L,2}^{\mathsf{ref}})_k = -1 \text{ and } (C_{L,3}^{\mathsf{ref}})_k = 0 \text{ for } k \in \{0, 1, \dots\}.$$

As only the two rear cylinders can be rotated, the influence of the control input on the lift coefficient $C_{L,3}$ of the front cylinder is almost negligible. Nevertheless, in the numerical experiments, forcing $C_{L,3}$ to zero yielded improved results. This way, it is probably avoided that the optimization drifts away in a regime where the model predicts (relatively) high absolute values of $C_{L,3}$. Since those values cannot occur in the real system, the model cannot represent the real behavior and is inaccurate when it predicts high absolute values of $C_{L,3}$. To achieve the control aim, an angular velocity of the cylinder rotation between -2 and 2 is allowed for both cylinders. In addition to the trajectory tracking error, a penalty term for too large deviations in the control inputs between consecutive time steps is introduced, leading to the

following MPC problem at time t_k (for $k \geq m + 2b$):

$$\min_{u_{k:k+p-1} \in ([-2,2]^2)^p} \sum_{i=k}^{k+p-1} \left(\sum_{j=1}^{3} \left| \tilde{z}_{j,i+1} - (C_{L,j}^{\text{ref}})_{i+1} \right|_2^2 + \beta \cdot \sum_{j=1}^{2} |u_{j,i} - u_{j,i-1}|^2 \right),$$
$$\text{s.t.} \quad \tilde{z}_{i+1} = \Phi_{\text{RNN}}^r(\tilde{z}_{i-d:i}, u_{i-d:i}) \quad \text{for } i \in \{k, \ldots, k+p-1\},$$

where the initial condition is given by the real system state, i.e., $\tilde{z}_{k-d:k} = z_{k-d:k}$. Note that $u_{1,i} = u_{2,i} = 0$ for $i < m + 2b$, i.e., during the initialization phase, cf. Section 3.1. In our study, we set $\beta = 0.1$ as preliminary experiments suggested that this is a reasonable value. The number of time steps in the prediction horizon is set to $p = 5$ for $Re = 100$, and for the more complicated tasks at $Re = 140$ and $Re = 200$, we found that $p = 10$ is a more suitable choice to achieve a robust control. The other chosen hyper-parameters are summarized in Table 3.1.

Table 3.1: Choice of RNN hyper-parameters.

	$Re = 100$	$Re = 140$	$Re = 200$
p	5	10	
m	5		
$d = 2b$	22		
N_l	400		
N_h	500		

To create training data, the fluidic pinball was simulated with random but smoothly varying control inputs, i.e., for both cylinders, a random rotation velocity between -2 and 2 was chosen independently every 0.5 second and interpolated by splines. The step size for the discretization of the control and hence the step size predicted by the resulting model is $\Delta t = 0.1$s. To get an accurate model of the fluidic pinball, the time discretization in the FEM simulation has to be chosen even smaller. Here, we choose $5 \cdot 10^{-3}$ seconds. To get a constant control input for the prediction step of $\Delta t = 0.1$s, the mean of the control over 20 time steps were taken. This way 150 000, 200 000 and 800 000 training data points for Reynolds number 100, 140 and 200 were computed, respectively. This large amount of training data was generated to ensure that all the dynamic properties of the system were represented in the data since there was no indication of how many data points were needed at the beginning of the experiments. However, the amount of data is certainly far too large and can be significantly reduced. This question is further addressed below.

The results of the experiments at $Re \in \{100, 140, 200\}$ over the interval $[t_0, t_f] = [0, 100]$ are shown in Figure 3.4 where the mean and the maximal error are computed by

$$e_{mean} = \frac{\Delta t}{t_f - 4} \sum_{k=\frac{4}{\Delta t}}^{\frac{t_f}{\Delta t}} \left(\frac{1}{3} \sum_{j=1}^{3} \left| (C_{L,j})_k - (C_{L,j}^{\text{ref}})_k \right|_2^2 \right),$$

and

$$e_{max} = \max_{\frac{4}{\Delta t} \leq k \leq \frac{t_f}{\Delta t}} \left(\frac{1}{3} \sum_{j=1}^{3} \left| (C_{L,j})_k - (C_{L,j}^{\mathsf{ref}})_k \right|_2^2 \right)$$

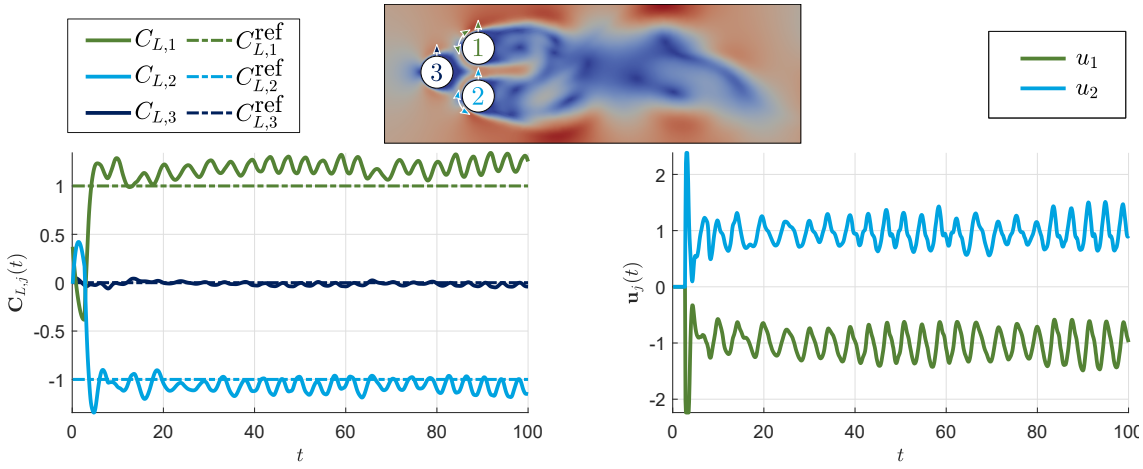
to measure the performance. Since the first steps belong to the initialization phase with $u_k = 0$, the error evaluation just starts after 4 seconds. It can be observed that the DeepMPC scheme leads to a good control performance for $Re = 100$. Nevertheless, $C_{L,1}$ can be controlled less effective than $C_{L,2}$. This is not surprising as the uncontrolled system possesses an asymmetric periodic solution, cf. Figure 3.3a. In comparison, the performance for $Re = 140$ and $Re = 200$ is considerably worse. As chaos can be observed in the uncontrolled systems, a prediction of the system state (by the RNN) is much more challenging and, at the same time, much harder to control. This is also observed in [POR20].

To study the performance of the DeepMPC scheme in more depth and to improve the control performance, different adaptions to the approach are made in the following.

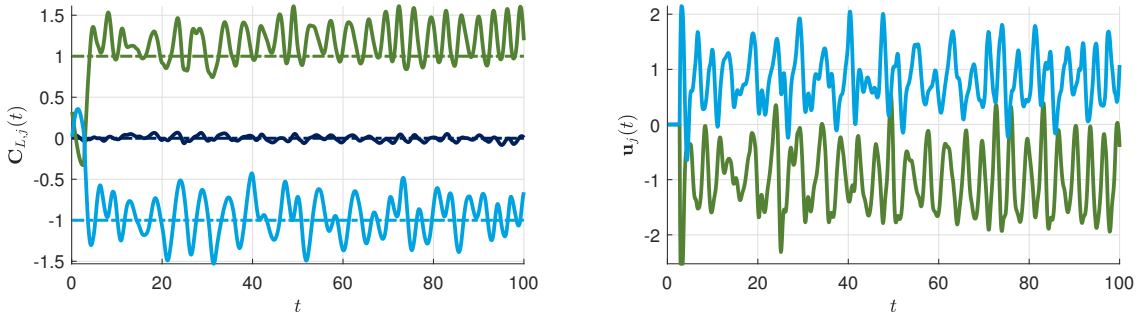
Exploiting the symmetry A reasonable idea to improve the prediction quality by the surrogate model, and hence the control performance is to incorporate knowledge about the system (behavior). In the case of the fluidic pinball, the domain is symmetric along the x_1-axis (the vertical coordinate axis crossing the midpoint of the front cylinder). Hence, the behavior of lift and drag is correspondingly contrary, i.e., we can double the training data by exploiting the symmetry as

$$\begin{aligned} \widehat{u} &= (-u_2, -u_1)^\top, \\ \widehat{C}_L &= (-C_{L,2}, -C_{L,1}, -C_{L,3})^\top, \\ \widehat{C}_D &= (C_{D,2}, C_{D,1}, C_{D,3})^\top. \end{aligned} \tag{3.1}$$

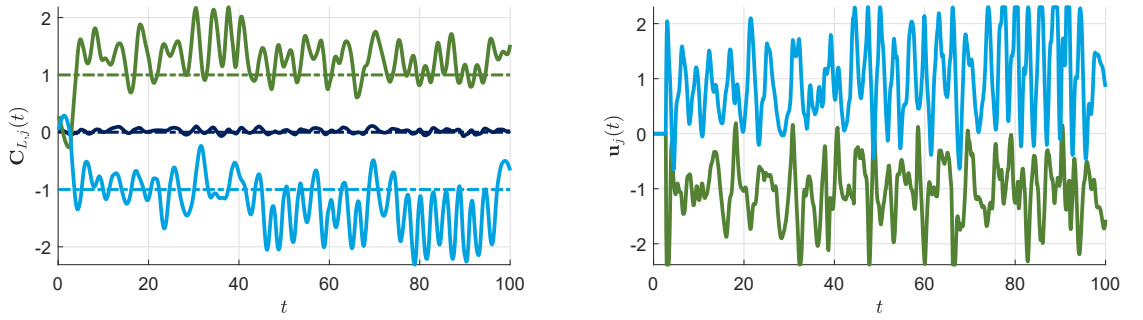
Experiments are performed for $Re \in \{100, 140, 200\}$ again, and the results are presented in Figure 3.5. For $Re = 100$, the performance is not improved. This is caused by the fact that the training data already captured the symmetry, as the system can easily switch between meta-stable states by random control inputs. The effort for $Re = 140$ is quite significant. The error is decreased by nearly 50%, and the result appears to be much more regular. Nevertheless, still, one lift is controlled more accurately than the other, probably caused by the existence of two stable solutions which are asymmetric. In the case of $Re = 200$, the improvement is even larger, but still, the performance is worse than for $Re = 140$. Presumably, this is because the oscillation around the stable states caused by the chaotic behavior occurs with larger amplitudes in the uncontrolled system and is, therefore, harder to control. Note that the presented results were generated from an RNN created in a single training run and that performance can vary significantly from one training run to another. Therefore, the performance increase should be taken cautiously and may reflect only a trend. This also becomes evident when looking at the results in the next section, where multiple training runs are performed, and a high variance of the results is observed.



(a) $Re = 100$: $e_{\text{mean}} = 0.016224$ and $e_{\text{max}} = 0.050267$



(b) $Re = 140$: $e_{\text{mean}} = 0.043693$ and $e_{\text{max}} = 0.14484$



(c) $Re = 200$: $e_{\text{mean}} = 0.14466$ and $e_{\text{max}} = 0.64097$

Figure 3.4: Results of the DeepMPC approach for various Reynolds numbers. The reference values for the lift coefficients are 1, -1 and 0 for $C_{L,1}$, $C_{L,2}$ and $C_{L,3}$, respectively. In accordance with Figure 3.2, $\mathbf{C}_{L,1}(t)$ is shown in green, $\mathbf{C}_{L,2}(t)$ in cyan and $\mathbf{C}_{L,3}(t)$ in dark blue. Accordingly, $\mathbf{u}_1(t)$ is shown in green and $\mathbf{u}_2(t)$ in cyan.

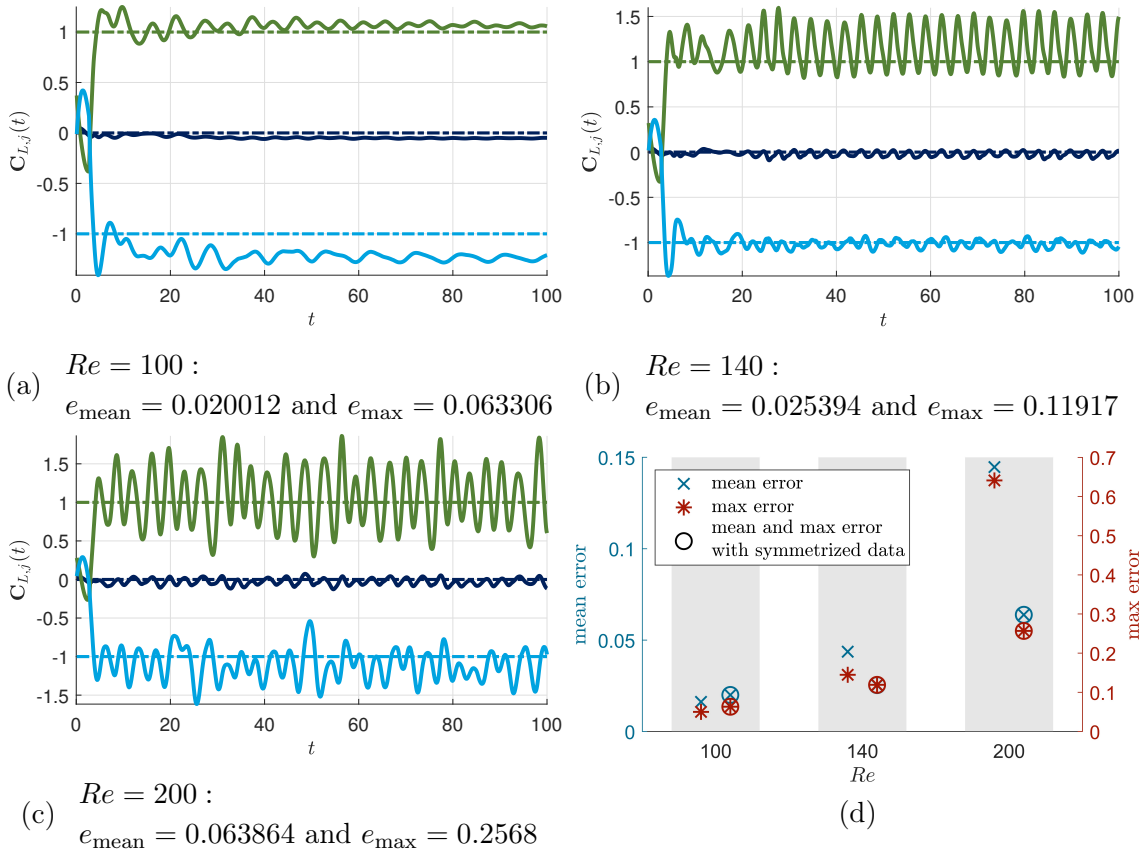


Figure 3.5: In (a)-(c): Results of the DeepMPC approach where the RNN is trained with symmetrized data according to (3.1) for varying Reynolds number. In accordance with Figure 3.2, $\mathbf{C}_{L,1}(t)$ is shown in green, $\mathbf{C}_{L,2}(t)$ in cyan and $\mathbf{C}_{L,3}(t)$ in dark blue. In (d): The mean error e_{mean} (blue) and the maximal error $e_{\max}$ (red) of the experiments for $Re \in \{100, 140, 200\}$ with original data (without circle) and symmetrized data (with circle).

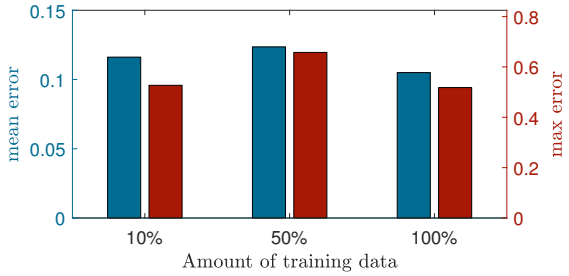


Figure 3.6: Mean error e_{mean} (blue) and maximal error $e_{\max}$ (red) for experiments at $Re = 200$ with varying amounts of training data both averaged over 5 training runs.

Reduce amount of data As it is not clear a priori how much data is needed to train the RNN to get a sufficiently accurate prediction, in the first experiments, a huge amount of data was used to be on the safe side. Obviously, it is of great interest to know whether fewer data would also be sufficient. In particular, when having in mind that the symmetry does not necessarily have to be represented in the data and can be added manually. Especially for $Re = 100$, the experiments with symmetrized data suggest that much fewer data would be sufficient. Hence, we conducted experiments varying the number of data points. We ran five experiments, each with 100%, 50%, and 10% of the data at $Re = 200$, where the training data were again doubled by exploiting the symmetry. The results are summarized in Figure 3.6. Both the maximum error and the error averaged over time only vary a little as the amount of data decreases. In particular, the differences are negligible if one additionally considers the standard deviation of 0.15 for the maximum and 0.03 for the averaged error. Probably, shorter time series already capture the essential dynamics in this case. On the other hand, this suggests that it would not be beneficial to add further data to the training process created with random control inputs to increase performance. Especially due to the unknown influence of the control input, it is nearly impossible to estimate a priori whether the essential dynamics are already captured in the training data. Perhaps the performance could be increased by using a larger RNN or by reducing the time step predicted by the RNN.

Online learning Another possibility to further increase the performance of the surrogate model, and therefore, the efficiency of the control, could be to collect additional data in an area of interest, i.e., it can be beneficial to collect the data online during the control process and do online learning, cf. Section 3.1. This is demonstrated for $Re = 100$, and the results are presented in Figure 3.7. We only collect a small amount of data offline before the execution of the control task and collect data during the online phase after a fixed time of 25 seconds, i.e., by exploiting the symmetry again, we get 500 additional data points per time period. The tracking error is reduced very effectively, especially after the first intervals. In addition, it can be observed that the control cost, averaged over the intervals of 25 seconds, decreases simultaneously, which could indicate a more stable control law produced by the DeepMPC scheme. This indicates the importance of accordingly collecting training data again.

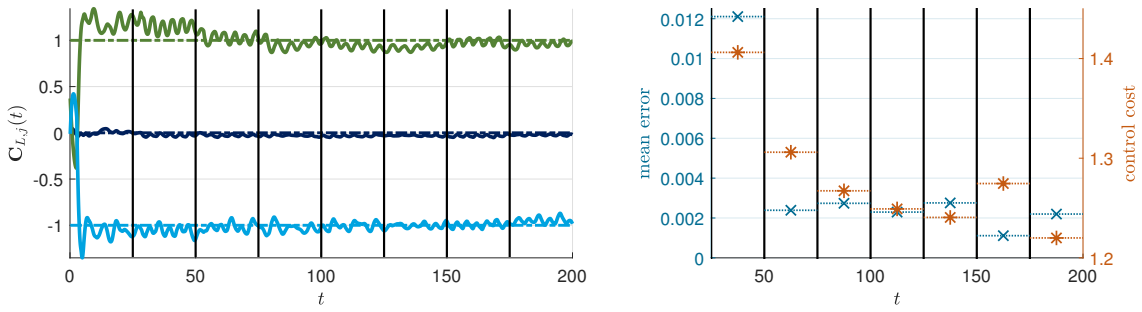


Figure 3.7: On the left, the result of the DeepMPC approach at $Re = 100$ with online learning is shown. The RNN is updated every 25 second, marked by black vertical lines. In accordance with Figure 3.2, $\mathbf{C}_{L,1}(t)$ is shown in green, $\mathbf{C}_{L,2}(t)$ in cyan and $\mathbf{C}_{L,3}(t)$ in dark blue. On the right, the mean error e_{mean} (blue) and the averaged control cost $\|u_i\|_2$ (violet) over each interval are shown.

3.3 Discussion

The presented approach, based on an RNN architecture working with sensor data, yields positive results in the studied complex flow control problem, and the performed experiments can be considered as a promising feasibility study. Even for the chaotic system at $Re = 200$, the results are convincing. Although the presented study is an important proof of concept and takes a step towards control of real systems, there remain open questions and many possibilities to further improve the approach, which are discussed in this section.

First, some basic questions concerning the RNN architecture are unclear. Although the RNN performed well, other model classes might lead to improved results. For instance, an LSTM or an ESN might predict the system state more precisely or equally well, as these are also specifically tailored to time series prediction. Furthermore, both were already used in an MPC framework [Arm+19; Igl+18; Jor+18]. Conversely, the optimization during MPC might be more expensive or less robust since highly nonlinear functions are applied to the control in these architectures. Although we did not test different architectures, the hyper-parameters of the RNN, such as the number of neurons and delay time steps or the choice of activation functions, were chosen in a trial-and-error manner. Hence, it is not clear whether these are optimal. To find optimal architectures and good choices for the hyper-parameters, *hyper-parameter optimization* [Bis+21] or *neural architecture search* [EMH19; WRP19] could be a way to go. Another aspect, which should be analyzed further, is the training procedure. Although the presented three-step approach proved to work well, other methods could also be tried. Especially, the initialization of the network parameters with the Xavier initialization, which was originally derived for sigmoid activation functions, is probably not the best choice, and one could try whether, for instance, the initialization presented in [He+15], which is specifically tailored to ReLU activation functions (as used in the presented RNN architecture), delivers improved results.

From a practical point of view, it would be of interest how the approach deals

with noisy data, which naturally occur in real applications due to measuring errors. Since modern algorithms for training NNs are particularly designed for robustness against noise, it could be expected that this would not lead to a highly decreased performance of the approach. Another question that probably arises when applying the approach to a real flow problem is how it might react if the Reynolds number slightly changes. Usually, the system dynamics should not change too much such that this would not lead to issues. Nevertheless, when the dynamical system undergoes a bifurcation, the dynamics can change abruptly, leading to wrong predictions by the RNN, which results in poor control performance. Improving online learning or transfer learning might be a suitable approach to overcome this issue. Besides the possibility to react to small changes in the system behavior, online learning provides the ability to use data collected on the fly in the relevant system regime, which can reduce the amount of data that has to be collected beforehand in a time-consuming offline procedure. Hence, it would be beneficial to improve the robustness of online learning further, especially when the system exhibits chaotic behavior.

A more general aspect is to incorporate system knowledge into the approach, i.e., to use physics-informed machine learning [Kar+21]. Obviously, it should always be advantageous to use all the knowledge available. Here, the knowledge of symmetries in the system was used for the data acquisition process. More advanced approaches include the system knowledge directly into the surrogate model. Besides better performance, this might also lead to reduced data requirements.

Despite the high relevance of all the issues mentioned above, no additional studies and experiments addressing these are done within this thesis. Instead, we focus on two other aspects. As shown in the previous section, it is not known a priori how much data is necessary to build an appropriate surrogate model. This is already the case when modeling autonomous uncontrolled systems and becomes even worse when adding a time-dependent control input. This issue is addressed in Chapter 4, where a framework is presented which is based on a discretization of the control space $U \in \mathbb{R}^n$ and only needs to build surrogate models which are able to predict the behavior of the system for a single constant control input each.

Switching from the control perspective back to the machine learning perspective, a known issue from which NNs suffer is *over-parameterization* of the model. A smaller network with fewer parameters likely leads to a similar performance. Such a model may be easier to interpret, and, more importantly, over-parameterized models are more likely to overfit to the training data. To avoid this issue, different techniques are followed. The most common are *dropout* and the regularization of the training objective with the ℓ_1- or ℓ_2-norm of the network parameters [GB10]. The latter approach is addressed in Chapter 5, where the regularization problem is transformed into an MOP. Since an NN is, in general, a nonconvex function, from a multiobjective perspective, the typically used penalty approach does not lead to all optimal compromises. Hence, a continuation method for a nonconvex objective and the ℓ_1-regularization term, which may also be used for the training of NNs, is developed.

4 | Utilizing Autonomous Models for Model Predictive Control

As discussed in the last section, one issue when using data-based methods for surrogate modeling in MPC is the question of how to generate the data. In particular, it is usually unclear in advance how much data is needed and where it should be sampled. This question is already hard to answer in the autonomous, uncontrolled case and becomes even more complicated when adding a control input. On the one hand, this is simply caused by the fact that the problem dimension increases, which usually results in more complex models and the need for a larger amount of data. On the other hand, due to the force induced by the control input, the system state is typically not restricted to a low-dimensional manifold, as it is often the case in the uncontrolled setting. A further issue when building surrogate models of dynamical systems with control input is that many modeling techniques are specifically tailored to autonomous systems. In principle, these methods can usually be used nevertheless to model controlled systems by applying state augmentation, i.e., by considering $\hat{y} = (y, u)$ as the input state. However, some approaches can only be used at the cost of a (highly) increased modeling effort. For instance, when using projection-based methods as POD, in the context of flow control problems, special attention to boundary conditions has to be paid as shown in [BCB05; GPT99], where the control of a fluid flowing around a cylinder was considered. Other examples are approaches based on the Koopman operator like eDMD, which have to be adapted to incorporate a control input, cf. Section 2.2.3. Although approaches working with the augmented state were recently developed, see, for instance, [KKB18; KM18a; PBK16], they still suffer from high demand for data [PK19].

This motivates the *QuaSiModO* approach presented in this chapter, where the control problem is adapted in such a way that autonomous models, i.e., models without additional control input, can be used to control the system efficiently. The main idea is to discretize the control space (*Quantization*) in order to construct – based on simulation data from the underlying real system (*Simulation*) – finitely many autonomous models (*Modeling*), each for one fixed control input. This results in a (mixed) integer control problem that can be solved with the help of optimization techniques from mixed-integer optimal control (*Optimization*). A similar idea was already presented in [PK19] for the Koopman operator and was further developed for the Koopman generator in [POR20]. In [PK19], the system was transformed into a *switching system*, meaning that only a finite number of control input values is

allowed. This is similar to our quantization step. The authors proved convergence of the single approximations of the Koopman operators to the real autonomous systems with fixed control input. However, the error introduced by the quantization step was not examined. In [POR20], the idea was transferred to Koopman generators. To derive error bounds for the real system, only control affine systems were considered, allowing for continuous control inputs. In this chapter, the approach is generalized to arbitrary surrogate models. Furthermore, error bounds are derived not only for control affine systems but for arbitrary nonlinear systems under the assumption that the approximation error caused by the model is known.

In the remainder of this chapter, first, the framework is explained in detail in Section 4.1. In addition, this section provides a brief overview of the *sum-up-rounding* (SUR) algorithm presented in [SBD12] which is used to solve the integer control problem. Afterwards, error bounds for the computed optimal trajectories of the original control problem and the surrogate mixed integer problem are derived in Section 4.2 justifying the approach. Subsequently, the feasibility is demonstrated with various numerical examples in Section 4.3. Although the reduced dimension of the surrogate models may intuitively lead to reduced data requirements, this is not directly clear since the considered state space remains the same and has to be captured in the model-building process. Hence, the chapter concludes with a discussion on the data requirements compared to models with control inputs based on numerical experiments in Section 4.4.

Large parts of the content of this chapter were already published in [PB23], to which both authors contributed equally. The theoretical and numerical results presented here are mainly based on the work of the author of this thesis.

4.1 The Basic Idea of the QuaSiModO Framework

As introduced in Section 2.1.2, we aim to control a dynamical system (in discrete time) given by an ODE with corresponding time-T-map Φ using MPC. As before, the control task is modeled by the objective function J or P. Since we want to consider error bounds for the open-loop problem, we start by introducing our original control problem as the optimization problem resulting from the MPC approach for the time-T-map of the underlying real system:

$$\begin{aligned} \min_{u_{0:p-1}\in U^p} J(y_{1:p}) &= \sum_{i=0}^{p-1} P(t_{i+1}, y_{i+1}), \\ \text{s.t.} \quad y_{i+1} &= \Phi(y_i, u_i), \qquad i \in \{0, \dots, p-1\}. \end{aligned} \tag{I}$$

For ease of notation, we write down the system only for the first time step. Furthermore, in contrast to Section 2.1, we assume that J and P do not depend directly on the control inputs $u_{0:p-1} = (u_i)_{i=0,\dots,p-1}$ but only on the states $y_{1:p} = (y_i)_{i=1,\dots,p}$. See Remark 4.2.6 for a discussion of the case where the objective depends on $u_{0:p-1}$ as well. In addition, we have to assume that the original control set $U \subseteq \mathbb{R}^{n_u}$ is bounded to derive meaningful error bounds. From a practical point of view, this is usually not

a strong limitation. For instance, U is often limited by box constraints. In the following, the components of the QuaSiModO framework are explained. In Figure 4.1, these steps, including different solution strategies, are summarized.

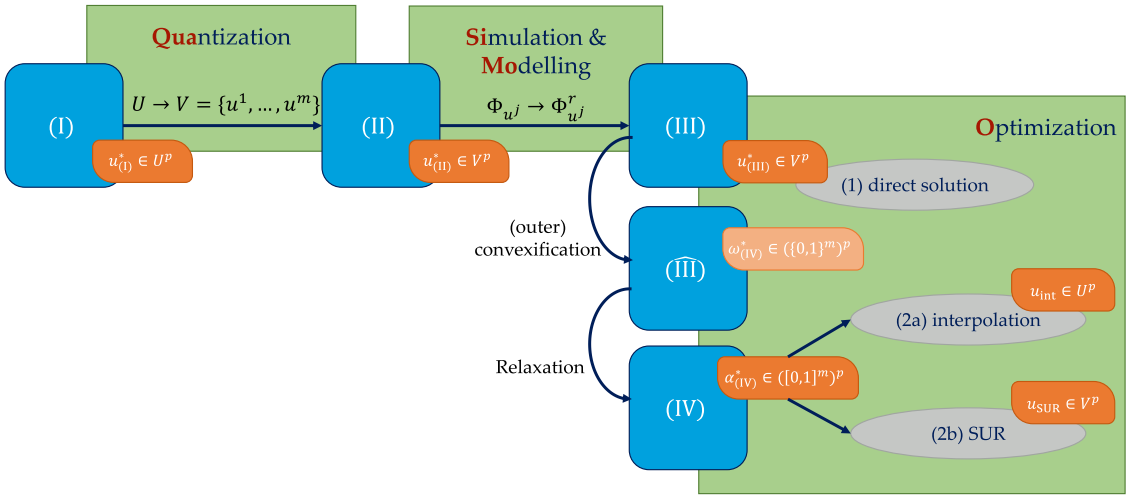


Figure 4.1: Overview of the steps followed by the presented QuaSiModO approach (green) and the different solution possibilities (grey), including the various MPC optimization problems (blue) and the resulting (optimal) solutions (orange).

Quantization The main idea of the QuaSiModO approach is to restrict the control space U to a finite subset of control inputs, i.e., we introduce the discrete control set $V = \{u^1, \dots, u^m\} \subseteq U$, $m \in \mathbb{N}$. This leads to the discretized problem

$$\begin{aligned} \min_{u_{0:p-1} \in V^p} J(y_{1:p}) &= \sum_{i=0}^{p-1} P(t_{i+1}, y_{i+1}), \\ \text{s.t.} \quad y_{i+1} &= \Phi_{u_i}(y_i), \qquad i \in \{0, \dots, p-1\}, \end{aligned} \tag{II}$$

where we write $\Phi_{u^j}(y)$ instead of $\Phi(y, u^j)$ for $j \in \{1, \dots, m\}$ and $y \in Y$. Note that u with an upper index (usually j) denotes the elements in V whereas u with a lower index (usually i) refers to the control input applied to the system at time t_i. If we consider Problem (II), it obviously also holds $u_i \in V$, i.e., for all $i \in \{0, \dots, p-1\}$, it exists $j \in \{1, \dots, m\}$ such that $u_i = u^j$. Now, we may interpret $\Phi_{u^j}(y)$ as time-T-map for the autonomous system with constant control input $u^j \in V$, which are replaced by a surrogate model in a subsequent step. Obviously, the choice of V is crucial. On the one hand, the set should be as small as possible to reduce the effort for the surrogate modeling and to allow for efficient optimization. On the other hand, it should include enough and suitable values to ensure the ability to predict the system's behavior and, hence, a good control performance. In Section 4.2, it is shown that the set of reachable states corresponding to U has to be within the set of reachable states corresponding to V to introduce no error by the quantization step, cf. Lemma 4.2.1 and Theorem 4.2.3. In the case of control affine systems where U is given by box constraints, this is ensured by choosing V to be the set of the control bounds, cf. Remark 4.2.2.

Simulation & modeling Now, the discrete control space allows us to build multiple surrogate models, one for each control $u^j \in V$, that do not get the current control as input. Hence, these models are autonomous (in contrast to a single model getting the control as an additional input). In accordance with Section 2.1.2, we refer to those surrogate models as $\Phi^r_{u^j}$ and assume that they are not acting on the full state space Y but in the observed space $Z = h_{\mathsf{obs}}(Y)$ given by measurements of the system, i.e., $\Phi^r_{u^j}(z_i) = \tilde{z}_{i+1} \approx z_{i+1} = h_{\mathsf{obs}}(\Phi(u^j, y_i))$, where $z_i = h_{\mathsf{obs}}(y_i)$ is the observed state at time t_i. Thus, instead, we consider the MPC problem

$$\begin{aligned} \min_{u_{0:p-1} \in V^p} J_{\mathsf{obs}}(\tilde{z}_{1:p}) &= \sum_{i=0}^{p-1} P_{\mathsf{obs}}(t_{i+1}, \tilde{z}_{i+1}), \\ \text{s.t.} \quad \tilde{z}_{i+1} &= \Phi^r_{u_i}(\tilde{z}_i), \qquad i \in \{0, \dots, p-1\}, \end{aligned} \tag{III}$$

where $P_{\mathsf{obs}} : [t_0, t_p] \times Z \to \mathbb{R}$ is the adapted objective function and $\tilde{z}_0 = z_0 = h_{\mathsf{obs}}(y_0)$ is the initially observed state of the system.

As discussed in Section 2.1.2, due to switching to the observed space, the objective function has to be altered as well, and we assume that it is consistent with the original objective, i.e., we assume that Equation (2.5) holds:

$$P_{\mathsf{obs}}(t, h_{\mathsf{obs}}(y)) = P(t, y) \quad \forall t \in [t_0, t_p] \text{ and } y \in Y. \tag{4.1}$$

Hence, there is no error introduced by replacing the objective.

In order to train the models, a (long) time series with random control inputs in V can be created and afterwards be split according to the applied control, leading to training sets for the respective models $\Phi^r_{u^j}$. This way, it can be ensured that (in the expected value) the entire state space is covered if the set V is appropriately chosen. Although the entire state space has still to be captured, due to the autonomous models, it is not necessary to cover the entire product space $Y \times U$ but only $Y \times V$. Intuitively, this leads to a reduced amount of required data. Nevertheless, it is unclear whether the state space is covered equally fast (and accurate) by random control inputs in V compared to random control inputs in U. Therefore, this aspect is further investigated in Section 4.4 through numerical studies.

Optimization The optimization problem resulting from using multiple autonomous models is hard to solve since it is discrete. In principle, we have three possibilities to derive a solution of Problem (III):

(1) We can solve Problem (III) directly, e.g., with dynamic programming [BD15] or by evaluating all possible solutions.

(2) We can use relaxation approaches to obtain a continuous problem, which can be solved with standard solution methods for nonlinear constrained optimization. With this solution, we can proceed in two ways:

   (2a) We can apply the corresponding interpolated control directly to the system.

(2b) We can use rounding techniques, e.g., sum-up-rounding (SUR) [SBD12], cf. Section 4.1.1, resulting in a control function with values in V.

If the number of discrete control inputs m and the MPC horizon p are small, Option (1) might be a suitable strategy. Otherwise, this possibility would not be feasible to solve the combinatorial optimization problem due to the curse of dimensionality. Therefore, we make use of relaxation techniques. More specifically, we follow the approach of (*outer*) *convexification* proposed in [Sag05] and further discussed in [SBD12]. To this end, in a first step, we derive an equivalent formulation of Problem (III) by introducing a binary control variable $\omega \in \{0,1\}^m$ determining which control in V is active, i.e., instead of $\Phi_u^r(\tilde{z})$ we write $\sum_{j=1}^m \omega_j \Phi_{u^j}^r(\tilde{z})$ where

$$\omega_j = \begin{cases} 1, & \text{if } u^j = u, \\ 0, & \text{else,} \end{cases} \qquad \text{for } j \in \{1,\dots,m\}.$$

Thus, to transform Problem (III), we replace $u_{0:p-1} \in V^p$ by the binary control variable $\omega_{0:p-1} = (\omega_i)_{i=0,\dots,p-1} \in (\{0,1\}^m)^p$ as described above. We denote the entries of ω_i by $\omega_{i,j}$, $i \in \{0,\dots,p-1\}$ and $j \in \{1,\dots,m\}$, i.e., $\omega_{i,j} \in \{0,1\}$ denotes whether u^j is applied to the system at time t_i. This leads to the following equivalent formulation of Problem (III):

$$\begin{aligned} \min_{\omega_{0:p-1} \in (\{0,1\}^m)^p} J_{\mathsf{obs}}(\tilde{z}_{1:p}) &= \sum_{i=0}^{p-1} P_{\mathsf{obs}}(t_i, \tilde{z}_{i+1}), \\ \text{s.t.} \quad \tilde{z}_{i+1} &= \sum_{j=1}^{m} \omega_{i,j} \Phi_{u^j}^r(\tilde{z}_i) \text{ and} \qquad (\widehat{\text{III}}) \\ \sum_{j=1}^{m} \omega_{i,j} &= 1, \qquad \text{for } i \in \{0,\dots,p-1\}. \end{aligned}$$

The second condition ensures that only one control input from V is considered in each time step, i.e., for every $i \in \{0,\dots,p-1\}$, it exists $j' \in \{1,\dots,m\}$ with

$$\omega_{i,j} = \begin{cases} 1, & \text{if } j = j', \\ 0, & \text{if } j \neq j', \end{cases}$$

such that

$$\sum_{j=1}^{m} \omega_{i,j} \Phi_{u^j}^r(z_i) = \Phi_{u^{j'}}^r(z_i).$$

Replacing the discrete set $\{0,1\}$ by the interval $[0,1]$, i.e., replacing $\omega_{0:p-1} \in (\{0,1\}^m)^p$ by a continuous control variable $\alpha_{0:p-1} = (\alpha_i)_{i=0,\dots,p-1} \in ([0,1]^m)^p$, yields a continuous control problem based on the discrete autonomous surrogate

models:

$$\begin{aligned} \min_{\alpha_{0:p-1}\in([0,1]^m)^p} J_{\mathsf{obs}}(\tilde{z}_{1:p}) &= \sum_{i=0}^{p-1} P_{\mathsf{obs}}(t_{i+1}, \tilde{z}_{i+1}), \\ \text{s.t.} \quad \tilde{z}_{i+1} &= \sum_{j=1}^{m} \alpha_{i,j} \Phi^r_{u^j}(\tilde{z}_i), \qquad \text{(IV)} \\ \text{and} \quad \sum_{j=1}^{m} \alpha_{i,j} &= 1, \qquad \text{for } i \in \{0,\dots,p-1\}. \end{aligned}$$

This relaxed problem can now be solved with standard techniques from nonlinear constraint optimization, e.g., with a gradient descent method or an SQP solver [NW06]. As stated above, there are now two possibilities to construct a control u from the optimal solution $\alpha^*_{(\text{IV})}$ of Problem (IV). The first one, Option (2a), is to simply calculate the convex combination given by $\alpha^*_{(\text{IV})}$, i.e.,

$$u_i = \sum_{j=1}^{m} \alpha^*_{(\text{IV})_{i,j}} u^j, \qquad \text{for } i \in \{0,\dots,p-1\}. \tag{4.2}$$

Obviously, this is only valid if the control set U is convex. Otherwise, the calculated control would not be feasible. Furthermore, this way, an interpolation error is introduced. Nevertheless, if a control affine system – or a "nearly affine" system – is considered, this is the best choice since no additional interpolation error is introduced in this case. Alternatively, rounding techniques can be used to derive a control u with values exclusively in V, cf. Option (2b). Here, we use the SUR algorithm presented in [SBD12]. Hence, a short introduction to this algorithm, including some results on error bounds that are also used to derive bounds for the presented QuaSiModO approach in Section 4.2, is given in the following subsection.

4.1.1 SUR for (Mixed) Integer Control Problems

To compute a control $u_{0:p-1}$ (nearly) optimal for Problem (III) from the optimal solution $\alpha^*_{(\text{IV})}$ of Problem (IV), we want to use rounding techniques. More precisely, the *sum-up-rounding* (SUR) algorithm, originally presented in [Sag05], is used to derive a binary control $\omega_{\mathsf{SUR}} \in (\{0,1\}^m)^p$ from $\alpha^*_{(\text{IV})}$ which can be proven to be nearly optimal for Problem ($\widehat{\text{III}}$). By identifying the binary control ω_{SUR} with a control $u_{\mathsf{SUR}} \in V^p$, we get a solution for Problem (III). In [Sag05], the SUR algorithm was introduced to solve mixed integer control problems via relaxation and subsequent rounding. It takes into account the rounding decision made in previous time steps and, in contrast to other rounding strategies, provides the *special ordered set property* for the rounded solution, i.e., the second constraint stated in Problem ($\widehat{\text{III}}$) is fulfilled by ω_{SUR}. Hence, in each time step, $\omega_{\mathsf{SUR}_i} \in \{0,1\}^m$ can uniquely be identified with

a control value in V. The rounding is done as follows:

$$\hat{\omega}_{i,j} = \sum_{k=0}^{i} \alpha^*_{(\mathrm{IV})_{k,j}} - \sum_{k=0}^{i-1} \omega_{\mathsf{SUR}k,j} \tag{4.3a}$$

$$\omega_{\mathsf{SUR}i,j} = \begin{cases} 1, & \text{if } j = \min(\{l \in \{1,\dots,m\} : \hat{\omega}_{i,l} = \max_{\rho\in\{1,\dots,m\}} \hat{\omega}_{i,\rho}\}), \\ 0, & \text{else.} \end{cases} \tag{4.3b}$$

Now, to compute the control u_{SUR}, we simply have to sum up over the $u^j \in V$, i.e.,

$$u_{\mathsf{SUR}i} = \sum_{j=1}^{m} \omega_{\mathsf{SUR}i,j} u^j, \qquad i \in \{0,\dots,p-1\}, \tag{4.3c}$$

whereby only one summand is unequal zero and, hence, $u_{\mathsf{SUR}i} \in V$.

Note that, in principle, the rounding could be done on a finer grid than the one used to optimize the control. Indeed, to guarantee the existence of a trajectory created by a discrete control that is arbitrary close to the optimal trajectory resulting from a continuous control input, it is necessary to allow for an arbitrary small *switching time* Δt_{SUR}, i.e., the time on which a constant control is applied to the system. Nevertheless, the switching time is limited by practical requirements of the real system (and in the case of numerical simulation by the time discretization used for the numerical solver). In Figure 4.2, the SUR procedure is briefly demonstrated by means of two examples.

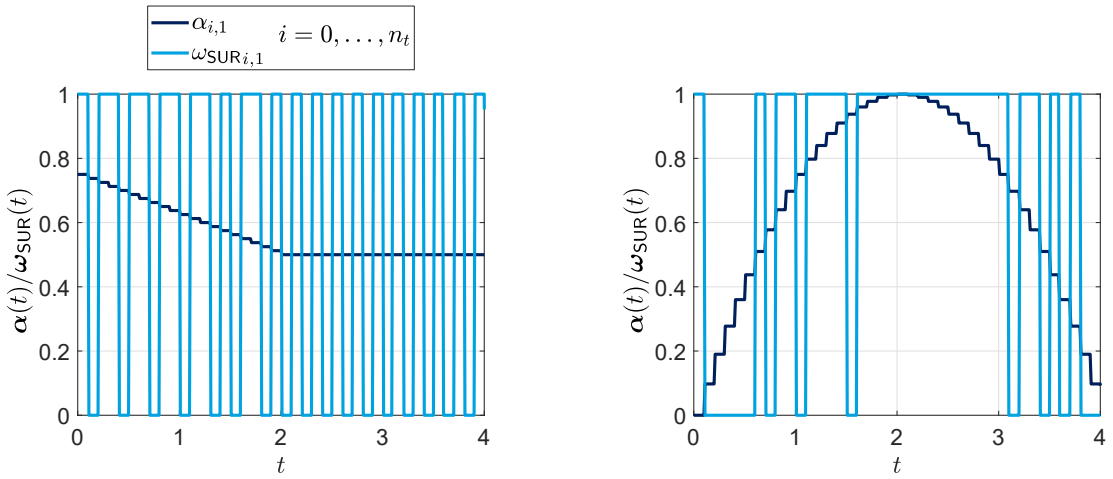


Figure 4.2: Two examples for the SUR algorithm for $V = \{u^1, u^2\}$, i.e., $m = 2$ (with discretization $\Delta t = \Delta t_{\mathsf{SUR}} = 0.1$). The first components of the relaxed control ($\alpha_{i,1}$) and the rounded control ($\omega_{\mathsf{SUR}i,1}$) are shown in dark blue and cyan, respectively. (The second component is uniquely determined by $1 - \alpha_{i,1}$ or $1 - \omega_{\mathsf{SUR}i,1}$.)

To derive error bounds for the presented approach, in [SBD12], Grönwall's inequality [Grö19] is used to show the similarity of trajectories resulting from the relaxed

control and the corresponding control derived via the SUR algorithm. Therefore, the system in continuous time is considered there. Hence, instead of the time discrete trajectories and control variables, we consider the continuous control functions $\boldsymbol{\alpha} : [t_0, t_p] \to [0,1]^m$, $\boldsymbol{\omega}_{\mathsf{SUR}} : [t_0, t_p] \to \{0,1\}^m$ and $\mathbf{u}_{\mathsf{SUR}} : [t_0, t_p] \to V$ and the respective trajectories of the time continuous system. Nevertheless, the switching time is still greater than zero, i.e., $\boldsymbol{\omega}_{\mathsf{SUR}}$ and $\mathbf{u}_{\mathsf{SUR}}$ are constant over the intervals of length Δt_{SUR} and we can still refer to these constant values by $\omega_{\mathsf{SUR}i} \in \{0,1\}$ or $u_{\mathsf{SUR}i}$. Thus, the formula for the SUR algorithm for a control $\boldsymbol{\alpha} : [t_0, t_p] \to [0,1]^m$ is given by (4.3), where (4.3a) is replaced by

$$\hat{\omega}_{i,j} = \int_{t_0}^{t_i} (\boldsymbol{\alpha}(t))_j \, dt - \sum_{k=0}^{i-1} \omega_{\mathsf{SUR}k,j} \cdot \Delta t_{\mathsf{SUR}}. \tag{4.4}$$

In [SBD12], the main result, which we use to derive the error bounds in the subsequent section for the presented approach as well, is that the trajectories resulting from the relaxed control and the corresponding control derived by the SUR algorithm are arbitrarily close if the switching time Δt_{SUR} is arbitrarily small. Assuming that J (and P) are Lipschitz continuous, this directly implies that the optimal value of J of Problem ($\widehat{\text{III}}$) (and hence of Problem (III)) is arbitrarily close to the optimal value of J in Problem (IV). To show the similarity of the two trajectories, the authors formulated the following variant of Grönwall's inequality which is a slight adaption and is used in the subsequent section to prove the error bounds.

Unless otherwise stated, in the remainder of this chapter, we denote by $\|\cdot\|$ the maximum norm $\|\cdot\|_\infty$. Furthermore, by $L^1([t_0, t_p], Y)$ we denote the Banach space of Lebesgue integrable functions mapping from the interval $[t_0, t_p]$ to the space $Y \subseteq \mathbb{R}^{n_y}$ and by $L^\infty([t_0, t_p], \mathbb{R})$ the space of essential bounded functions mapping from $[t_0, t_p]$ to $\mathbb{R}$.

Lemma 4.1.1 (Adapted Grönwall's Lemma)**.** *Let $a, b : [t_0, t_p] \to \mathbb{R}$ be two real-valued integrable functions on an interval $[t_0, t_p]$. Assume that for a constant $L \geq 0$ it holds for almost all $t \in [t_0, t_p]$*

$$a(t) \leq b(t) + L \int_{t_0}^{t} a(\tau) d\tau.$$

Then, for almost all $t \in [t_0, t_p]$, we have

$$a(t) \leq b(t) + L \int_{t_0}^{t} e^{L(t-\tau)} b(\tau) d\tau.$$

Moreover, if b belongs to $L^\infty([t_0, t_p], \mathbb{R})$, it holds for almost all $t \in [t_0, t_p]$ that

$$a(t) \leq \|b\| \, e^{L(t-t_0)}. \tag{4.5}$$

If a, in addition, is continuous, (4.5) is satisfied for all $t \in [t_0, t_p]$ (and not only for almost all t).

Proof. The proof can be found in [Ger11]. □

Based on Grönwall's inequality, the following lemma can be proven which allows us to estimate the distance between trajectories given by different time-T-maps and different control inputs. A similar idea is used, for instance, in [SBD12] (and [MK20]) to derive error bounds for the SUR algorithm. We formulate this as a separate lemma since we use the result to derive our own error bounds as well.

Note that we here (and in the subsequent theorem) consider the case where the function g describing the system dynamics is allowed to depend explicitly on the time t, although we excluded this case before. This is necessary to derive an error bound for the SUR algorithms applied to the system given by the discrete-time surrogate models, later on, cf. Lemma 4.2.10.

Lemma 4.1.2. *Let $g, \bar{g} : [t_0, t_p] \times Y \times U \to Y$ and $\mathbf{u}, \bar{\mathbf{u}} : [t_0, t_p] \to U$ be measurable functions with $Y \subseteq \mathbb{R}^{n_y}$ and $U \subseteq \mathbb{R}^{n_u}$. Furthermore, assume that $g(\cdot, \mathbf{y}(\cdot), \mathbf{u}(\cdot))$ and $\bar{g}(\cdot, \bar{\mathbf{y}}(\cdot), \bar{\mathbf{u}}(\cdot)) \in L^1([t_0, t_p], Y)$ and we define $\mathbf{y}, \bar{\mathbf{y}} : [t_0, t_p] \to Y$ by*

$$\mathbf{y}(t) = y_0 + \int_{t_0}^{t} g(\tau, \mathbf{y}(\tau), \mathbf{u}(\tau))\, d\tau \qquad \text{and}$$
$$\bar{\mathbf{y}}(t) = \bar{y}_0 + \int_{t_0}^{t} \bar{g}(\tau, \bar{\mathbf{y}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau,$$

with $y_0, \bar{y}_0 \in Y$. Moreover, assume that

$$\|\bar{g}(t, \mathbf{y}(t), \bar{\mathbf{u}}(t)) - \bar{g}(t, \bar{\mathbf{y}}(t), \bar{\mathbf{u}}(t))\| \leq L_{\bar{g}} \|\mathbf{y}(t) - \bar{\mathbf{y}}(t)\|$$

holds for almost all $t \in [t_0, t_p]$. If, in addition,

$$\sup_{t \in [t_0, t_p]} \left\| \int_{t_0}^{t} g(\tau, \mathbf{y}(\tau), \mathbf{u}(\tau)) - \bar{g}(\tau, \mathbf{y}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau \right\| \leq M,$$

then,

$$\|\mathbf{y}(t) - \bar{\mathbf{y}}(t)\| \leq (M + \|y_0 - \bar{y}_0\|) e^{L_{\bar{g}} t} \quad \forall t \in [t_0, t_p].$$

Proof. Let $t \in [t_0, t_p]$. Then,

$$\begin{aligned}
\|\mathbf{y}(t) - \bar{\mathbf{y}}(t)\| &= \left\| y_0 - \bar{y}_0 + \int_{t_0}^{t} g(\tau, \mathbf{y}(\tau), \mathbf{u}(\tau)) - \bar{g}(\tau, \bar{\mathbf{y}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau \right\| \\
&\leq \|y_0 - \bar{y}_0\| + \left\| \int_{t_0}^{t} g(\tau, \mathbf{y}(\tau), \mathbf{u}(\tau)) - \bar{g}(\tau, \mathbf{y}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau \right\| \\
&\quad + \left\| \int_{t_0}^{t} \bar{g}(\tau, \mathbf{y}(\tau), \bar{\mathbf{u}}(\tau)) - \bar{g}(\tau, \bar{\mathbf{y}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau \right\| \\
&\leq \|y_0 - \bar{y}_0\| + M + \int_{t_0}^{t} \|\bar{g}(\tau, \mathbf{y}(\tau), \bar{\mathbf{u}}(\tau)) - \bar{g}(\tau, \bar{\mathbf{y}}(\tau), \bar{\mathbf{u}}(\tau))\|\, d\tau \\
&\leq \|y_0 - \bar{y}_0\| + M + L_{\bar{g}} \int_{t_0}^{t} \|\mathbf{y}(\tau) - \bar{\mathbf{y}}(\tau)\|\, d\tau.
\end{aligned}$$

We can now apply Grönwall's lemma, cf. Lemma 4.1.1, and obtain

$$\|\mathbf{y}(t) - \bar{\mathbf{y}}(t)\| \leq (M + \|y_0 - \bar{y}_0\|) \cdot e^{L_{\bar{g}} t} \quad \forall t \in [t_0, t_p].$$

□

In the MPC context, $y_0 = \bar{y}_0$ is the current state, and we obtain a bound for the distance between two trajectories related to different systems, for instance, for the relaxed and the discrete system. Note that the bounds $L_{\bar{g}}$ and M have only to hold for the considered trajectories, not over the entire state and control space. If $\bar{g}$ is Lipschitz continuous in the second argument, the constant $L_{\bar{g}}$ can be replaced by the Lipschitz constant. Now, to estimate the distance between the optimal trajectory of the relaxed and the discrete system, we can derive a bound M and apply Lemma 4.1.2. This is also done in [SBD12], and the main results from there that we use in this chapter are summarized in the following theorem.

Theorem 4.1.3. *Let $Y \subseteq \mathbb{R}^n$ and $V = \{u^1, \dots, u^m\} \subseteq \mathbb{R}^{n_u}$. In addition, assume that $g : [t_0, t_p] \times Y \times V \to Y$, $\mathbf{y} : [t_0, t_p] \to Y$ and $\boldsymbol{\alpha} : [t_0, t_p] \to [0, 1]^m$ are measurable functions and that $\boldsymbol{\omega} : [t_0, t_p] \to \{0, 1\}^m$ is constructed from $\boldsymbol{\alpha}$ via SUR (cf. Eq. (4.3) and (4.4)) with switching time Δt_{SUR}. Then,*

$$\left\| \int_{t_0}^{t} \boldsymbol{\alpha}(\tau) - \boldsymbol{\omega}(\tau)\, d\tau \right\| \leq (m-1)\Delta t_{\mathsf{SUR}} \quad \forall t \in [t_0, t_p].$$

Furthermore, assume that $g(\cdot, \mathbf{y}(\cdot), u^j)$ is differentiable for almost all $t \in [t_0, t_p]$ and that constants C_1 and $C_2 \in \mathbb{R}$ exist for all $u^j \in V$ such that for almost all $t \in [t_0, t_p]$:

$$\left\| \frac{d}{dt} g(t, \mathbf{y}(t), u^j) \right\| \leq C_1 \qquad \text{and} \qquad \left\| g(t, \mathbf{y}(t), u^j) \right\| \leq C_2.$$

Then,

$$\sup_{t \in [t_0, t_p]} \left\| \int_{t_0}^{t} \sum_{j=1}^{m} g(\tau, \mathbf{y}(\tau), u^j)(\boldsymbol{\alpha}_j(\tau) - \boldsymbol{\omega}_j(\tau))\, d\tau \right\| \leq \underbrace{(C_2 + p\Delta t \cdot C_1)(m-1)\Delta t_{\mathsf{SUR}}}_{=: M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})}.$$

Proof. This result is proven in [SBD12]. □

Remark 4.1.4. *The bound derived in the previous theorem can be improved in two ways. First, in [SBD12], it was proven that for $m = 2$ the factor $(m-1) = 1$ can be reduced to 0.5. Furthermore, (for $m \geq 2$) m can be replaced by the maximal number of nonzero elements $\boldsymbol{\alpha}_j(t)$, $t \in [t_0, t_p]$, i.e., m can be reduced to $\max_{t \in [t_0, t_p]} |\{j : 1 \leq j \leq m \text{ and } \boldsymbol{\alpha}_j(t) \neq 0\}|$.*

Remark 4.1.5. *A similar result can be found in [MK20] but with weaker assumptions on the function g. There, the authors proved that if $g(t, \mathbf{y}(\cdot), u^j) \in L^1([t_0, t_p], Y)$ for all $u^j \in V$ and*

$$\lim_{\Delta t_{\mathsf{SUR}} \to 0} \left\| \int_{t_0}^{t} \boldsymbol{\alpha}(\tau) - \boldsymbol{\omega}(\tau)\, d\tau \right\| = 0,$$

then

$$\lim_{\Delta t_{\mathsf{SUR}} \to 0} \sup_{t \in [t_0, t_p]} \left\| \int_{t_0}^{t} \sum_{j=1}^{m} g(\tau, \mathbf{y}(\tau), u^j)(\boldsymbol{\alpha}_j(t) - \boldsymbol{\omega}_j(\tau))\, d\tau \right\| = 0.$$

Moreover, their results are not limited to ODEs, but they proved the error bound for systems given by semilinear PDEs and derived the result for ODEs as a simple, special case. Here, we use the result from [SBD12], as we want to use the concrete error bound for a specific switching time given therein.

4.2 Error Bounds

In this section, we aim to derive error bounds for the presented approach. More precisely, we consider the open-loop problems introduced in the previous section and derive a bound on the difference of the value of the objective function J resulting from the computed control input from the QuaSiModO approach and the minimal value of J, i.e., the value of J for the optimal solution $u^*_{(\mathrm{I})}$ of Problem (I). To this end, we first show that the trajectory induced by the optimal solution of (II) can be arbitrarily close to the optimal trajectory $y^*_{(\mathrm{I})}$ of (I). Afterwards, we incorporate the errors caused by the approximation of the system by a surrogate model and the different methods to solve the discretized control problem (III), i.e., by the SUR procedure or the linear interpolation of the control. Note that we do not consider specific surrogate models but leave the choice open and therefore assume that the model error is given and is not part of our study, cf. Assumption 4.2.7.

We start by showing that the optimal solutions of (I) and (II) can get arbitrarily close, i.e., we examine the error caused by the quantization step. Obviously, the optimal value $J^*_{(\mathrm{I})}$ of (I) is always at least as small as the optimal value $J^*_{(\mathrm{II})}$ of (II) as $V \subseteq U$. For the other direction, we show that for each control $\mathbf{u} : [t_0, t_p] \to U$, we can construct a sequence of controls $(\mathbf{v}_n)_{n \in \mathbb{N}}$ with $\mathbf{v}_n : [t_0, t_p] \to V$ which leads to trajectories converging to the trajectory induced by $\mathbf{u}$. In [Waż63], it was already proven that this holds if the set of points in the state space reachable with control inputs from U is a subset of the convex hull of the points reachable with control inputs in V. Here, we show the same, but more constructively. This allows us to derive concrete error bounds in the case that this assumption is not satisfied, and, more importantly, we derive a bound for the error as a function of the switching time Δt_{SUR} allowed for (II).

Although we have introduced the framework for discrete time systems in the previous section, for simplicity and in agreement with [SBD12], cf. Section 4.1.1, we prove the bounds for continuous-time systems. The error bounds for the optimization problem in discrete time then follow directly.

To this end, we utilize Lemma 4.1.2. In the MPC context, $y_0 = \bar{y}_0$ is the current state, and we obtain a bound for the distance between two trajectories related to the two MPC problems (I) and (II). Note that the given bound M only has to hold for the optimal trajectory of the original MPC problem (I), not on the entire state

space Y. In order to prove an error bound between the optimal solutions of (I) and (II), according to Lemma 4.1.2, we need to construct a control $\mathbf{v}$ with $\mathbf{v}(t) \in V$ for the optimal solution $\mathbf{u}^*$ of (I) and derive the following bound:

$$\sup_{t\in[t_0,t_p]} \left\| \int_{t_0}^{t} g(\mathbf{y}^*(\tau), \mathbf{u}^*(\tau)) - g(\mathbf{y}^*(\tau), \mathbf{v}(\tau))\, d\tau \right\| \leq M. \tag{4.6}$$

Therefore, we follow a similar idea as in Section 4.1.1 for the solution derived by SUR, i.e., we consider the (outer) relaxation of (II). Hence, we consider the trajectory of the system given by

$$\begin{aligned} \dot{\mathbf{y}}(t) &= \sum_{j=1}^{m} \boldsymbol{\alpha}_j(t) g(\mathbf{y}(t), u^j), \quad \text{i.e.,} \\ \mathbf{y}(t) &= y_0 + \int_{t_0}^{t} \sum_{j=1}^{m} \boldsymbol{\alpha}_j(\tau) g(\mathbf{y}(\tau), u^j)\, d\tau. \end{aligned} \tag{4.7}$$

We start by proving that there is a trajectory of system (4.7) which is arbitrarily close to the optimal trajectory of (I), i.e., we prove that (4.6) holds for those two trajectories with a certain bound M. The existence of a solution of (II) that is arbitrarily close to this constructed relaxed trajectory and a corresponding bound can then be proven by employing the results from the previous section.

Lemma 4.2.1. *Let $U \subseteq \mathbb{R}^{n_u}$ be bounded and $V = \{u^1, \dots, u^m\} \subseteq U$ be a finite subset. Furthermore, let $g : Y \times U \to Y$ and $\mathbf{y} : [t_0, t_p] \to Y$ be continuous and $\mathbf{u} : [t_0, t_p] \to U$ be measurable. Then, there exists a measurable function $\boldsymbol{\alpha} : [t_0, t_p] \to [0, 1]^m$ such that $\sum_{j=1}^{m} \boldsymbol{\alpha}_j(t) = 1$ $\forall t \in [t_0, t_p]$ and*

$$\sup_{t\in[t_0,t_p]} \left\| \int_{t_0}^{t} g(\mathbf{y}(\tau), \mathbf{u}(\tau)) - \sum_{j=1}^{m} g(\mathbf{y}(\tau), u^j)\boldsymbol{\alpha}_j(\tau)\ d\tau \right\| \leq p\Delta t \cdot D =: M_V,$$

where D is the maximal distance between the reachable set corresponding to U and the convex hull of the reachable set corresponding to V over the time interval $[t_0, t_p]$. More precisely,

$$D = \sup_{t\in[t_0,t_p]} \sup_{\bar{y}\in g(\mathbf{y}(t),U)} \mathrm{d}(\bar{y}, \mathrm{Conv}(g(\mathbf{y}(t), V)))$$

where d denotes the distance between a point $a \in \mathbb{R}^n$ and a set $B \subseteq \mathbb{R}^n$, i.e., $\mathrm{d}(a, B) = \inf_{b\in B} \|a - b\|$.

Proof. For every $t \in [t_0, t_p]$, let $\mathbf{y}_c(t)$ be the element in $\mathrm{Conv}(g(\mathbf{y}(t), V))$ which is closest to $g(\mathbf{y}(t), \mathbf{u}(t))$, i.e.,

$$\|g(\mathbf{y}(t), \mathbf{u}(t)) - \mathbf{y}_c(t)\| = \mathrm{d}(g(\mathbf{y}(t), \mathbf{u}(t)), \mathrm{Conv}(g(\mathbf{y}(t), V))).$$

We can write $\mathbf{y}_c(t)$ as a convex combination, i.e.,

$$\mathbf{y}_c(t) = \sum_{j=1}^{m} g(\mathbf{y}(t), u^j)\boldsymbol{\alpha}_j(t),$$

with $\boldsymbol{\alpha}(t) \in [0,1]^m$ and $\sum_{j=1}^m \boldsymbol{\alpha}_j(t) = 1$. Note that it is possible to choose $\mathbf{y}_c(t)$ and $\boldsymbol{\alpha}(t)$ such that $t \mapsto \boldsymbol{\alpha}(t)$ is measurable. Therefore, it holds for every $t \in [t_0, t_p]$

$$\begin{aligned}
&\sup_{t\in[t_0,t_p]} \left\| \int_{t_0}^{t} g(\mathbf{y}(\tau),\mathbf{u}(\tau)) - \sum_{j=1}^{m} g(\mathbf{y}(\tau),u^j)\boldsymbol{\alpha}_j(\tau)\ d\tau \right\| \\
\leq\ &\sup_{t\in[t_0,t_p]} \int_{t_0}^{t} \|g(\mathbf{y}(\tau),\mathbf{u}(\tau)) - \mathbf{y}_c(\tau)\|\ d\tau \\
\leq\ &p\Delta t \cdot \sup_{t\in[t_0,t_p]} \|g(\mathbf{y}(t),\mathbf{u}(t)) - \mathbf{y}_c(t)\| \\
\leq\ &p\Delta t \cdot \underbrace{\sup_{t\in[t_0,t_p]} \sup_{\bar{y}\in g(\mathbf{y}(t),U)} \mathrm{d}(\bar{y}, \mathrm{Conv}(g(\mathbf{y}(t),V)))}_{=D} = M_V.
\end{aligned}$$

Note that $D < \infty$ since g and $\mathbf{y}$ are continuous and U is bounded. □

Remark 4.2.2. *According to Lemma 4.2.1, V has to be chosen such that the extreme points of the reachable set corresponding to U can be reached with the control inputs $u^j \in V$ for the (optimal) trajectory $\mathbf{y}$, in order to obtain $M_V = 0$. Since the optimal trajectory is, in general, not known a priori, it is necessary to ensure this condition for all states $y \in Y$ or at least for all states in the "optimal region" of the system, which may be estimated beforehand. For arbitrary systems, it is usually challenging to determine the reachable sets. Hence, this condition is hard to prove or maybe even impossible to verify a priori. One exception is the case of control affine systems. There, only the extreme points of U have to be included in V, which is typically easy to prove. In the case of box constraints, V can simply be chosen as the corners of the box. Nevertheless, even when the system is not control affine, in many practical applications, the engineer or expert may have a feeling for which control inputs the system states changes the most.*

Using Theorem 4.1.3 to estimate the error between the relaxed and the discrete control, we can now ensure that for each continuous control $\mathbf{u} : [t_0, t_p] \to U$, there is corresponding discrete control $\bar{\mathbf{u}} : [t_0, t_p] \to V$ yielding a trajectory that is arbitrary close to the trajectory resulting from $\mathbf{u}$.

Theorem 4.2.3. *Let $U \subseteq \mathbb{R}^{n_u}$ be bounded and $V = \{u^1, \dots, u^m\} \subseteq U$ be a finite subset. Assume $g : Y \times U \to Y$, $Y \subseteq \mathbb{R}^{n_y}$, is continuous, $\mathbf{u} : [t_0, t_p] \to U$ is measurable and $\mathbf{y} : [t_0, t_p] \to Y$ is given by*

$$\mathbf{y}(t) = y_0 + \int_{t_0}^{t} g(\mathbf{y}(\tau),\mathbf{u}(\tau))\, d\tau, \quad y_0 \in Y.$$

Furthermore, let $g(\mathbf{y}(\cdot), u^j)$ be differentiable for almost all $t \in [t_0, t_p]$ and all $u^j \in V$. Moreover, assume that there exist constants C_1 and $C_2 \in \mathbb{R}$ such that for all $u^j \in V$ and for almost all $t \in [t_0, t_p]$ it holds

$$\left\| \frac{d}{dt} g(\mathbf{y}(t), u^j) \right\| \leq C_1 \qquad \text{and} \qquad \left\| g(\mathbf{y}(t), u^j) \right\| \leq C_2.$$

In addition, assume that g is Lipschitz continuous in the first argument with Lipschitz constant L_g for all $u^j \in V$. Then, for every $\varepsilon > 0$, there exists a discrete control function $\bar{\mathbf{u}} : [t_0, t_p] \to V$, such that for $\bar{\mathbf{y}}$ given by

$$\bar{\mathbf{y}}(t) = \bar{y}_0 + \int_{t_0}^{t} g(\bar{\mathbf{y}}(\tau), \bar{\mathbf{u}}(\tau))\, d\tau, \quad \bar{y}_0 \in Y,$$

it holds

$$\|\mathbf{y}(t) - \bar{\mathbf{y}}(t)\| \leq (M_V + \varepsilon + \|y_0 - \bar{y}_0\|) \cdot e^{L_g t} \quad \forall t \in [t_0, t_p].$$

Proof. Lemma 4.2.1 ensures that a measurable function $\boldsymbol{\alpha} : [t_0, t_p] \to [0,1]^m$ exists with $\sum_{j=1}^m \boldsymbol{\alpha}_j(t) = 1 \ \forall t \in [t_0, t_p]$ and $M_V > 0$, such that

$$\sup_{t \in [t_0, t_p]} \left\| \int_{t_0}^{t} g(\mathbf{y}(\tau), \mathbf{u}(\tau)) - \sum_{j=1}^{m} g(\mathbf{y}(\tau), u^j) \boldsymbol{\alpha}_j(\tau)\, d\tau \right\| \leq M_V.$$

Now, let $\boldsymbol{\omega} : [t_0, t_p] \to \{0,1\}^m$ with $\sum_{j=1}^m \boldsymbol{\omega}_j(t) = 1 \ \forall t \in [t_0, t_p]$ be the binary control derived via SUR from $\boldsymbol{\alpha}$ with switching time Δt_{SUR}. Then, according to Theorem 4.1.3, it holds

$$\sup_{t \in [t_0, t_p]} \left\| \int_{t_0}^{t} \sum_{j=1}^{m} g(\mathbf{y}(\tau), u^j)(\boldsymbol{\alpha}_j(\tau) - \boldsymbol{\omega}_j(\tau))\, d\tau \right\| \leq M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}),$$

with $M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) = (C_2 + p\Delta t \cdot C_1)(m-1)\Delta t_{\mathsf{SUR}}$. Therefore, we obtain

$$\sup_{t \in [t_0, t_p]} \left\| \int_{t_0}^{t} g(\mathbf{y}(\tau), \mathbf{u}(\tau)) - \sum_{j=1}^{m} g(\mathbf{y}(\tau), u^j) \boldsymbol{\omega}_j(\tau)\, d\tau \right\| \leq M_V + M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}). \quad (4.8)$$

Now, let $\varepsilon > 0$. Choosing the switching time Δt_{SUR} sufficiently small ensures that $M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) < \varepsilon$. Thus, inserting the control function

$$\bar{\mathbf{u}} : [t_0, t_p] \to V, \ \bar{\mathbf{u}}(t) = \sum_{j=1}^{m} \boldsymbol{\omega}_j(t) u^j$$

into (4.8) yields the bound

$$\sup_{t \in [t_0, t_p]} \left\| \int_{t_0}^{t} g(\mathbf{y}(\tau), \mathbf{u}(\tau)) - g(\mathbf{y}(\tau), \bar{\mathbf{u}}(t))\, d\tau \right\| \leq M_V + \varepsilon.$$

Thus, by applying Lemma 4.1.2, we obtain the desired result. □

Remark 4.2.4. *According to Remark 4.1.4, the factor $(m-1)$ in $M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})$ in Theorem 4.2.3 can be reduced to $(\hat{m}-1)$ with*

$$\hat{m} = \max_{t \in [t_0, t_p]} |\{j : 1 \leq j \leq m \text{ and } \boldsymbol{\alpha}_j(t) \neq 0\}|.$$

Taking the proof of Lemma 4.2.1 into account, $\hat{m}$ is limited by the number of elements in V that are actually required to represent $\mathbf{y}_c(t)$ as convex combination of $g(\mathbf{y}(t), u^j)$. According to Carathéodory's theorem, these can be at most $n_y + 1$ elements (for each $t \in [t_0, t_p]$), cf. [Gal11]. Therefore, it is likely possible to prove a similar result even for an infinite set V.

Finally, with respect to the control problems (I) and (II), we can derive a relation between the optimal values of Problems (I) and (II). Note that for the discrete system (II), the switching time Δt_{SUR} is bounded below by the time discretization Δt. Furthermore, to stay in the same time grid, Δt_{SUR} must be a multiple of Δt.

Corollary 4.2.5. *Let $u^*_{0:p-1} \in U^p$ be an optimal solution of (I) with corresponding trajectory $y^*_{1:p} \in Y^p$ for a fixed initial value $y_0 \in Y$, where $P : [t_0, t_p] \times Y \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant L_P in the second argument for all $t \in [t_0, t_p]$. Assume that the function $g : Y \times U \to Y$ defining the underlying system corresponding to the time-T-map Φ satisfies the requirements of Theorem 4.2.3 for $V \subseteq U$ and the optimal trajectory. Then, there exists a tuple $(\bar{y}_{1:p}, \bar{u}_{0:p-1}) \in Y^p \times V^p$ which is feasible for (II) for the same initial value y_0, where we allow for a switching time $\Delta t_{\mathsf{SUR}} = k \cdot \Delta t$, $k \in \mathbb{N}$, such that*

$$\left|J(y^*_{1:p}) - J(\bar{y}_{1:p})\right| \leq E_{\mathsf{lin}}(V) + E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}),$$

where

$$\begin{aligned} E_{\mathsf{lin}}(V) &= \gamma(\Delta t) \cdot L_P \cdot M_V \\ E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) &= \gamma(\Delta t) \cdot L_P \cdot M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) \end{aligned}$$

with

$$\gamma(\Delta t) = \begin{cases} \frac{e^{L_g \Delta t}\left(e^{p L_g \Delta t} - 1\right)}{e^{L_g \Delta t} - 1} & \text{if } L_g > 0, \\ p & \text{if } L_g = 0. \end{cases}$$

Proof. First, we construct $\bar{u}_{0:p-1}$. Therefore, the control $\alpha_{0:p-1} \in ([0,1]^m)^p$ of the relaxed system is chosen as in Lemma 4.2.1 and $\omega_{0:p-1} \in (\{0,1\}^m)^p$ is constructed via SUR from $\alpha_{0:p-1}$ and we define $\bar{u}_i := \sum_{j=1}^m \omega_{i,j} u^j$. Furthermore, let $\bar{y}_{1:p}$ be the discrete trajectory derived by the control $\bar{u}_{0:p-1}$. By Theorem 4.2.3 and the Lipschitz continuity of P in the second argument, we directly obtain

$$\begin{aligned} \left|J(y^*_{1:p}) - J(\bar{y}_{1:p})\right| &\leq \sum_{i=0}^{p-1} L_P \left\|y^*_{i+1} - \bar{y}_{i+1}\right\| \\ &\leq L_P(M_V + M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})) \underbrace{\sum_{i=0}^{p-1} e^{L_g t_{i+1}}}_{=\gamma(\Delta t)}. \end{aligned}$$

□

Note again that $M_V = 0$ (and hence $E_{\mathsf{lin}}(V) = 0$) for an appropriate choice of V. Furthermore, it holds

$$\lim_{\Delta t_{\mathsf{SUR}} \to 0} M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) = 0 \quad \text{and} \quad \lim_{\Delta t \to 0} \gamma(\Delta t) = p.$$

Thus, the error $E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})$ can be made arbitrarily small by choosing Δt_{SUR} arbitrary small, which requires Δt to be arbitrarily small. Moreover, note that $L_g = 0$ would imply that the current state has no influence on the trajectory, i.e., the evolution of the system only depends on the control input, which is an irrelevant special case.

Remark 4.2.6. *An error bound can also be obtained if P (and J) explicitly depend on the control u, e.g., by including the control cost $\|u\|$. In this case, however, an additional term needs to be added that pessimistically bounds the distance between the optimal control input and the switching control with values in V. In the worst case, the distance can only be bounded by the diameter of U, i.e., by $\sup_{a,b\in U} \|a-b\|$.*

Combination with model error

Until now, we only proved that the error introduced by the quantization step can be made arbitrarily small by decreasing the switching time Δt_{SUR}. To derive error bounds for the whole QuaSiModO framework, the next question is how the model error influences the solution. Afterwards, it is possible to derive error bounds for the three different solution options (1), (2a) and (2b), proposed in Section 4.1.

To this end, we need to introduce the model error for the autonomous systems. As stated before, we assume that this error is known beforehand since it is out of the scope of this thesis to derive error bounds for different surrogate models. Hence, we make the subsequent assumption. For ease of notation, we consider only a single bound, i.e., the maximum of all bounds, for all models.

Assumption 4.2.7. *In the following, we assume that $E_{\mathsf{m}} : \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ exists with*

$$\|h_{\mathsf{obs}}(\Phi_{u^j}(y)) - \Phi^r_{u^j}(\tilde{z})\| \leq E_{\mathsf{m}}(\|h_{\mathsf{obs}}(y) - \tilde{z}\|, \Delta t) \ \forall y \in Y, \tilde{z} \in Z, j \in \{1, \dots, m\}.$$

In the first time step of an MPC optimization step, the model acts usually on the real system state, i.e., $\tilde{z}_0 = z_0 = h_{\mathsf{obs}}(y_0)$. In the subsequent time steps, the model error accumulates since the model gets as input only the approximation of the observed state, i.e., we obtain the following bounds on the model error $E_{\mathsf{model},i}$ at time t_i:

$$\begin{aligned} &E_{\mathsf{model},0} = 0, \\ &E_{\mathsf{model},i} \leq E_{\mathsf{m}}(E_{\mathsf{model},i-1}, \Delta t) \quad \text{for } i \in \{1, \dots, p\}. \end{aligned} \tag{4.9}$$

Depending on the technique used to build the surrogate models, such error bounds are already available. See, for instance, [Vol11] for bounds on POD or [Nüs+21] for the modeling error of models generated via eDMD with a finite amount of data. Given the increasing interest in machine learning and data-based methods, especially in recent years, it is likely that the theory will be extended to error bounds for various surrogate modeling techniques, justifying Assumption 4.2.7 for the moment.

Now, we are able to state the different error bounds. In contrast to the previously derived results, we directly consider the difference between the value of the objective

J for the optimal control input and the controls computed within our approach. To shorten the expressions and make it easier to read, we introduce a notation for the prediction of the state by the time-T-map and its observable by the autonomous models over the MPC horizon of p time steps, i.e., we define $\mathbf{\Phi} : U^p \to Y^p$ by

$$y_{1:p} = \mathbf{\Phi}(u_{0:p-1}),$$

where $y_{1:p} = (y_i)_{i=1,\dots,p}$ and $u_{0:p-1} = (u_i)_{i=0,\dots,p-1}$ with

$$y_{i+1} = \Phi(y_i, u_i) \text{ for } i \in \{0, \dots, p-1\},$$

where Φ is the "normal" time-T-map of the system, we introduced before. Analogously, we write $\tilde{z}_{1:p} = \mathbf{\Phi}^r(u_{0:p-1})$ for $u_{0:p-1} \in V^p$ in case of the surrogate models, i.e., $\tilde{z}_{i+1} = \Phi^r_{u_i}(\tilde{z}_i)$ where we assume $\tilde{z}_0 = z_0 = h_{\mathsf{obs}}(y_0)$. First, we study the influence of the modeling error on the objective, i.e., for a fixed control $u_{0:p-1} \in V^p$, we want to derive a bound for $|J(\mathbf{\Phi}(u_{0:p-1})) - J_{\mathsf{obs}}(\mathbf{\Phi}^r(u_{0:p-1}))|$ which consists of the Lipschitz constant of the objective and the model error.

Lemma 4.2.8. *Assume we are in the setting introduced in Section 4.1 and Assumption 4.2.7 holds. Let $u_{0:p-1} \in V^p$ be an arbitrary control sequence and P_{obs} be Lipschitz continuous in the second argument with Lipschitz constant $L_{P_{\mathsf{obs}}}$ for all $t \in [t_0, t_p]$. Then,*

$$|J(\mathbf{\Phi}(u_{0:p-1})) - J_{\mathsf{obs}}(\mathbf{\Phi}^r(u_{0:p-1}))| \leq L_{P_{\mathsf{obs}}} \sum_{i=1}^{p} E_{\mathsf{model},i}$$

with $\mathbf{\Phi}$ and $\mathbf{\Phi}^r$ as defined above and $E_{\mathsf{model},i}$ as in Equation (4.9).

Proof. Let be $y_{1:p} = \mathbf{\Phi}(u_{0:p-1})$ and $\tilde{z}_{1:p} = \mathbf{\Phi}^r(u_{0:p-1})$. Then,

$$\begin{aligned}
&|J(\mathbf{\Phi}(u_{0:p-1})) - J_{\mathsf{obs}}(\mathbf{\Phi}^r(u_{0:p-1}))| \\
&= \left| \sum_{i=0}^{p-1} P(t_{i+1}, \Phi_{u_i}(y_i)) - P_{\mathsf{obs}}(t_{i+1}, \Phi^r_{u_i}(\tilde{z}_i)) \right| \\
&\overset{(4.1)}{=} \left| \sum_{i=0}^{p-1} P_{\mathsf{obs}}(t_{i+1}, h_{\mathsf{obs}}(\Phi_{u_i}(y_i))) - P_{\mathsf{obs}}(t_{i+1}, \Phi^r_{u_i}(\tilde{z}_i)) \right| \\
&\leq \sum_{i=0}^{p-1} L_{P_{\mathsf{obs}}} \left\| h_{\mathsf{obs}}(\Phi_{u_i}(y_i)) - \Phi^r_{u_i}(\tilde{z}_i) \right\| \\
&\leq \sum_{i=1}^{p} L_{P_{\mathsf{obs}}} E_{\mathsf{model},i}.
\end{aligned}$$

□

Based on this lemma, we are now able to bound the error between the value of J for the optimal solution $u^*_{(\mathrm{I})}$ of Problem (I) and the value of J that can be obtained by

the different controls computed by applying the different solution methods presented in Section 4.1. Basically, the bounds are derived by summing up the different errors using the triangular inequality. Furthermore, special attention to the case of linear observables is paid as tighter bounds can be proven in this case.

We start by analyzing the most obvious approach (1), i.e., solving Problem (III) directly. If we have a sufficiently small set V and a small number of time steps in the prediction horizon p, this option can be very efficient. The derived error bound is given in the following lemma.

Lemma 4.2.9. *Let $u^*_{(\mathrm{I})} \in U^p$ be the optimal solution of Problem (I) and $u^*_{(\mathrm{III})} \in V^p$ the optimal solution of Problem (III). Moreover, assume that Assumption 4.2.7 holds and P and P_{obs} are Lipschitz continuous in the second argument with Lipschitz constant L_P and $L_{P_{\mathsf{obs}}}$, respectively. Furthermore, assume that the function $g : Y \times U \to Y$ defining the underlying system corresponding to the time-T-map Φ satisfies the requirements of Theorem 4.2.3 for $V \subseteq U$ and the optimal trajectory of Problem (I). Then,*

$$|J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{III})}))| \leq E_{\mathsf{lin}}(V) + E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) + E_{\mathsf{r}}(E_{\mathsf{m}}), \tag{E1}$$

where $E_{\mathsf{lin}}(V)$ and $E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})$ as in Corollary 4.2.5 and

$$E_{\mathsf{r}}(E_{\mathsf{m}}) = 2L_{P_{\mathsf{obs}}} \sum_{i=1}^{p} E_{\mathsf{model},i}. \tag{4.10}$$

Proof. First, we aim at deriving a bound for $|J(\mathbf{\Phi}(u^*_{(\mathrm{III})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{II})}))|$, where $u^*_{(\mathrm{II})}$ is the (global) optimal solution of Problem (II). Since $u^*_{(\mathrm{II})}$ is the optimal solution of Problem (II), we can neglect the absolute value, i.e., it holds

$$\begin{aligned}
&\left|J(\mathbf{\Phi}(u^*_{(\mathrm{III})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{II})}))\right| = J(\mathbf{\Phi}(u^*_{(\mathrm{III})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{II})})) \\
&= \underbrace{J(\mathbf{\Phi}(u^*_{(\mathrm{III})})) - J_{\mathsf{obs}}(\mathbf{\Phi}^r(u^*_{(\mathrm{III})}))}_{\leq \left|J(\mathbf{\Phi}(u^*_{(\mathrm{III})})) - J_{\mathsf{obs}}(\mathbf{\Phi}^r(u^*_{(\mathrm{III})}))\right|} + \underbrace{J_{\mathsf{obs}}(\mathbf{\Phi}^r(u^*_{(\mathrm{II})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{II})}))}_{\leq \left|J_{\mathsf{obs}}(\mathbf{\Phi}^r(u^*_{(\mathrm{II})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{II})}))\right|} \\
&\quad + \underbrace{(J_{\mathsf{obs}}(\mathbf{\Phi}^r(u^*_{(\mathrm{III})})) - J_{\mathsf{obs}}(\mathbf{\Phi}^r(u^*_{(\mathrm{II})}))}_{\leq 0, \text{ since } u^*_{(\mathrm{III})} \text{ is global minimum of (III)}} \\
&\overset{\text{Lem. 4.2.8}}{\leq} 2L_{P_{\mathsf{obs}}} \sum_{i=1}^{p} E_{\mathsf{model},i} = E_{\mathsf{r}}(E_{\mathsf{m}}).
\end{aligned}$$

Thus, together with Corollary 4.2.5, we get the error bound

$$\left|J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{III})}))\right| \leq E_{\mathsf{lin}}(V) + E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) + E_{\mathsf{r}}(E_{\mathsf{m}}).$$

□

Since in many practical applications we may have a finite set $V = \{u^1, \ldots, u^m\} \subseteq U$ which is too large to solve the combinatorial problem (III) directly, we introduced

other solution methods utilizing the relaxed Problem (IV). First, we consider Option (2b), i.e., using the SUR algorithm to derive a control $u_{\mathsf{SUR}} \in V^p$ from the optimal solution $\alpha^*_{(\mathrm{IV})}$ of Problem (IV). Essentially, the error is composed of the error (E1) and the error introduced by utilizing SUR. To bound the latter, we need to consider the SUR error with respect to the discrete system given by the surrogate models and the relaxation of the system, i.e., the distance between the optimal trajectory of Problem (IV) and the trajectory created by the control derived from the SUR algorithm applied to the optimal control $\alpha^*_{(\mathrm{IV})}$ of Problem (IV). Therefore, we introduce an artificial trajectory in continuous time for the relaxed system considered in Problem (IV) (and also Problem ($\widehat{\mathrm{III}}$)) by

$$\tilde{\mathbf{z}}(t) := z_0 + \int_{t_0}^{t} \sum_{j=1}^{m} \boldsymbol{\alpha}_j(\tau) \cdot g^r(\tau, \tilde{\mathbf{z}}(\tau), u^j)\, d\tau \quad \text{for } t \in [t_0, t_p]$$

with

$$g^r(t, z, u^j) = \frac{\Phi^r_{u^j}(\tilde{z}_i) - \tilde{z}_i}{\Delta t} \quad \text{for } t \in [t_i, t_{i+1}) \text{ and } i \in \{0, \dots, p-1\},$$

where Δt is the step size of the discrete time-T-map Φ and $\Phi^r_{u^j}$, respectively, and $\tilde{z}_{i+1} = \sum_{j=1}^{m} \alpha_{i,j} \Phi^r_{u^j}(\tilde{z}_i)$ for $\alpha_{i,j} = \boldsymbol{\alpha}_j(t_i)$ (with $\tilde{z}_0 = z_0$). This system coincides with the discrete time systems, i.e., the systems considered in Problems ($\widehat{\mathrm{III}}$) and (IV). Since Theorem 4.1.3 is stated for time-dependent systems, it may be applied to the system defined by g^r. To this end, let $\boldsymbol{\alpha}^* : [t_0, t_p] \to [0,1]^m$ be the optimal control for Problem (IV), i.e., the function in continuous time where the constant values over time Δt are given by $\alpha^*_{(\mathrm{IV})}$, and $\boldsymbol{\omega}_{\mathsf{SUR}} : [t_0, t_p] \to \{0,1\}^m$ be the solution derived by SUR from $\boldsymbol{\alpha}^*$. Then, it holds

$$\sup_{t \in [t_0, t_p]} \left\| \int_{t_0}^{t} \sum_{j=1}^{m} g^r(\tau, \tilde{\mathbf{z}}(\tau), u^j)(\boldsymbol{\alpha}^*_j(\tau) - \boldsymbol{\omega}_{\mathsf{SUR},j}(\tau))\, d\tau \right\| \leq M_{\mathsf{SUR}^r}(\Delta t_{\mathsf{SUR}}).$$

with

$$M_{\mathsf{SUR}^r}(\Delta t_{\mathsf{SUR}}) = (C_2^r + p\Delta t \cdot C_1^r)(m-1)\Delta t_{\mathsf{SUR}}, \tag{4.11}$$

where $C_1^r = 0$ since g^r is constant for almost all $t \in [t_0, t_p]$, and

$$C_2^r = \frac{1}{\Delta t} \max_{\substack{j \in \{1,\dots,m\} \\ i \in \{0,\dots,p-1\}}} (\Phi^r_{u^j}(\tilde{z}_i) - \tilde{z}_i)$$

with $\tilde{z}_i$ being the discrete trajectory corresponding to $\alpha^*_{(\mathrm{IV})}$. Analogously to the considerations before, cf. Corollary 4.2.5, we can estimate the error between the optimal objective value of (IV) and the value of J_{obs} corresponding to the solution created using SUR by

$$E_{\mathsf{SUR}^r}(\Delta t_{\mathsf{SUR}}) = p \cdot L_{P_{\mathsf{obs}}} \cdot M_{\mathsf{SUR}^r}(\Delta t_{\mathsf{SUR}}). \tag{4.12}$$

This allows us to state the following lemma.

Lemma 4.2.10. *Let $u^*_{(\mathrm{I})} \in U^p$ and $\alpha^*_{(\mathrm{IV})} \in ([0,1]^m)^p$ be the optimal solution of Problem (I) and (IV), respectively. Furthermore, let $u_{\mathsf{SUR}} \in V^p$ be the control derived by SUR from $\alpha^*_{(\mathrm{IV})}$, cf. Equation (4.3). In addition, assume that Assumption 4.2.7 holds and P and P_{obs} are Lipschitz continuous in the second argument with Lipschitz constant L_P and $L_{P_{\mathsf{obs}}}$, respectively. Furthermore, assume that the function $g : Y \times U \to Y$ defining the underlying system corresponding to the time-T-map Φ satisfies the requirements of Theorem 4.2.3 for $V \subseteq U$ and the optimal trajectory of Problem (I). Then,*

$$\left|J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) - J(\mathbf{\Phi}(u_{\mathsf{SUR}}))\right| \leq E_{\mathsf{lin}}(V) + E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) + E_{\mathsf{r}}(E_{\mathsf{m}}) + E_{\mathsf{SUR}^r}(\Delta t_{\mathsf{SUR}}), \tag{E2b.1}$$

where $E_{\mathsf{r}}(E_{\mathsf{m}}) = 2L_{P_{\mathsf{obs}}} \sum_{i=1}^{p} E_{\mathsf{model},i}$, with $E_{\mathsf{model},i}$ as in Equation (4.9) and $E_{\mathsf{lin}}(V)$ and $E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})$ as in Corollary 4.2.5. Furthermore, $E_{\mathsf{SUR}^r}(\Delta t_{\mathsf{SUR}})$ is the error caused by the SUR procedure with respect to the system given by the surrogate models and its relaxation, cf. Equations (4.11) and (4.12).

Proof. We denote the binary control variable derived by the SUR procedure from $\alpha^*_{(\mathrm{IV})}$ by ω_{SUR}, i.e.,

$$u_{\mathsf{SUR}i} = \sum_{j=1}^{m} \omega_{\mathsf{SUR}i,j} u^j \quad \text{for } i \in \{0, \dots, p-1\}.$$

Moreover, analogous to the notation of $\mathbf{\Phi}$ and $\mathbf{\Phi}^r$, we introduce $\bar{\mathbf{\Phi}} : ([0,1]^m)^p \to Y^p$ and $\bar{\mathbf{\Phi}}^r : ([0,1]^m)^p \to Z^p$ given by

$$\begin{aligned} y_{1:p} = \bar{\mathbf{\Phi}}(\alpha_{0:p-1}) \text{ with } y_i &= \sum_{j=1}^{m} \alpha_{i,j} \Phi_{u^j}(y_i) \quad \text{for } i \in \{0, \dots, p-1\} \quad \text{and} \\ \tilde{z}_{1:p} = \bar{\mathbf{\Phi}}^r(\alpha_{0:p-1}) \text{ with } \tilde{z}_i &= \sum_{j=1}^{m} \alpha_{i,j} \Phi^r_{u^j}(\tilde{z}_i) \quad \text{for } i \in \{0, \dots, p-1\}, \end{aligned} \tag{4.13}$$

where again $\tilde{z}_0 = z_0 = h_{\mathsf{obs}}(y_0)$. Then,

$$\begin{aligned} &\left|J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) - J(\underbrace{\mathbf{\Phi}(u_{\mathsf{SUR}})}_{=\bar{\mathbf{\Phi}}(\omega_{\mathsf{SUR}})})\right| = J(\bar{\mathbf{\Phi}}(\omega_{\mathsf{SUR}})) - J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) \\ \leq\ & \underbrace{J(\bar{\mathbf{\Phi}}(\omega_{\mathsf{SUR}})) - J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\omega_{\mathsf{SUR}}))}_{\leq \frac{1}{2} E_{\mathsf{r}}(E_{\mathsf{m}})} + \underbrace{J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\omega_{\mathsf{SUR}})) - J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\alpha^*_{(\mathrm{IV})}))}_{\leq E_{\mathsf{SUR}^r}(\Delta t_{\mathsf{SUR}})} \\ &+ \underbrace{J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\alpha^*_{(\mathrm{IV})})) - J_{\mathsf{obs}}(\mathbf{\Phi}^r(u^*_{(\mathrm{III})}))}_{\leq 0} + \underbrace{J_{\mathsf{obs}}(\mathbf{\Phi}^r(u^*_{(\mathrm{III})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{I})}))}_{\leq E_{\mathsf{lin}}(V) + E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) + \frac{1}{2} E_{\mathsf{r}}(E_{\mathsf{m}}), \text{ cf. proof of Lemma 4.2.9}} \\ \leq\ & E_{\mathsf{lin}}(V) + E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) + E_{\mathsf{r}}(E_{\mathsf{m}}) + E_{\mathsf{SUR}^r}(\Delta t_{\mathsf{SUR}}), \end{aligned}$$

where $u^*_{(\mathrm{III})}$ is again the optimal solution of Problem (III). □

In case that the observable h_{obs} is linear, e.g., when considering the full state observable $h_{\mathsf{obs}}(y) = y$, we can avoid the additional error considering the SUR error in the surrogate model and, hence, derive an enhanced bound. To be more precise, the error bound is equal to (E1) derived for Option (1), i.e., for solving (III) directly.

Lemma 4.2.11. *Let $u^*_{(\mathrm{I})} \in U^p$ and $\alpha^*_{(\mathrm{IV})} \in ([0,1]^m)^p$ be the optimal solution of Problem (I) and (IV), respectively, and we assume that h_{obs} is linear. Furthermore, let $u_{\mathsf{SUR}} \in V^p$ be the control derived by SUR from $\alpha^*_{(\mathrm{IV})}$. In addition, assume that Assumption 4.2.7 holds and P and P_{obs} are Lipschitz continuous in the second argument with Lipschitz constant L_P and $L_{P_{\mathsf{obs}}}$, respectively. Furthermore, assume that the function $g : Y \times U \to Y$ defining the underlying system corresponding to the time-T-map Φ satisfies the requirements of Theorem 4.2.3 for $V \subseteq U$ and the optimal trajectory of Problem (I). Then,*

$$\left|J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) - J(\mathbf{\Phi}(u_{\mathsf{SUR}}))\right| \leq E_{\mathsf{lin}}(V) + E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) + E_{\mathsf{r}}(E_{\mathsf{m}}), \tag{E2b.2}$$

where $E_{\mathsf{r}}(E_{\mathsf{m}}) = 2L_{P_{\mathsf{obs}}} \sum_{i=1}^{p} E_{\mathsf{model},i}$, with $E_{\mathsf{model},i}$ as in Equation (4.9) and $E_{\mathsf{lin}}(V)$ and $E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})$ as in Corollary 4.2.5.

Proof. If h_{obs} is linear, we can derive a bound introduced by the surrogate model for the relaxed systems similar to Lemma 4.2.8. To this end, let $\bar{\mathbf{\Phi}} : ([0,1]^m)^p \to Y^p$ and $\bar{\mathbf{\Phi}}^r : ([0,1]^m)^p \to Z^p$ be defined as in Equation (4.13) and for $\alpha_{0:p-1} \in ([0,1]^m)^p$ let be $y_{1:p} = \bar{\mathbf{\Phi}}(u_{0:p-1})$ and $\tilde{z}_{1:p} = \bar{\mathbf{\Phi}}^r(u_{0:p-1})$. Then,

$$\begin{aligned}
&\left|J(\bar{\mathbf{\Phi}}(\alpha_{0:p-1})) - J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\alpha_{0:p-1}))\right| \\
&= \left|\sum_{i=0}^{p-1} P\left(t_{i+1}, \sum_{j=1}^{m} \alpha_{i,j}\Phi_{u^j}(y_i)\right) - P_{\mathsf{obs}}\left(t_{i+1}, \sum_{j=1}^{m} \alpha_{i,j}\Phi^r_{u^j}(\tilde{z}_i)\right)\right| \\
&\overset{(4.1)}{=} \left|\sum_{i=0}^{p-1} P_{\mathsf{obs}}\left(t_{i+1}, h_{\mathsf{obs}}\left(\sum_{j=1}^{m} \alpha_{i,j}\Phi_{u^j}(y_i)\right)\right) - P_{\mathsf{obs}}\left(t_{i+1}, \sum_{j=1}^{m} \alpha_{i,j}\Phi^r_{u^j}(\tilde{z}_i)\right)\right| \\
&\overset{h_{\mathsf{obs}}\text{ linear}}{=} \left|\sum_{i=0}^{p-1} P_{\mathsf{obs}}\left(t_{i+1}, \sum_{j=1}^{m} \alpha_{i,j} h_{\mathsf{obs}}\left(\Phi_{u^j}(y_i)\right)\right) - P_{\mathsf{obs}}\left(t_{i+1}, \sum_{j=1}^{m} \alpha_{i,j}\Phi^r_{u^j}(\tilde{z}_i)\right)\right| \\
&\leq \sum_{i=0}^{p-1}\sum_{j=1}^{m} L_{P_{\mathsf{obs}}} \cdot \alpha_{i,j} \left\|h_{\mathsf{obs}}\left(\Phi_{u^j}(y_i)\right) - \Phi^r_{u^j}(\tilde{z}_i)\right\| \\
&\leq \sum_{i=1}^{p} L_{P_{\mathsf{obs}}} E_{\mathsf{model},i} = \frac{1}{2} E_{\mathsf{r}}(E_{\mathsf{m}}),
\end{aligned}$$

where $E_{\mathsf{model},i}$ as in Equation (4.9).

Now, as in Lemma 4.2.10, we denote the binary control variable derived by the SUR procedure from $\alpha^*_{(\mathrm{IV})}$ by ω_{SUR}. Furthermore, we denote by $\bar{\alpha}^* \in ([0,1]^m)^p$ the optimal solution of the relaxed problem of (II), i.e., the system equation is given by

(4.7). Then, we derive the error bound by

$$\begin{aligned}
&\left|J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) - J(\mathbf{\Phi}(u_{\mathsf{SUR}}))\right| = J(\bar{\mathbf{\Phi}}(\omega_{\mathsf{SUR}})) - J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) \\
= \;& \underbrace{J(\bar{\mathbf{\Phi}}(\omega_{\mathsf{SUR}})) - J(\bar{\mathbf{\Phi}}(\alpha^*_{(\mathrm{IV})}))}_{\leq E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})} + \underbrace{J(\bar{\mathbf{\Phi}}(\alpha^*_{(\mathrm{IV})})) - J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\alpha^*_{(\mathrm{IV})}))}_{\leq \frac{1}{2}E_{\mathsf{r}}(E_{\mathsf{m}})} \\
&+ \underbrace{J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\alpha^*_{(\mathrm{IV})})) - J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\bar{\alpha}^*))}_{\leq 0} + \underbrace{J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}^r(\bar{\alpha}^*)) - J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}(\bar{\alpha}^*))}_{\leq \frac{1}{2}E_{\mathsf{r}}(E_{\mathsf{m}})} \\
&+ \underbrace{J_{\mathsf{obs}}(\bar{\mathbf{\Phi}}(\bar{\alpha}^*)) - J(\mathbf{\Phi}(u^*_{(\mathrm{I})}))}_{\leq E_{\mathsf{lin}}(V)} \\
\leq \;& E_{\mathsf{lin}}(V) + E_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}}) + E_{\mathsf{r}}(E_{\mathsf{m}})
\end{aligned}$$

where $u^*_{(\mathrm{III})}$ is again the optimal solution of Problem (III). □

Finally, the third option is to solve (IV) and directly apply the interpolated control to the original system, cf. Equation (4.2). Obviously, this is only feasible if $\mathbf{Conv}(V) \subseteq U$. Furthermore, we introduce an additional error caused by the linear interpolation if the system is not control affine which may be arbitrary large. Nevertheless, many real systems are control affine and in this case, we can prove the bound presented in the following lemma provided that the observable is linear as well.

Lemma 4.2.12. *Let $u^*_{(\mathrm{I})} \in U^p$ be the optimal solution of Problem (I) where we assume that the system is control affine and h_{obs} is linear. Moreover, let $\alpha^*_{(\mathrm{IV})} \in ([0,1]^m)^p$ be the optimal solution of Problem (IV) and assume $\mathbf{Conv}(V) \subseteq U$. Let $u_{\mathsf{int}} \in U^p$ be the control derived by interpolation from $\alpha^*_{(\mathrm{IV})}$, cf. Equation (4.2). In addition, assume that Assumption 4.2.7 holds and P and P_{obs} are Lipschitz continuous in the second argument with Lipschitz constant L_P and $L_{P_{\mathsf{obs}}}$, respectively. Furthermore, assume that the function $g : Y \times U \to Y$ defining the underlying system corresponding to the time-T-map Φ satisfies the requirements of Lemma 4.2.1 for $V \subseteq U$ and the optimal trajectory of Problem (I). In addition, assume that g is Lipschitz continuous in the first argument with Lipschitz constant L_g for all $u^j \in V$. Then,*

$$\left|J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) - J(\mathbf{\Phi}(u_{\mathsf{int}}))\right| \leq E_{\mathsf{lin}}(V) + E_{\mathsf{r}}(E_{\mathsf{m}}), \tag{E2a}$$

where $E_{\mathsf{r}}(E_{\mathsf{m}}) = 2L_{P_{\mathsf{obs}}} \sum_{i=1}^{p} E_{\mathsf{model},i}$, with $E_{\mathsf{model},i}$ as in Equation (4.9) and $E_{\mathsf{lin}}(V)$ as in Corollary 4.2.5.

Proof. Let $\bar{\mathbf{\Phi}} : ([0,1]^m)^p \to Y^p$ be defined as in Equation (4.13). Since the system is control affine, it holds $\mathbf{\Phi}(u_{\mathsf{int}}) = \bar{\mathbf{\Phi}}(\alpha^*_{(\mathrm{IV})})$. Hence, as in the proof of Lemma 4.2.11, it follows

$$\left|J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) - J(\mathbf{\Phi}(u_{\mathsf{int}}))\right| = J(\bar{\mathbf{\Phi}}(\alpha^*_{(\mathrm{IV})})) - J(\mathbf{\Phi}(u^*_{(\mathrm{I})})) \leq E_{\mathsf{lin}}(V) + E_{\mathsf{r}}(E_{\mathsf{m}}).$$

□

Note that, in contrast to the previously derived error bounds, only a part of the requirements of Theorem 4.2.3 have to hold for Lemma 4.2.12. More precisely, the assumptions for Lemma 4.2.1 have to be satisfied and g has to be Lipschitz continuous. Due to the direct interpolation, the bound $M_{\mathsf{SUR}}(\Delta t_{\mathsf{SUR}})$ is not involved in the error estimate, and thus, the assumptions concerning g from Theorem 4.1.3 can be ignored, i.e., the constants C_1 and C_2 do not have to exist.

The derived bounds are summarized in Table 4.1, along with an indication of whether control affinity of the system and linearity of the observable h_{obs} must be assumed.

Table 4.1: Summary of error bounds corresponding to the various solution methods.

Approach		Error bound	Control affine	h_{obs} linear	Type of optimization
Opt. (1)	– direct solution	Eq. (E1)	—	—	combinatorial
Opt. (2b)	– SUR	Eq. (E2b.1)	—	—	continuous
		Eq. (E2b.2)	—	✓	continuous
Opt. (2a)	– interpolation	Eq. (E2a)	✓	✓	continuous

4.3 Numerical Experiments

To justify the approach from a numerical perspective, it is tested on several control systems with different surrogate modeling techniques. As a first example, the Lorenz system with approximations of the respective Koopman operators via eDMD, cf. Section 2.2.3, is presented in Section 4.3.1. There, the main focus lies on the difference between affine control systems and systems with nonlinear control input. Afterwards, in Section 4.3.2, the Mackey-Glass equation is considered, which is an example of a DDE. There, ESNs are used as surrogate models, cf. Section 2.2.2. Due to the memory incorporated into the reservoir, special attention to the use and initialization of the reservoir has to be paid. Hence, a brief explanation of the exact use of ESNs in the QuaSiModO framework is added. In Section 4.3.3, we conclude with a flow control example where the surrogate models are LSTMs, cf. Section 2.2.2.

In all examples, the aim is to track a given reference trajectory z_{ref} in the observed space $h_{\mathsf{obs}}(Y) \subseteq Z$. To achieve this, the relaxed Problem (IV) is solved and depending on the followed solution strategy the corresponding interpolated or the rounded control is applied to the considered system, i.e., Option (2a) or (2b) is used, respectively.

4.3.1 Lorenz System & Koopman Operator:

We first consider the Lorenz system, which, despite its simple equations, exhibits chaotic behavior and is probably one of the most commonly considered systems

for studying chaotic behavior. Numerous studies show that the prediction of the system over a moderate time horizon with the help of different data-based methods is possible. For instance, NNs are used in [SM19; Zha17] and sparse regression in [BPK16; KKB18].

We consider the Lorenz system with an additive control term in the second dimension, i.e., the system given by the ODE

$$\frac{d}{dt}\begin{pmatrix}\mathbf{y}_1\\ \mathbf{y}_2\\ \mathbf{y}_3\end{pmatrix} = \begin{pmatrix}\sigma(\mathbf{y}_2-\mathbf{y}_1)\\ \mathbf{y}_1(\rho-\mathbf{y}_3)-\mathbf{y}_2\\ \mathbf{y}_1\mathbf{y}_2-\beta\mathbf{y}_3\end{pmatrix} + \begin{pmatrix}0\\ \mathbf{u}\\ 0\end{pmatrix} \tag{4.14}$$

with $(\sigma,\rho,\beta) = (10, 28, \frac{8}{3})$. Moreover, the full state is observed, i.e., for a state $y \in Y$ the observed state is given by $z = h_{\mathsf{obs}}(y) = y$. The control task is to force the second variable to the reference trajectory given by $\mathbf{z}_2^{\mathsf{ref}}(t) = \mathbf{y}_2^{\mathsf{ref}}(t) = 1.5 \cdot \sin(4\pi \cdot t/T_{\mathsf{MPC}})$ over the time interval $[0, T_{\mathsf{MPC}}]$ with $T_{\mathsf{MPC}} = 20.0$. Hence, the objective minimized in the optimization step at time t_i is given by

$$P(t_{i+1}, y_{i+1}) = \left\|\mathbf{y}_2^{\mathsf{ref}}(t_{i+1}) - y_{i+1,2}\right\|_2^2,$$

where $y_{i,2}$ denotes the second dimension of the state at time t_i. The control sets are given by $U = [-50, 50]$ and $V = \{-50, 50\}$, respectively. To construct the surrogate models, we use approximations of the Koopman operator via eDMD, cf. Section 2.2.3. As training data, a time series over 100 seconds using piecewise constant inputs from V is collected, resulting in 2000 data points since the step size for the model is chosen to be $\Delta t = 0.05$. For the dictionary in eDMD monomials up to degree 3 are considered. The time horizon for the MPC problem is chosen to be $p = 3$. The detailed parameters of the setting are summarized in Table 4.2.

Since the considered system is control affine and we observe the full state, the interpolated control resulting from the solution of Problem (IV) can directly be applied to the real system without introducing an additional interpolation error, cf. Equation (E2a). Hence, the excellent performance of the interpolated control, shown in Figure 4.3a in dark blue, was to be expected. The control input derived by the SUR strategy leads to a slightly worse performance caused by the switching time $\Delta t_{\mathsf{SUR}} = 5 \cdot 10^{-4}$. Note that, due to the relatively short switching time, the single control values cannot be identified in Figure 4.3a and, instead, appear as an area in cyan in the plot.

In a second step, we now want to study the effect of nonlinear control inputs. Therefore, we slightly adapt the Lorenz system (4.14) by a nonlinear substitution of the control input, namely, the second system equation is replaced by

$$\dot{\mathbf{y}}_2 = \mathbf{y}_1(\rho - \mathbf{y}_3) - \mathbf{y}_2 + 50 \cdot \cos(\mathbf{u}), \tag{4.15}$$

and we consider $U = [0, \pi]$ as control set. For the quantization step, we choose $V = \{0, \pi\}$ to ensure that $M_V = 0$ in Lemma 4.2.1 still holds. Due to the nonlinear control input, an additional interpolation error is introduced by directly applying the linear interpolated control computed from the solution of Problem (IV) to the system. Therefore, the control calculated with the SUR algorithm yields a significant increase in the performance, cf. Figure 4.3b.

Table 4.2: Lorenz system with Koopman-based surrogate model.

	Parameter	Value
System parameters	(σ, ρ, β)	$(10, 28, \frac{8}{3})$
Quantization	U	$[-50, 50]$ or $[0, \pi]$
	V	$\{-50, 50\}$ or $\{0, \pi\}$
	m	2
Training data	Δt_{ODE}	$5 \cdot 10^{-4}$
	T_{train}	100.0
	Input	Piecewise constant, random $u_i \in V$
Surrogate model	Δt	$100 \cdot \Delta t_{\mathsf{ODE}} = 0.05$
	observable	$z = h_{\mathsf{obs}}(y) = y$
	ψ	Monomials up to degree 3
MPC	T_{MPC}	20.0
	p	3
	reference	$\mathbf{y}_2^{\mathsf{ref}}(t) = 1.5 \cdot \sin(4\pi \cdot t / T_{\mathsf{MPC}})$
	$P(t_i, y_i)$	$\left\|\mathbf{y}_2^{\mathsf{ref}}(t_i) - y_{2,i}\right\|_2^2$
	Δt_{SUR}	$5 \cdot 10^{-4}$

4.3.2 Mackey-Glass Equation & ESN

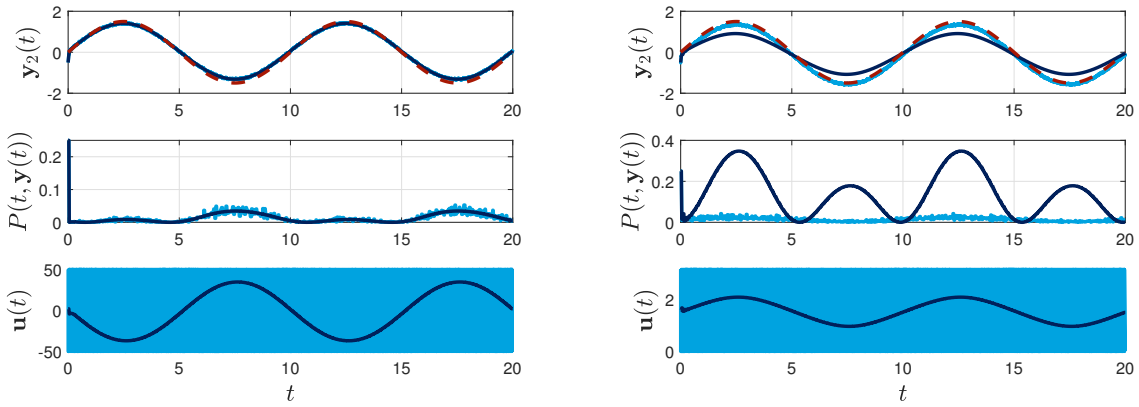
As a second example, we consider a system described by a DDE. The Mackey-Glass equation models the reproduction of blood cells [GM79; MG77] and is again equipped with an additive control term, which models an increase or decrease in the number of blood cells caused by, e.g., a transfusion or an injury. The system is given by

$$\dot{\mathbf{y}}(t) = \beta \frac{\mathbf{y}(t-\tau)}{1 + \mathbf{y}(t-\tau)^{\eta}} - \gamma \mathbf{y}(t) + \mathbf{u}(t) \quad \text{with } \beta, \gamma, \eta > 0, \tag{4.16}$$

where the state function $\mathbf{y}$ can be seen as a measure of the amount of blood cells and the control $\mathbf{u}$ as the amount of blood cells added or removed from outside to the body. The control task is to stabilize the blood flow by achieving a constant value for $\mathbf{y}$. Similar control problems were already studied in [KR17; SDL11], where the authors derived different feedback laws to stabilize the systems.

The uncontrolled system ($\mathbf{u}(t) = 0$) has been studied for various parameters and chaotic behaviour was proven among other choices for $(\beta, \gamma, \eta) = (2, 1, 9.65)$ [GM79] which is the choice of parameters considered here. Similar to the Lorenz system for ODEs, the Mackey-Glass equation was studied a lot in the context of chaotic delay systems, and there are numerous works on the prediction of the system state with the help of different data-based methods, for instance, by different forms of neural networks [FLK19; Lóp+16; MBS93; Zha+14] including ESNs [GSS15; HL06; Jae10].

Here, we want to use ESNs as surrogate models as well since these are particularly well suited to capture the dynamics caused by the delay due to the recurrent con-



(a) Control affine Lorenz system (4.14).

(b) Lorenz system with nonlinear control input (4.15).

Figure 4.3: Results for the Lorenz system generated by Koopman-based surrogate models derived by eDMD for (a) the control affine case and (b) with nonlinear control input. The result obtained by the linearly interpolated control is shown in dark blue, and the result generated via SUR after each optimization step in cyan. The reference trajectory $\mathbf{y}_2^{\text{ref}}$ is given by the red dashed line in the upper plots.

nections in the reservoir without explicitly using delay coordinates, cf. Section 2.2.2. Moreover, it can be nicely integrated into the QuaSiModO framework. Recall that the equations of an ESN predicting the state y_{k+1} are given by

$$\begin{aligned} r_{k+1} &= \sigma(W^{\mathsf{in}} u_k + W^{\mathsf{res}} r_k + W^{\mathsf{fb}} y_k), \\ y_{k+1} &= W^{\mathsf{out}} r_{k+1}, \end{aligned}$$

where for uncontrolled systems, the input term $W^{\mathsf{in}} u_k$ is usually omitted.

There are some papers discussing the use of ESNs in an MPC context, for instance, [Arm+19; Jor+18]. Therein, the control input u_k at time t_k serves as additional input to the reservoir. This causes an increased input dimension, and indirectly, a larger amount of training data is needed, as it is shown in the subsequent section for the Lorenz system. By using the QuaSiModO framework, the input dimension no longer increases since we train an individual ESN for each control $u^j \in V$. As only the various output layers W_j^{out} contain information corresponding to the control u^j, we can use the same reservoir for every ESN, cf. Figure 4.4. In addition, this way, it is ensured that the reservoir state r_k, saving the system dynamics of the past steps, is updated correctly in every time step. The training is done similarly to the standard training of ESNs described in Section 2.2.2, i.e., a long trajectory created with random control inputs in V is computed and mapped to a trajectory of reservoir states by the randomly initialized reservoir. Then, the two trajectories are divided into data sets corresponding to the different controls $u^j \in V$, i.e., if $u_k = u^j$, then r_{k+1}, derived from y_k, and y_{k+1} are data points in the j-th set. Based on these data sets, for each output layer W_j^{out}, the corresponding linear problem is solved.

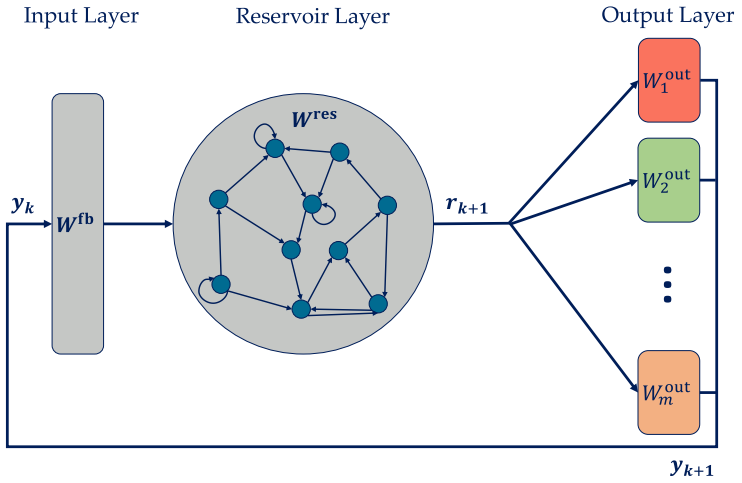


Figure 4.4: Schematic of an ESN with different readout layers corresponding to different autonomous systems.

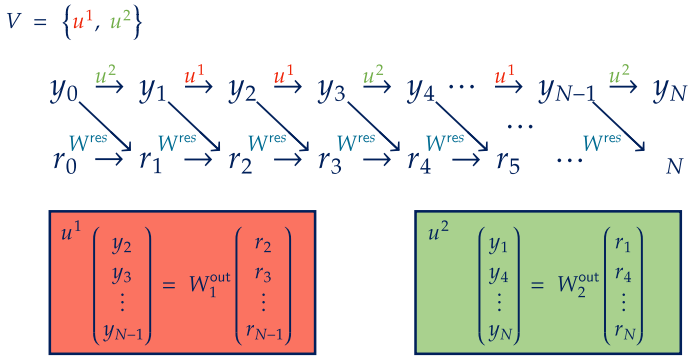


Figure 4.5: Visualization of the data processing for ESNs in the QuaSiModO framework.

As explained in Section 2.2.2, the prediction of the ESNs highly depends on the reservoir state as it incorporates the past (observed) system states. Hence, usually, the first time steps of the training data are used to initialize the reservoir state and not to optimize the output layer. This is done here as well. Furthermore, the prediction of the ESN usually follows directly after the last time step of the training data, i.e., the training data initialize the reservoir state for the prediction task. Therefore, in the MPC context, we have two possibilities. The first one is to start the MPC task with the last training data point. The second option is to start with an initialization phase, where the system is not controlled or at least not controlled by means of MPC based on the ESN model.

The control task which should be solved is the stabilization of the blood flow at $y^{\text{ref}} = 1.0$ with control inputs in $U = [-0.1, 0.5]$. The remaining parameter choices are summarized in Table 4.3. Since the control enters linearly, similar to the Lorenz system before, there is no additional error introduced by the interpolation. The obtained results are presented in Figure 4.6 and show that the system can be stabilized at 1.0 using the interpolated control.

Table 4.3: Mackey-Glass DDE with ESN surrogate model.

	Parameter	Value
System paramters	$(\beta, \gamma, \eta, \tau)$	$(2, 1, 9.65, 2)$
Quantization	U	$[-0.1, 0.5]$
	V	$\{-0.1, 0.5\}$
	m	2
Training data	Δt_{DDE}	0.1
	T_{train}	600.0
	Input	Random $u_i \in V$
Surrogate model	Δt	0.1
	observable	$z = h_{\mathsf{obs}}(y) = y$
	size residuum	1000
	spectral radius (W^{res})	1.5
	sparsity (W^{res})	0.3
MPC	T_{MPC}	20.0
	p	5
	$P(t_i, y_i) = P(y_i)$	$\|y_i - 1.0\|_2^2$

4.3.3 Kármán Vortex Street & LSTM

To conclude the numerical evaluation with a more complex system, we consider a flow control example, similar to the fluidic pinball in Section 3.2. In contrast to the fluidic pinball, this time, we consider the flow around a single cylinder instead of three cylinders. Despite that, the setting is the same as described in Section 3.2 and is shown in Figure 4.7. The system is again simulated via the OpenFOAM solver. It exhibits periodic vortex shedding at $Re = 100$, i.e., the so-called *von Kármán vortex street* can be observed [Dea+91; Noa+03].

Similar to Section 3.2, the aim is to control the flow field by only observing the forces acting on the cylinder (the lift C_L and the drag C_D) without any knowledge of the flow field itself using the rotation of the cylinder, i.e., the control u is the velocity of the cylinder rotation. As before, the objective is given by a reference trajectory for the lift coefficient of the cylinder $\mathbf{C}_L^{\mathsf{ref}}(t)$.

This setting was already considered in [Bie+20], but there the architecture presented in Chapter 3 was applied, i.e., an RNN which gets the control as additional input was used to build the surrogate model. Here, we use LSTMs for the single autonomous models. The detailed parameters of the setting are summarized in Table 4.4.

As the results in Figure 4.8 show, we achieve a nearly perfect tracking of the reference trajectory and observe again that the interpolated control leads to better results than the rounded one. The results by the SUR could be improved by choosing a shorter switching time which is infeasible here since it is already equal to the time discretization of the simulation.

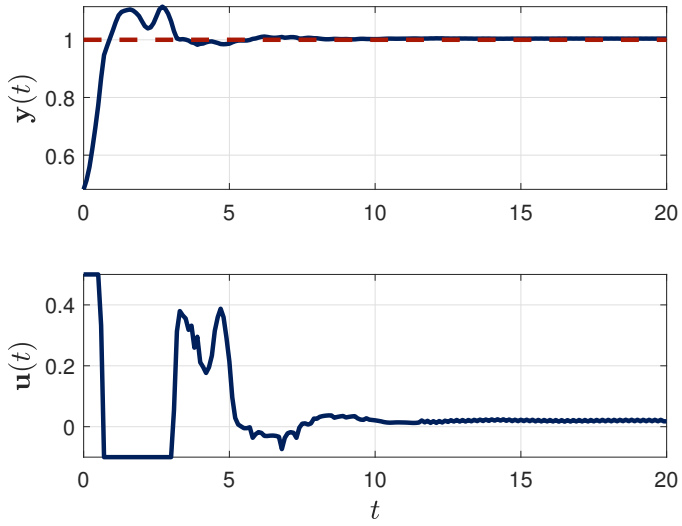


Figure 4.6: Results for the control of the Mackey-Glass system where the surrogate models are ESNs. The result obtained by the linearly interpolated control is shown in dark blue. The reference trajectory $y^{\text{ref}} = 1.0$ is given by the red dashed line in the upper plot.

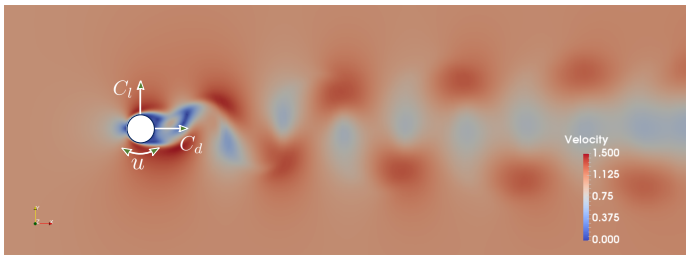


Figure 4.7: Setting of the cylinder surrounded by a fluid flowing from the left to the right with a snapshot of the velocity of the uncontrolled system. The colorbar shows the color corresponding to the norm of the velocity vector. Furthermore, the forces acting on the cylinder, i.e., the lift and the drag coefficients, as well as the control inputs, are specified.

4.4 Numerical Experiments on Data Efficiency

Although the results presented in the previous section give a numerical verification for the presented framework, the question of whether it is beneficial in comparison to the intuitive approach based on the augmented state, i.e., using the control as an additional input to the model, remains still open. This aspect is addressed in this section.

As already discussed in the introduction of this chapter, there are some methods, especially projection-based methods such as POD, which are specifically tailored to autonomous systems. Hence, it may be difficult to incorporate a control input. This is avoided by the presented approach. Furthermore, existing error bounds for the autonomous case may be directly transferred and combined with the error bounds derived in Section 4.2. The other important motivation for the QuaSiModO

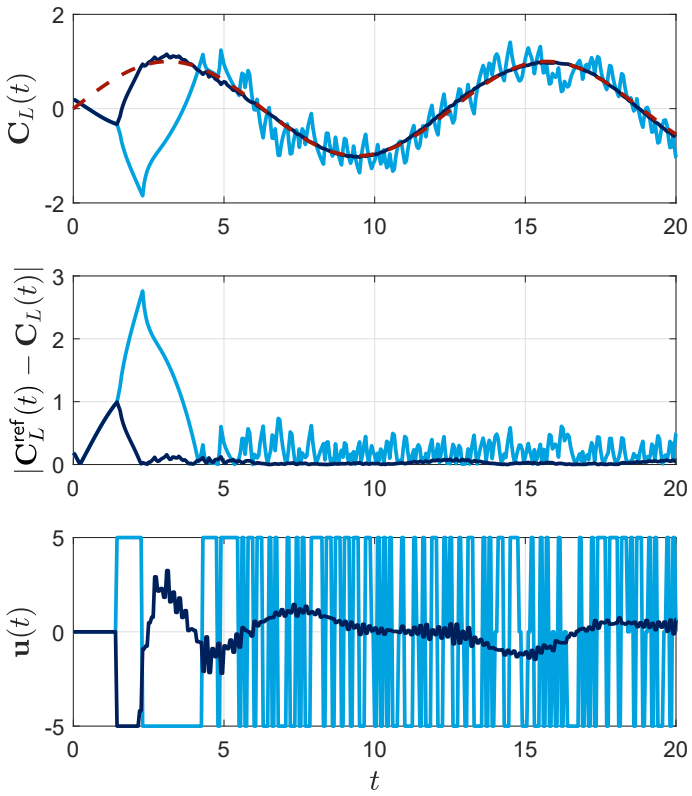


Figure 4.8: Results for the control of the flow around a single cylinder where the surrogate models are LSTMs. The result obtained by the linearly interpolated control is shown in dark blue and the result generated via SUR after each optimization step in cyan. The reference trajectory $\mathbf{C}_L^{\text{ref}}$ is given by the red dashed line in the upper plot.

Table 4.4: 2D Cylinder flow with LSTM surrogate model.

	Parameter	Value
System parameters	Re	100
Quantization	U	$[-5, 5]$
	V	$\{-5, 0, 5\}$
	m	3
Training data	Δt_{PDE}	0.01
	T_{train}	$2 \cdot 500$
	Input	Piecewise constant, random $u_i \in V$
Surrogate model	Δt	0.1
	observable	$z = h_{\mathsf{obs}}(y) = (C_D, C_L)$
	# delay coordinates	15
	Neurons per LSTM-cell	500
	batch size	75
	# epochs	2
MPC	T_{MPC}	20.0
	p	5
	reference	$\mathbf{C}_L^{\mathsf{ref}}(t) = \sin(\frac{1}{2}t)$
	$P(t_i, z_i)$	$\left\|\mathbf{C}_L^{\mathsf{ref}}(t_i) - C_{L,i}\right\|_2^2$
	Δt_{SUR}	0.1

approach is the data sampling process. First, due to a smaller control space (V instead of U), it may become easier to choose data points, i.e., one can choose random control inputs in V and create a long trajectory. More important, however, is the question of how much data is needed to achieve an acceptable control performance for both methods in comparison. Intuitively, due to the decreased dimensionality, the problem complexity is reduced for the autonomous systems, and we would assume that less data is needed. Unfortunately, the answer to this question is more complicated. Due to the control input, the dynamics are usually no longer restricted to a low-dimensional subspace, and for both approaches, the entire state space has to be somehow represented in the data. On the one hand, by the QuaSiModO approach, we do not need to capture the dynamics for the entire control space U but only for V, which leads to a decreased dimensionality of the input space of the model. This gives us the hope that less data is needed. On the other hand, we cannot assume that the reachable state space of the system is known beforehand, and hence, we may use random control inputs from U or V to move through it. There is a chance that it is a lot harder (or even impossible if the requirements of Lemma 4.2.1 are not satisfied) to capture the state space by using controls from V than from U leading indirectly to a larger amount of data needed for the training. This issue is addressed and attempted to be answered, at least partially, by the following presented numerical studies.

In the experiments, we consider again the Lorenz system, cf. Section 4.3.1, and study the prediction error of different ESN and LSTM models, cf. Section 2.2.2

and Section 4.3.2 or Section 4.3.3, respectively. As in Section 4.3.1, we consider $U = [-50, 50]$ and $V = \{-50, 50\}$ for the control affine Lorenz system (4.14) and $U = [0, \pi]$ and $V = \{0, \pi\}$ for the system with nonlinear control input (4.15). To compare the performance of both approaches, we train the models with training data sets of varying size

(i) for the autonomous systems corresponding to the controls $u^j \in V$, and

(ii) for the original system based on the augmented state $\hat{y} = (y, u)$, i.e., with the control as additional input to the model.

We refer to the second model as *full model*. To train the models, we collect time series created by random control inputs in U and V, respectively. Since the performance is subjected to random influences, e.g., the created training data, the initialization of the reservoir of the ESN, or the SGD algorithm in the LSTM training, we repeat the following experiment 100 times to derive statistically meaningful results:

1) Create training data over 750 seconds using random inputs from V or U, respectively, resulting in 15 000 data points ($\Delta T = 0.005$).

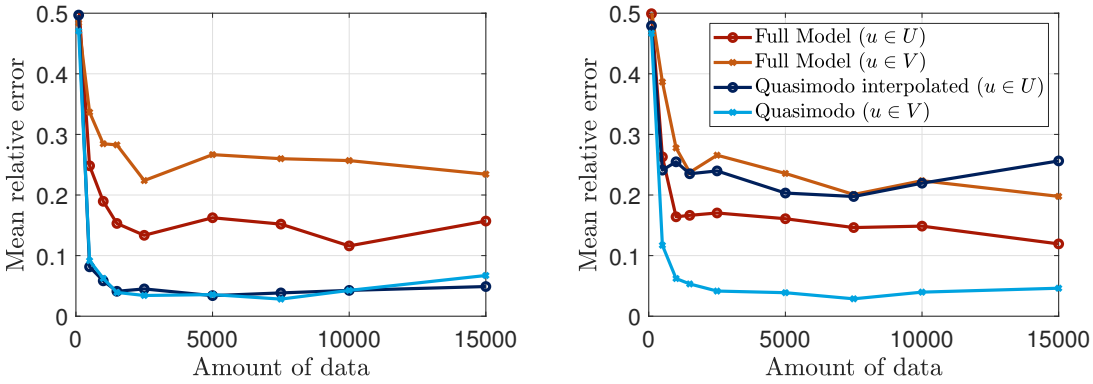
2) To study the dependency on the size of the training data set, use subsets of different sizes of the collected data (ranging from 100 to 15 000 data points).

3) Use these data sets to train the models (for ESN and LSTM) for the two approaches, i.e., train a model which gets the control as additional input and train two models which belong to the controls in $V = \{-50, 50\}$ or $V = \{0, \pi\}$.

4) Compare the relative L^2 prediction error over a horizon of 2 seconds, averaged over 100 simulations with random control sequences from either U or V. To predict the state by the autonomous models when control inputs from U are applied to the system, linear interpolation between the separate predictions is used.

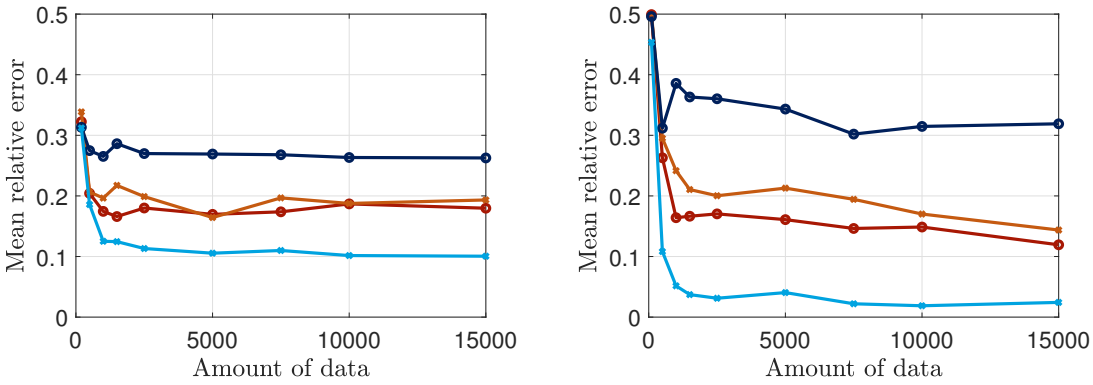
Note that the number of data points for the autonomous models together is the same as for the single model based on the augmented state, as for both variants, a single (long) time series of the same length is created in every experiment.

In Figure 4.9, the relative model error averaged over the time steps and the 100 random runs is plotted against the amount of data. In Figure 4.9a, the results for the control affine Lorenz system are presented, whereas in Figure 4.9b, the results with the adapted nonlinear control are shown. Naturally, for the affine control system, the performance of the QuaSiModO models is the same for control inputs in U and V and is of significantly higher quality than the single models getting the control as input. In particular, less data is needed to get a low model error, which is not surprising as we implicitly use the linearity of the underlying equations. In the nonlinear case, the results of the interpolated control are worse, as the interpolation error is added to the model error. However, the performance is still comparable to the full model. Furthermore, the SUR procedure yields controls from V anyways, such that this is the relevant control set for the comparison in the context of the QuaSiModO framework. For this case, the prediction accuracy of the autonomous models is, again, significantly better. For the same experiment with LSTM models,



(a) ESN for (4.14) with $U = [-50, 50]$, $V = \{-50, 50\}$.

(b) ESN for (4.15) with $U = [0, \pi]$,$V = \{0, \pi\}$.



(c) LSTM for (4.15) with $U = [0, \pi]$,$V = \{0, \pi\}$.

(d) ESN for (4.15) with $U = [0, \pi]$,$V = \{0, 0.8\pi\}$.

Figure 4.9: Relative L^2 prediction error (least-squares error) averaged over 100 experiments with random control inputs from U or V versus the amount of data used in the training of the various ESN and LSTM models. The coloring is the same in all diagrams, according to the legend in (b).

we observe a quite similar behavior, cf. Figure 4.9c. However, in contrast to the ESN case, there is quite a large gap between the interpolated solution derived from the autonomous models and the predictions by the full model. In the so far considered experiments, the controls in V were chosen to be the bounds of the set U, which leads to $M_V = 0$ for the presented system, cf. Lemma 4.2.1. Unfortunately, in practical applications, it may not always be that easy to choose an appropriate set V. Thus, we repeat the same experiment for the nonlinear Lorenz system with $V = \{0, 0.8\pi\}$ and use again ESNs as surrogate models. The result is presented in Figure 4.9d. The prediction for control inputs from V is significantly better using the autonomous systems, probably caused by simpler system dynamics. At the same time, the tracking performance for the interpolated (or, more precisely, extrapolated) solution shown by the dark blue line decreased significantly, as expected. However, we would like to mention that many real systems can be modeled in a control affine manner or allow for an informed selection of the control values needed to ensure a small M_V in Lemma 4.2.1.

The presented numerical experiments based on the prediction error suggest that the proposed framework is also beneficial concerning the data requirements. However, note that for the LSTMs as well as for the ESNs, the prediction performance is highly influenced by the chosen hyper-parameters, e.g., the size of the reservoir or the learning rate. Hence, the results presented here do not imply that the full model always performs worse. Nevertheless, the results were obtained without any fine tuning on the model setting and hence may give a good idea for the data requirements and model performance.

A further aspect slightly touching the topic of data requirements may occur in the setting of online learning. Due to the use of autonomous systems, online learning might be more robust against overfitting in the presented framework, especially in case the control input stays in one part of the control space for a longer time. Nevertheless, this does not guarantee the absence of overfitting since it can also occur when the state stays in one region of the state space for a long time. Since this case can also occur for the full model, the hope is that online learning can be performed more robustly in the presented framework. This, however, remains an open question and may be addressed in future work.

5 | Treating ℓ_1-Regularized Problems via Multiobjective Continuation

In the previous section, the main focus laid on the data assimilation process, which was simplified by adapting the control problem, i.e., by replacing the control set with a discretization. This addresses only one of the issues identified in Section 3.3. Another point mentioned there was the over-parameterization of neural networks, which may lead to overfitting. This is not only an issue of neural networks but is also true for other machine learning models, e.g., *support vector machines* (SVMs). The problem of overfitting can be addressed by enforcing sparse solutions, i.e., by ensuring that as many trainable parameters as possible are equal to zero and only some parameters are unequal to zero. Besides overfitting, this may also make the resulting models easier to interpret. This aspect is exploited, for instance, in the SINDy approach (sparse identification of nonlinear dynamics) [BPK16], where, based on data, governing equations of a dynamical system are sought from a given dictionary. In order to get an interpretable model and avoid overfitting, only some dictionary functions should be chosen, i.e., a sparse solution is sought. Other applications where this approach is followed can be found, for instance, in signal processing, compressed sensing, or medical imaging [Bar+10; EFM10; EK12; Plu+10; VBL13]. A common way to ensure sparsity of the solution, especially in data-based methods, is to add a regularization term to the original objective, cf. Sections 2.2.1 and 2.2.2, i.e., instead of minimizing the original objective $f : \mathbb{R}^n \to \mathbb{R}$, the regularized (or penalized) problem

$$\min_{x \in \mathbb{R}^n} f(x) + \lambda \|x\| \quad \text{with} \quad \lambda \in \mathbb{R}^{\geq 0} \tag{5.1}$$

is considered, where $\|\cdot\|$ is typically the ℓ_1-norm or the squared ℓ_2-norm as a smooth alternative. In this chapter, we consider the case where the regularization term is given by the ℓ_1-norm. Obviously, the solution to this problem crucially depends on the choice of the parameter λ. If λ is too small, the solution may not be sparse, which can lead to overfitting. On the other hand, if λ is too large, the solution might be too sparse in the sense that it is not able to fulfill the desired task, for instance, to predict the data correctly. In other words, we are searching for a certain compromise between f and the ℓ_1-norm where accuracy and sparsity are adequately balanced, or if we look at it from the multiobjective optimization perspective, the

aim is to find a suitable solution to the MOP

$$\min_{x \in \mathbb{R}^n} \begin{pmatrix} f(x) \\ \|x\|_1 \end{pmatrix}. \tag{MOP-ℓ_1}$$

In this context, solving (5.1) is similar to the weighted sum approach, i.e., solving $\min_{x \in \mathbb{R}^n} \alpha_1 f(x) + \alpha_2 \|x\|_1$ with $\alpha_1, \alpha_2 \in [0,1]$, $\alpha_1 + \alpha_2 = 1$, cf. Section 2.3.2. As discussed in Section 2.3.2, it is not possible to compute all Pareto optimal solutions by the weighted sum approach if f is nonconvex, cf. Figure 2.12. Since the loss functions optimized for NN training are nonconvex (due to the form of the NNs), well-generalizing sparse solutions may not be computable via the penalty approach.

Furthermore, even if the relevant compromises are solutions of (5.1), estimating an appropriate weight λ for the penalty term is difficult a priori. A naive approach would be to simply test multiple values of λ. However, in general, this procedure is very inefficient. A more sophisticated approach would be to use continuation methods. For the case of two (or more) smooth objectives, these were already introduced in Section 2.3.2. Since a first-order approximation of the Pareto critical set is used to obtain a good initial estimate (predictor) that allows fast computation of new critical points (corrector) by local optimization, the computation of the Pareto critical set is usually very efficient.

Unfortunately, since the ℓ_1-norm is nonsmooth, these methods are not directly applicable to (MOP-ℓ_1). Nonetheless, for a special case of (5.1) that occurs in signal or image processing, methods have been developed which can be interpreted as continuation methods. Similar to the previous example of the SINDy approach, in these applications, it is common to consider (over-determined) linear regression models, resulting in a minimization problem of the form

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \lambda \|x\|_1, \tag{5.2}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^{\geq 0}$. This problem is referred to as *Lasso* [Tib11; Tib96] or *basis pursuit* [CD94]. The so-called *homotopy methods* aim at computing the entire *regularization path*, i.e., all solutions that can be derived by varying $\lambda \geq 0$ in Problem (5.2) [Bri+18; DT08; MCW05; OPT00]. Since the regularization path is piecewise linear in this special case, the main idea of these methods is to determine points where "kinks" occur. This is done by utilizing the necessary optimality conditions of (5.2), which are based on the subdifferential of the ℓ_1-norm. In principle, these methods can be understood as continuation methods where no corrector step is required because the predictor provides exact results due to the linearity of the path. Furthermore, generalizations of these homotopy methods to arbitrary convex loss functions or convex penalty terms were introduced in [PH07; Ros04; ZY04].

Nevertheless, there is no similar method that is applicable to (MOP-ℓ_1) if f is nonconvex. Hence, this chapter aims at developing a continuation method for the general case where the first objective f is twice continuously differentiable and the second objective is the ℓ_1-norm. Although due to the nonsmoothness of the ℓ_1-norm,

it cannot be assumed that the solution set is smooth, we show that it is piecewise smooth for this particular problem class. This allows us to apply classical continuation methods (cf. Section 2.3.2) to compute the Pareto critical set up to kinks. We show how these kinks can be detected and addressed within the continuation by exploiting the special structure of the optimality conditions of (MOP-ℓ_1).

Another approach to solve problem (MOP-ℓ_1) with methods of multiobjective optimization in the context of NN training is presented in [Rei+22]. There, a multiobjective descent method in combination with *pruning* (a strategy to avoid overfitting by eliminating some of the trainable parameters during the optimization) is used to train the NN. However, the nonsmoothness of the ℓ_1-norm was not taken into account. Nevertheless, the method scales to higher dimensions since existing training methods from the area of neural networks can be utilized. Beyond that, there are alternative approaches in the intersection of sparse and multiobjective optimization based on the ε-constraint method (cf. Section 2.3.2). For instance, in [AS12; AS14], the authors train (linear) SVMs using this method. In [BF09], an efficient algorithm to approximate the regularization path of the Lasso problem, i.e., all solutions that can be derived by varying $\lambda \geq 0$ in (5.2), is presented, and in [BF11], a generalization to arbitrary convex penalty terms is derived. Another approach is followed in [Li+12], where the solution of (MOP-ℓ_1) is computed via evolutionary algorithms. In comparison to evolutionary methods, which also approximate the entire Pareto set (cf. Section 2.3.2), the approximation by the presented continuation method is usually closer to the true Pareto (critical) set while requiring fewer function evaluations. Moreover, the proposed method allows for analyzing structural properties, such as kinks or connectivity of the Pareto set, which remain unnoticed in (standard) evolutionary methods.

After introducing the continuation method in Section 5.1, including the derivation of optimality conditions for the concrete problem (MOP-ℓ_1), first numerical examples are presented in Section 5.2. Afterwards, we consider two possible extensions of our work. First, in Section 5.3, we discuss how the method may be adapted to solve higher dimensional problems. Second, we address the question of which structural results hold for more general regularization terms in Section 5.4.

Apart from Section 5.4, the work presented in this chapter was already published in [BGP22], to which the author of this thesis was the main contributor. The results shown in Section 5.4 summarize the work published in [GBP22] to which the author made significant contributions.

5.1 The Continuation Method

Similar to the continuation method for smooth MOPs presented in Section 2.3.2, the goal of the newly developed continuation method for (MOP-ℓ_1) is to compute the Pareto critical set $\mathcal{P}_c$ or, more precisely, its single connected components. In this special case, a possible starting value is always $x_{\mathsf{start}} = 0$ since it is Pareto critical as a global minimum of the ℓ_1-norm. However, in the case of multiple connected components, we still need to compute appropriate starting values that are Pareto

critical.

Assuming a Pareto critical point x_{start} is given, the idea is to walk along $\mathcal{P}_c$ by performing a predictor step along the tangent space of $\mathcal{P}_c$ and then a corrector step which results in the next Pareto critical point close by. Due to the nonsmoothness of the ℓ_1-norm, the Pareto critical set of (MOP-ℓ_1) may contain kinks, cf. Example 2.3.16, such that we cannot exclusively rely on the tangent space in the predictor step. Thus, we need to check whether we have reached a kink, and if this is the case, we additionally require a mechanism that computes directions in which the Pareto critical set continues. The rough concept of the method is summarized in Algorithm 4. To specify the single steps of the algorithm, we first have a more detailed look at the KKT condition given in (2.13) for the particular case of Problem (MOP-ℓ_1) in Section 5.1.1. Afterwards, we discuss the various ingredients of our method (e.g., the predictor and the corrector step) in Sections 5.1.2 to 5.1.4 and sum up the results in Section 5.1.5 by formulating our algorithm in more detail.

Algorithm 4 The rough concept of the continuation method.

Input: $f : \mathbb{R}^n \to \mathbb{R} \in C^2$, $x_{\text{start}} \in \mathbb{R}^n$ satisfying (2.13)
Output: Discretization of the Pareto critical set $\mathcal{P}_{\text{PC}}$

$\mathcal{P}_{\text{PC}} = \emptyset$, $x^0 = x_{\text{start}}$
while End of $\mathcal{P}_c$ component has not been reached **do**
  $\mathcal{P}_{\text{PC}} = \mathcal{P}_{\text{PC}} \cup \{x^0\}$
  if $\mathcal{P}_c$ is locally smooth **then**
    Predictor step tangential to $\mathcal{P}_c$: $x^p \leftarrow \text{predictor}(x^0)$
    Corrector step with $x^c \in \mathcal{P}_c$: $x^c \leftarrow \text{corrector}(x^p)$
    $x^0 = x^c$
  else
    Find new smooth part of $\mathcal{P}_c$.
  end if
end while

5.1.1 Optimality Conditions for (MOP-ℓ_1)

In Section 2.3.1, the basic concepts of multiobjective optimization, including a short discussion on optimality conditions, were introduced. Here, most important are the necessary conditions for a point $x \in \mathbb{R}^n$ to be Pareto optimal in the case where the objectives are nonsmooth, i.e., the KKT condition given in Proposition 2.3.13. Points fulfilling this condition are called Pareto critical, and the set of all Pareto critical points is denoted by $\mathcal{P}_c$. These are the points we want to compute using the continuation method. To construct a method specifically tailored to (MOP-ℓ_1), we start by analyzing the KKT condition for this MOP. Since we assume f to be at least twice continuously differentiable, the gradient $\nabla f(x)$ exists for every $x \in \mathbb{R}^n$. The subdifferential of the ℓ_1-norm was already considered in Example 2.3.12. Since the subgradients only depend on which of the entries of the variable x are zero or not, we introduce the following terminology.

Notation 5.1.1. *For $x \in \mathbb{R}^n$, we call an index $j \in \{1, \dots, n\}$* active *if $x_j \neq 0$ and* inactive *otherwise. Furthermore, we introduce the* active set *in x as*

$$A(x) := \{j \in \{1, \dots, n\} \mid x_j \neq 0\} \tag{5.3}$$

and denote the number of active indices by $n^A(x) := |A(x)|$ and the number of inactive indices by $n^0(x) := n - n^A(x)$. With this notation, the subdifferential is now the convex hull of $M(x) := 2^{n^0(x)}$ vectors g^i that are the extreme points of the subdifferential:

$$\begin{aligned} \partial f_{\ell_1}(x) = \mathbf{Conv}\left(\{g^i \mid i \in \{1, \dots, M(x)\}\}\right) \quad & \textit{with} \\ g^i_j = \operatorname{sgn}(x_j)\ \forall j \in A(x) \quad \textit{and} \quad & g^i_j \in \{-1, 1\}\ \forall j \notin A(x), \end{aligned}$$

where f_{ℓ_1} denotes the ℓ_1-norm, i.e., $f_{\ell_1}(x) = \|x\|_1$ for $x \in R^n$. Instead of $n^A(x)$, $n^0(x)$ and $M(x)$ we usually write n^A, n^0 and M, respectively, if it is clear to which variable x we refer to.

Inserting this into the KKT condition (2.14), a point x is Pareto critical for (MOP-ℓ_1) if and only if there exist $\alpha_1, \alpha_2 \in [0, 1]$ with $\alpha_1 + \alpha_2 = 1$ and

$$\alpha_1 \nabla f(x) + \alpha_2(\beta_1 g^1 + \dots + \beta_M g^M) = 0, \tag{5.4}$$

where $\beta_i \in [0, 1]$ for $i \in \{1, \dots, M\}$ and $\sum_{i=1}^M \beta_i = 1$. In this case, we refer to α_1 and α_2 as *KKT multipliers*. The following theorem further simplifies this condition and is the main result of this section that we use for the construction of the continuation method.

Theorem 5.1.2. *Let $x \in \mathbb{R}^n$ with $x \neq 0$ and let $a \in A(x)$, cf. (5.3), be any active index. Then, the following statements are equivalent:*

1. *x is Pareto critical for (MOP-ℓ_1).*
2. *The following conditions are satisfied for all $j \in \{1, \dots, n\}$:*
   - *(a) If $j \in A(x)$:*
     - *i. $\operatorname{sgn}(\nabla f(x)_j) = -\operatorname{sgn}(x_j)$ if $\nabla f(x)_j \neq 0$ and*
     - *ii. $|\nabla f(x)_j| = |\nabla f(x)_a|$.*
   - *(b) If $j \notin A(x)$: $|\nabla f(x)_j| \leq |\nabla f(x)_a|$.*

Proof. First, we prove the implication 1 $\Rightarrow$ 2, i.e., we assume that $x \in \mathcal{P}_c$. Then, there exist $\alpha_1, \alpha_2 \in [0, 1]$ with $\alpha_1 + \alpha_2 = 1$ and $\beta_i \in [0, 1]$, $i \in \{1, \dots, M\}$, with $\sum_{i=1}^M \beta_i = 1$ such that (5.4) is satisfied. Now, let be $j \in \{1, \dots, n\}$. In case that $j \in A(x)$, for the related indices of the subgradients spanning the subdifferential of the ℓ_1-norm, it holds $g^i_j = \operatorname{sgn}(x_j)$ for all $i \in \{1, \dots, M\}$ and

$$\begin{aligned} 0 &= \alpha_1 \nabla f(x)_j + \alpha_2 \operatorname{sgn}(x_j) \\ &= \alpha_1 \nabla f(x)_j + (1 - \alpha_1) \operatorname{sgn}(x_j). \end{aligned} \tag{5.5}$$

Since $x \neq 0$, we know that such an active j exists (e.g., $a \in A(x)$), which implies that $\alpha_1 > 0$ has to hold. Hence, (5.5) is equivalent to

$$\nabla f(x)_j = -\underbrace{\frac{1-\alpha_1}{\alpha_1}}_{\geq 0} \operatorname{sgn}(x_j). \tag{5.6}$$

Therefore, either $\nabla f(x)_j = 0$ (and $\alpha_1 = 1$) or $\operatorname{sgn}(\nabla f(x)_j) = -\operatorname{sgn}(x_j)$, i.e., 2(a)i holds. Furthermore, from (5.6), it follows for every $j \in A(x)$

$$|\nabla f(x)_j| = \frac{1-\alpha_1}{\alpha_1}$$

which means, in particular, that

$$|\nabla f(x)_j| = |\nabla f(x)_a|\,,$$

proving that 2(a)ii holds. Now we assume that $j \notin A(x)$ and define

$$\beta_{j,-} := \sum_{i:g_j^i=-1} \beta_i, \quad \beta_{j,+} := \sum_{i:g_j^i=1} \beta_i.$$

From (5.4), we obtain

$$\begin{aligned} 0 \quad &= \alpha_1 \nabla f(x)_j + \alpha_2(\beta_{j,+} - \beta_{j,-}) \\ \Leftrightarrow \quad \nabla f(x)_j \quad &= (\beta_{j,-} - \beta_{j,+})\frac{1-\alpha_1}{\alpha_1} \\ \Rightarrow \quad |\nabla f(x)_j| &= \underbrace{|\beta_{j,-} - \beta_{j,+}|}_{\leq 1} \frac{1-\alpha_1}{\alpha_1} \leq \frac{1-\alpha_1}{\alpha_1} = |\nabla f(x)_a|\,, \end{aligned}$$

i.e., 2b is satisfied.

To prove the opposite direction 2 $\Rightarrow$ 1, we assume that x satisfies the Conditions 2a and 2b. Note that 2(a)i implies $\nabla f(x)_a = 0$ or $\operatorname{sgn}(x_a)\nabla f(x)_a < 0$. In the first case, due to 2(a)ii and 2b, it follows directly $\nabla f(x) = 0$, i.e., x is critical with $\alpha = (1,0)^\top$. In the latter case, it follows

$$1 - \operatorname{sgn}(x_a)\nabla f(x)_a > 1$$

and we can define

$$\alpha_1 := \frac{1}{1 - \operatorname{sgn}(x_a)\nabla f(x)_a} \in (0,1).$$

By 2(a)i and 2(a)ii we obtain

$$\alpha_1 = \frac{1}{1+|\nabla f(x)_a|} = \frac{1}{1+|\nabla f(x)_j|} \quad \forall j \in A(x),$$

which is equivalent to

$$\frac{\alpha_1}{1-\alpha_1}|\nabla f(x)_j| = \frac{1}{\frac{1}{\alpha_1}-1}|\nabla f(x)_j| = 1 \quad \forall j \in A(x).$$

In particular, with 2(a)i it follows

$$\frac{\alpha_1}{1-\alpha_1}\nabla f(x)_j = \operatorname{sgn}(\nabla f(x)_j) = -\operatorname{sgn}(x_j).$$

Analogously, for all inactive j, we can use 2b to obtain

$$\begin{aligned} & \frac{\alpha_1}{1-\alpha_1}\left|\nabla f(x)_j\right| \leq \frac{\alpha_1}{1-\alpha_1}\left|\nabla f(x)_a\right| = 1 \\ \Rightarrow \quad & \frac{\alpha_1}{1-\alpha_1}\nabla f(x)_j \in [-1,1]. \end{aligned}$$

As a result, the vector $-\frac{\alpha_1}{1-\alpha_1}\nabla f(x)$ can be written as a convex combination of the subgradients g^i. Let $\beta_1, ..., \beta_M$ be the corresponding coefficients and define $\alpha_2 = 1 - \alpha_1$. Then

$$\begin{aligned} & -\frac{\alpha_1}{1-\alpha_1}\nabla f(x) = \beta_1 g^1 + \cdots + \beta_M g^M \\ \Leftrightarrow \quad & \alpha_1 \nabla f(x) + \alpha_2(\beta_1 g^1 + \cdots + \beta_M g^M) = 0, \end{aligned}$$

showing that (5.4) holds, i.e., $x \in \mathcal{P}_c$. □

Theorem 5.1.2 states that a point $x \in \mathbb{R}^n$ is Pareto critical if and only if the absolute values of all gradient entries belonging to active indices are identical and, at the same time, the absolute values of gradient entries belonging to inactive indices are less than or equal to the absolute value of the gradient entries corresponding to active indices. Furthermore, the proof of the theorem yields an explicit formula for the KKT multipliers of Pareto critical points. This allows us to infer some properties of the Pareto front later (cf. Remark 5.1.5).

Remark 5.1.3. *If $x \neq 0$ is Pareto critical for (MOP-ℓ_1) and a is an active index, then the KKT multipliers corresponding to x are given by*

$$\alpha_1 = \frac{1}{1+|\nabla f(x)_a|}, \quad \alpha_2 = \frac{|\nabla f(x)_a|}{1+|\nabla f(x)_a|}. \tag{5.7}$$

In particular, the KKT multipliers are unique (except for $x = 0$).

Since (MOP-ℓ_1) is a nonsmooth problem, the Pareto critical set generally contains points in which the tangent space does not exist, i.e., kinks in the Pareto critical set can be observed, cf. Example 2.3.16. Based on Theorem 5.1.2, one can show that such kinks can only occur if the activation structure changes, i.e., when an index switches from inactive to active (or the other way around), which can only happen if 2b in Theorem 5.1.2 is satisfied with equality. This is shown more precisely in the following corollary.

Corollary 5.1.4. *Let $x^0 \neq 0$ be a Pareto critical point of (MOP-ℓ_1) and assume that the inequality in 2b in Theorem 5.1.2 is strict. Furthermore, let the active*

indices at x^0 *be given by* $A(x^0) = \{j_1 < ... < j_{n^A(x^0)}\}$. *We define the projection of* $x \in \mathbb{R}^n$ *onto the entries corresponding to the active indices of* x^0 *and its inverse by*

$$p^A : \mathbb{R}^n \to \mathbb{R}^{n^A(x^0)}, \quad p^A(x)_i = x_{j_i} \quad \forall i \in \{1, ..., n^A(x^0)\},$$
$$l^A : \mathbb{R}^{n^A(x^0)} \to \mathbb{R}^n, \quad l^A(\bar{x})_j = \begin{cases} 0, & \text{if } j \notin A(x^0), \\ \bar{x}_i, & \text{if } j = j_i \in A(x^0), \end{cases} \quad \forall j \in \{1, ..., n\}.$$

Then there is an open set $U \subseteq \mathbb{R}^n$ *with* $x^0 \in U$ *such that* $x^* \in U$ *is Pareto critical for* (MOP-ℓ_1) *if and only if* $A(x^*) = A(x^0)$ *and* $p^A(x^*)$ *is Pareto critical for the (smooth) MOP*

$$\min_{\bar{x} \in \mathbb{R}^{n^A(x^0)}} \begin{pmatrix} f(l^A(\bar{x})) \\ \|\bar{x}\|_1 \end{pmatrix}. \tag{5.8}$$

Proof. Note that if $A(x^0) = \{1, \dots, n\}$, i.e., $x^0_j \neq 0$ for all $j \in \{1, \dots, n\}$, the ℓ_1-norm is smooth locally around x^0 and the corollary trivially holds. So, we assume from now on that $A(x^0) \subsetneq \{1, \dots, n\}$. Moreover, let $a \in A(x^0)$ be an arbitrary active index. This implies that $\nabla f(x^0)_a \neq 0$ since otherwise the inequality in 2b in Theorem 5.1.2 cannot be strict.

First, note that there exists an open neighbourhood $U_1 \subseteq \mathbb{R}^n$ of x^0 such that for all $x \in U_1$ it holds

$$\operatorname{sgn}(x_j) = \operatorname{sgn}(x^0_j) \neq 0 \quad \forall j \in A(x^0).$$

This implies, in particular,

$$A(x^0) \subseteq A(x) \quad \forall x \in U_1. \tag{5.9}$$

Since f is at least continuously differentiable, its gradient is continuous, and as the inequality in 2b in Theorem 5.1.2 is strict, there has to be an open set $U_2 \subseteq \mathbb{R}^n$ with $x^0 \in U_2$ such that for all $x \in U_2$

$$|\nabla f(x)_j| < |\nabla f(x)_a| \ \forall j \notin A(x^0). \tag{5.10}$$

According to Theorem 5.1.2, this implies that every index j that is inactive in x^0 is also inactive in $x \in U_2$ if x is Pareto critical for (MOP-ℓ_1), i.e.,

$$A(x) \subseteq A(x^0) \quad \forall x \in U_2 \cap \mathcal{P}_c. \tag{5.11}$$

Now, we define $U := U_1 \cap U_2$ and assume that $x \in U$. If $x \in \mathcal{P}_c$, then, by (5.9) and (5.11), x has the same activation structure as x^0, i.e.,

$$A(x) = A(x^0).$$

In addition, according to Theorem 5.1.2, it holds

$$\begin{aligned} \operatorname{sgn}(\nabla f(x)_j) &= -\operatorname{sgn}(x_j) \quad \forall j \in A(x^0) \quad \text{and} \\ |\nabla f(x)_j| &= |\nabla f(x)_a| \quad \forall j \in A(x^0). \end{aligned} \tag{5.12}$$

Recall that $\nabla f(x)_j \neq 0$ for all $j \in A(x^0)$ since $A(x^0) \subsetneq \{1, \ldots, n\}$.

By applying Theorem 5.1.2 to the MOP (5.8), which has the same form as (MOP-ℓ_1), we see that these are precisely the conditions for $p^A(x)$ to be Pareto critical. For the other direction, we assume that $A(x) = A(x^0)$ and that $p^A(x)$ is Pareto critical for (5.8), i.e., (5.12) holds. Since $x \in U \subseteq U_2$, (5.10) holds as well. Thus, according to Theorem 5.1.2, x is Pareto critical for (MOP-ℓ_1). □

The previous corollary implies that if the inequality in 2b in Theorem 5.1.2 is strict for a point $x^0 \in \mathcal{P}_c$, then, locally around x^0, the Pareto critical set of (MOP-ℓ_1) is identical to the Pareto critical set of the MOP (5.8) lifted into the $\mathbb{R}^n$ via the map l^A. Note that the KKT multipliers are identical for both problems (since they are given by (5.7) in both cases). Because the objective functions of (5.8) are smooth around $p^A(x^0)$ (as all entries of $p^A(x^0)$ are nonzero by construction), we can apply the theory for smooth MOPs. Thus, we can conclude that the tangent space of the Pareto critical set $\mathcal{P}_c$ must exist in x^0, cf. Theorem 2.3.17. Hence, $\mathcal{P}_c$ can only possess a kink in x^0 if 2b in Theorem 5.1.2 is satisfied with equality for at least one $j \notin A(x^0)$. These are exactly the points where the active set can potentially change, i.e., where an inactive index could be activated (or vice versa).

In particular, this allows us to construct the predictor (Section 5.1.2) and the corrector (Section 5.1.3) as in the smooth case based on (5.8), except for potential kinks. We already know that these kinks can be detected by observing the inequality in 2b in Theorem 5.1.2. In order to overcome such points, we basically just need to discuss how we can continue from there, i.e., how we can choose a suitable direction for the subsequent continuation step, cf. Section 5.1.4. Finally, we summarize the results by presenting the overall algorithm in Section 5.1.5.

Furthermore, based on the uniqueness of the KKT multipliers, cf. Remark 5.1.3, we can make the following remark concerning kinks in the Pareto front or, more precisely, in the image of a connected component of the Pareto critical set.

Remark 5.1.5. *In [Hil01], it was proven that the KKT multipliers are orthogonal to the tangent space of the image of the Pareto critical set if the objectives are smooth. If we consider the image of two smooth parts of the Pareto critical set of* (MOP-ℓ_1) *that touch in a point where the activation structure changes, these two paths (in the image space) have the same derivative in the connecting point according to* (5.7)*, i.e., there is no kink in the image. However, kinks in the image of a connected component of the Pareto critical set can occur in the form of so-called turning points, which are discussed later, see Example 5.1.6. Furthermore, there may be multiple disconnected parts of the Pareto front that belong to different connected components of the Pareto (critical) set. Note that disconnected components of the Pareto (critical) set do not necessarily lead to a disconnected Pareto front.*

5.1.2 Predictor

Based on the results of the previous section, we can now introduce the predictor step. To this end, we assume that a point $x^0 \in \mathbb{R}^n$ is given which is Pareto critical

for (MOP-ℓ_1) and for which 2b in Theorem 5.1.2 is satisfied strictly, i.e., according to Corollary 5.1.4, an open neighborhood around x^0 exists in which the activation structure (in the Pareto critical set) does not change. As discussed before, this allows us to perform the predictor (and later the corrector) step for the reduced MOP (5.8). Since we assume that f is at least twice continuously differentiable, both objectives in (5.8) are twice continuously differentiable, and we can compute the predictor at the point $p^A(x^0)$ as explained in Section 2.3.2 or [Hil01]. Therefore, similar to (2.22), we define the map

$$H : \mathbb{R}^{n^A} \times [0,1]^2 \to \mathbb{R}^{n^A+1}, \quad H(\bar{x}, \alpha) = \begin{pmatrix} \alpha_1 \nabla (f \circ l^A)(\bar{x}) + \alpha_2 \nabla f_{\ell_1}(\bar{x}) \\ \alpha_1 + \alpha_2 - 1 \end{pmatrix}, \tag{5.13}$$

where we consider the active set with respect to x^0, i.e., $n^A = n^A(x^0)$ and l^A and p^A as in Corollary 5.1.4. The zero set of H consists of the points $(\bar{x}, \alpha) \in \mathbb{R}^{n^A} \times [0,1]^2$ that fulfill the KKT condition for (5.8), i.e.,

$$\begin{aligned} & H(\bar{x}, \alpha) = 0 \\ \Leftrightarrow \quad & \bar{x} \in \mathbb{R}^{n^A} \text{ is Pareto critical for (5.8) with KKT multipliers } \alpha_1 \text{ and } \alpha_2 \\ \Leftrightarrow \quad & l^A(\bar{x}) \in \mathbb{R}^n \text{ is Pareto critical for (MOP-}\ell_1\text{) with KKT multipliers } \alpha_1 \text{ and } \alpha_2. \end{aligned}$$

The Jacobian of H in $(p^A(x^0), \alpha) \in \mathbb{R}^{n^A} \times [0,1]^2$ is given by

$$\begin{aligned} & DH(p^A(x^0), \alpha) \\ &= \begin{pmatrix} \alpha_1 \nabla^2 (f \circ l^A)(p^A(x^0)) & \nabla (f \circ l^A)(p^A(x^0)) & \nabla f_{\ell_1}(p^A(x^0)) \\ 0 & 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \alpha_1 \nabla^2 f(x^0)_{|A(x^0)} & \nabla f(x^0)_{|A(x^0)} & s \\ 0 & 1 & 1 \end{pmatrix} \in \mathbb{R}^{(n^A+1)\times(n^A+2)}, \end{aligned}$$

where $(\nabla^2 f(x))_{|A(x^0)} \in \mathbb{R}^{n^A \times n^A}$ and $\nabla f(x)_{|A(x^0)} \in \mathbb{R}^{n^A}$ denote the Hessian and the gradient of f reduced to the active indices, respectively, and $s = \operatorname{sgn}(p^A(x^0))$ is the result of the sign function applied componentwise to the nonzero entries of x^0. According to Theorem 2.3.17, if $DH(p^A(x^0), \alpha)$ has rank $n^A + 1$, the zero set of H is a one-dimensional submanifold locally around x^0. Hence, to ensure that this property holds, we assume that the reduced Hessian matrix

$$(\nabla^2 f(x^0))_{|A(x^0)} \text{ is regular.}$$

This is the generic case, as the set of singular matrices is a null set, and we compute a pointwise approximation of the Pareto critical set. Note that if $(\nabla^2 f(x^0))_{|A(x^0)}$ would be singular, bifurcations may occur, as discussed for general continuation methods in [AG90] and shown for a smooth MOP in [Geb22, Example 2.2.13].

Since we assume that the reduced Hessian $(\nabla^2 f(x^0))_{|A(x^0)}$ is regular, the one-dimensional tangent space of the respective manifold is given by

$$\ker(DH(p^A(x^0), \alpha)) = \ker \begin{pmatrix} \alpha_1 \nabla^2 f(x^0)_{|A(x^0)} & \nabla f(x^0)_{|A(x^0)} & s \\ 0 & 1 & 1 \end{pmatrix} \tag{5.14}$$

and we obtain a unique vector (up to scaling and sign) as result. In addition, we can also give an explicit formula for the kernel. To this end, let $a \in A(x^0)$ be an arbitrary active index. Then, Theorem 5.1.2 implies $\nabla f(x^0)_{|A(x^0)} = -|(\nabla f(x^0))_a|\, s$, and hence, a kernel vector $\bar{v}$ of (5.14) is given by

$$\begin{aligned} \bar{v} &= \gamma \cdot (\bar{v}_1, \bar{v}_2, \bar{v}_3)^\top \quad \text{with} \quad \gamma \in \mathbb{R} \quad \text{and} \\ \bar{v}_1 &= \frac{(1 + |\nabla f(x^0)_a|)}{\alpha_1} \left(\nabla^2 f(x^0)_{|A(x^0)}\right)^{-1} s, \\ &\overset{(5.7)}{=} (1 + |\nabla f(x^0)_a|)^2 \left(\nabla^2 f(x^0)_{|A(x^0)}\right)^{-1} s, \\ \bar{v}_2 &= 1, \\ \bar{v}_3 &= -1. \end{aligned} \tag{P}$$

If we would perform a predictor step for (5.8), i.e., a step in the reduced space of active indices, it would be given by $\bar{v}_1$, while $\bar{v}_2$ and $\bar{v}_3$ represent the corresponding direction in the space of KKT multipliers. The predictor direction for our original problem (MOP-ℓ_1) is then given by $v_1 = l^A(\bar{v}_1)$, i.e., the result of the predictor is given by

$$x^p = x^0 + \underbrace{h \cdot (\pm v_1)}_{=:v^p}, \tag{5.15}$$

where it remains to choose the sign and the scaling $h \in \mathbb{R}^{>0}$ of v_1 to obtain the final predictor step v_p.

Sign of the direction

After computing the vector v_1, there still remain two possible choices for the direction of the predictor step, namely v_1 and $-v_1$. The most intuitive approach would be to use the directional derivative of f as an indicator. That is if we want to move along the Pareto front in the direction in which the value of f decreases (and the value of the ℓ_1-norm increases), we choose $v^p = \pm v_1$ such that $\nabla f(x^0)^\top v^p < 0$, and $>$ otherwise, as suggested in [Hil01] and [MS17].

As long as $\nabla f(x^0)^\top v_1 \neq 0$ holds, this approach is usually suitable. However, it may happen that during the continuation, so-called *turning points* are passed. These are points where we follow the Pareto critical set in the same direction, but the corresponding direction in the objective space vanishes and then turns around, i.e., by following the Pareto critical set, the sign of $\nabla f(x)^\top v_1$ is first negative, then becomes zero, and afterwards positive. This phenomenon is not related to the nonsmoothness of the ℓ_1-norm but can also occur in the smooth parts, i.e., on a single activation structure, as shown in the following example.

Example 5.1.6. *Consider* (MOP-ℓ_1) *with*

$$f(x) = (x_1 + 1)^2 + (x_2 - 1)^4 - \tfrac{1}{2}\left(x_2 - \tfrac{1}{4}\right)^3. \tag{5.16}$$

Figure 5.1 shows the Pareto critical set for this problem (which can be computed analytically). We can observe two turning points, labeled x^1 and x^2, where the direction in the objective space turns around.

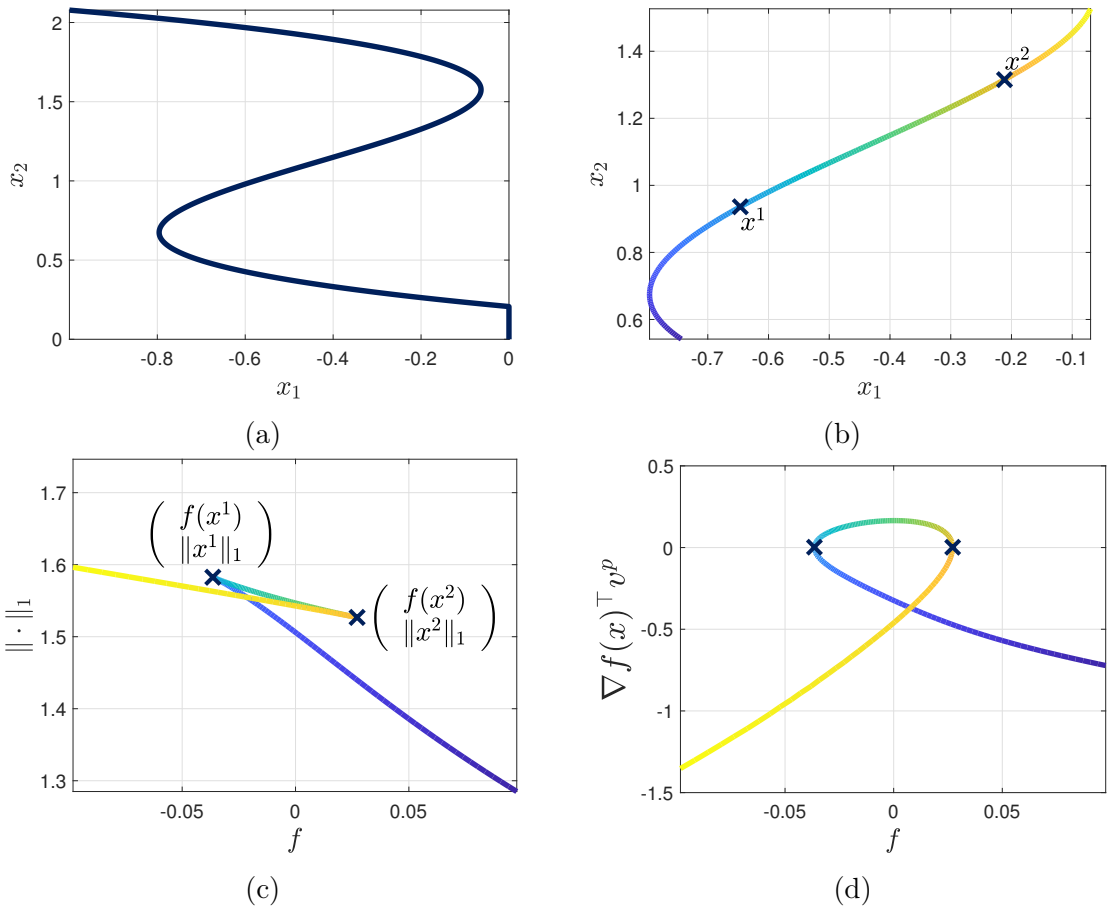


Figure 5.1: (a) $\mathcal{P}_c$ for the MOP in Example 5.1.6. (b) Part of $\mathcal{P}_c$ where the turning points x^1 and x^2 occur. (c) Image of $\mathcal{P}_c$ where the image of a point has the same color as the point in the variable space in (b). (d) Derivative of f in the direction of the predictor, i.e., $\nabla f(x)^\top v_1$, where again the corresponding coloring is used.

Since these turning points also occur if we consider the neuronal network training, we use a different criterion that stems from classical continuation for dynamical systems and is proposed in [AG90]. There, the authors suggest to observe the sign of the determinant of the *augmented Jacobian* $J_a = \left(DH(p^A(x^0), \alpha)^\top, \bar{v}\right)^\top$, which determines the *orientation of the curve*. In our case, the augmented Jacobian is given by

$$J_a := \begin{pmatrix} \alpha_1 \nabla^2 f(x^0)_{|A(x^0)} & \nabla f(x^0)_{|A(x^0)} & s \\ 0 & 1 & 1 \\ \bar{v}_1^\top & \bar{v}_2 & \bar{v}_3 \end{pmatrix}. \tag{5.17}$$

Furthermore, a simple (but rather technical) calculation based on the Laplace expansion shows that

$$\det(J_a) = -\bar{v}_2 \cdot z \cdot \det\left(\alpha_1 \nabla^2 f(x^0)_{|A(x^0)}\right), \tag{5.18}$$

where $z = 2+(1+|\nabla f(x^0)_a|)^4 \left\|(\nabla^2 f(x^0)_{|A(x^0)})^{-1} s\right\|_2^2 > 0$ with $a \in A(x^0)$. As a result, only the determinant of $\nabla^2 f(x^0)_{|A(x^0)}$ has to be computed to determine the sign of

$\det(J_a)$. If the sign of $\det(J_a)$ is constant in one activation structure, then this ensures that $\bar{v}$ has the same orientation with respect to the image of $DH(p^A(x^0), \alpha)$. This way, it prevents us from walking back in the direction we came from in the variable space. As discussed in Section 5.1.4, the correct direction is usually known in points where we change the activation structure. Thus, keeping the sign of $\det(J_a)$ constant over parts with constant activation structure is a suitable criterion to choose the direction in each predictor step.

Step size

Now that we know how to compute the direction v_1 and determine its sign, we still need to choose a step size $h \in \mathbb{R}^{>0}$ in (5.15). To this end, different strategies can be followed. Here, we aim for a uniform coverage of the Pareto critical set. Therefore, we choose a constant step size in the variable space, i.e., for a predefined constant $\tau \in \mathbb{R}^{>0}$, we choose

$$h := \frac{\tau}{\|v_1\|_2}. \tag{5.19}$$

Another possibility would be to derive an even distribution in the objective space. In this case, one may choose

$$h := \frac{\tau}{\|Jv_1\|_2}$$

with $J = (\nabla f(x^0), \operatorname{sgn}(x^0))^\top$ as step size, as suggested in [Hil01; MS17]. Furthermore, it could be useful to consider the curvature of the Pareto critical set. In [AG90], different criteria are suggested to take the curvature into account, for instance, the length of the corrector step or the angle between the predictor directions of subsequent iterations.

In addition, it is necessary to ensure that the activation structure remains constant to be able to perform the corrector step in a smooth region of the problem. To this end, in case the predictor step crosses zero in an active index, the step size h has to be shortened, i.e., we choose

$$\hat{h} = \min(h, \bar{h}) \tag{5.20}$$

as step size, where

$$\bar{h} = \sup(\{h \in \mathbb{R}^{>0} : A(x^p) = A(x^0) \text{ for } x^p = x^0 + hv_1\}).$$

5.1.3 Corrector

Similar to the usual continuation methods for smooth MOPs, the aim of the corrector step is to map the endpoint of the predictor x^p to the closest Pareto critical point, cf. Section 2.3.2. Here, we additionally need to ensure that the zero structure does not change, as it otherwise would be difficult to formulate the correct KKT condition, i.e., to choose suitable subgradients. Hence, we are searching for a point $x^c \in \mathcal{P}_c$

which is close to x^p and for which $A(x^c) = A(x^p) = A(x^0)$ holds. Theoretically, to derive such a point, one could solve the equations given by the KKT condition, cf. Equation (5.4), using x^p as a starting point, whereby constraints have to be added to ensure the constant zero structure. Depending on the number of nonzero elements, this can lead to very large systems as the subdifferential of the ℓ_1-norm is spanned by $M(x^0) = 2^{n^0(x^0)}$ subgradients. Fortunately, Theorem 5.1.2 provides an alternative that allows us to avoid the explicit use of subgradients. Therefore, in the corrector step, we solve the following optimization problem:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \|x - x^p\|_2^2, \\ \text{s.t.} \quad &(\nabla f(x)_j)^2 - (\nabla f(x)_a)^2 = 0 \text{ if } x_j^0 \neq 0, \\ &(\nabla f(x)_j)^2 - (\nabla f(x)_a)^2 \leq 0 \text{ if } x_j^0 = 0, \\ &\nabla f(x)_j \leq 0 \text{ if } x_j^0 > 0, \\ &\nabla f(x)_j \geq 0 \text{ if } x_j^0 < 0, \\ &x_j \geq 0 \text{ if } x_j^0 > 0, \\ &x_j \leq 0 \text{ if } x_j^0 < 0, \\ &x_j = 0 \text{ if } x_j^0 = 0, \end{aligned} \tag{C}$$

where $a \in A(x^0)$ is some index that is currently active, i.e., $x_a^0 \neq 0$ (and hence, also $x_a^p \neq 0$).

5.1.4 Changing the Activation Structure

With the predictor and the corrector step, we are now able to compute the smooth parts of the Pareto critical set, i.e., the parts with constant activation structure. Hence, if we were able to overcome kinks caused by indices becoming inactive or active, we could compute the entire Pareto critical set or, to be more precise, a connected component of it. Thus, this section aims at deriving mechanisms that allow for changing the activation structure $A(x)$, i.e., *activating* and *deactivating* indices, during the continuation and proceed the continuation in the new activation structure.

The most obvious case for deactivating an index is when the predictor crosses zero in one of the entries. As already discussed in Section 5.1.2, in this case we reduce the step length h to $\bar{h}$ as the largest step size such that

$$x_j^p \leq 0 \text{ if } x_j^0 < 0 \quad \text{and} \quad x_j^p \geq 0 \text{ if } x_j^0 > 0$$

for all $j \in A(x^0)$, and the indices in $A(x^0) \setminus A(x^p)$ are considered as inactive in the subsequent corrector step. If the corresponding Problem (C) has a solution, then the continuation method can continue on the new activation structure. Otherwise, we choose a step length in $(0, \bar{h})$ to find a new Pareto critical point with respect to the old active set $A(x^0)$. The likelihood of an active index being falsely deactivated (meaning that it is deactivated too early and we miss relevant parts of the Pareto critical set) by this mechanism depends on the constant step size τ and the curvature of the Pareto critical set.

The second situation, where the active set may change, is the case where the corrector terminates at a point where equality holds in 2b in Theorem 5.1.2 for some index $j \in \{1, \dots, m\}$, i.e., $x_j = 0$ and $|\nabla f(x)_j| = |\nabla f(x)_a|$ with $a \in A(x)$. We call such an index j *potentially active* and denote the set of all potentially active indices by

$$A_p(x) := \{j \in \{1, \dots, n\} \mid x_j = 0 \text{ and } |\nabla f(x)_j| = |\nabla f(x)_a|\}. \tag{5.21}$$

Potentially active indices can occur if the corrector computes a point x^c where one of the previously active indices is now zero, i.e., $x_j^c = 0$ for some $j \in A(x^0)$. Note that this situation is relatively unlikely compared to the predictor crossing the zero in an active index. Much more likely is that the corrector ends in a point x^c where an index j is potentially active that was inactive before, i.e., $x_j^0 = 0$, and has to be activated in x^c. Nevertheless, mathematically, both situations are equivalent, so we do not distinguish between them.

Thus, let us assume that we are in a Pareto critical point x^* with $A_p(x^*) \neq \emptyset$. If x^* is an endpoint of the current activation structure, then the construction of the corrector ensures that we reach it after a finite number of steps (if the step length τ is sufficiently small). If x^* is not an endpoint, then we are, in general, unable to detect it during continuation since we move with a constant step size. Fortunately, the latter points are less likely to exist, as discussed briefly in Remark 5.1.14.

Note that not all potentially active indices necessarily need to be activated or deactivated, so a way to identify the relevant indices is required. Denoting the number of potentially active indices with p, we have 2^p possible activation structures. For every possible activation structure $A_p^l \subseteq A_p(x^*)$, $l \in \{1, \dots, 2^p\}$, we can compute a vector $\bar{v}^l = (\bar{v}_1^l, \bar{v}_2^l, \bar{v}_3^l)$ as in the predictor step (Section 5.1.2). If we assume $|\nabla f(x^*)_a| > 0$, then according to 2a in Theorem 5.1.2, the correct sign of the possibly active indices is given by $-\operatorname{sgn}(\nabla f(x^*))_j$. Therefore, we have to consider the directions $\bar{v}^l$ given by

$$\ker \begin{pmatrix} \alpha_1 \nabla^2 f(x^*)_{|A(x^*) \cup A_p^l} & \nabla f(x^*)_{|A(x^*) \cup A_p^l} & s^l \\ 0 & 1 & 1 \end{pmatrix}, \tag{5.22}$$

with $s^l = -\operatorname{sgn}(\nabla f(x^*)_{|A(x^*) \cup A_p^l})$. If the Hessian $\nabla^2 f(x^*)_{|A(x^*) \cup A_p^l}$ is regular, we obtain, similar to (P),

$$\begin{aligned} \bar{v}_1^l &= (1 + |\nabla f(x^*)_a|)^2 \left(\nabla^2 f(x^*)_{|A(x^*) \cup A_p^l}\right)^{-1} s^l, \\ \bar{v}_2^l &= 1, \\ \bar{v}_3^l &= -1. \end{aligned} \tag{5.23}$$

Analogously to Section 5.1.2, we obtain the direction $v_1^l \in \mathbb{R}^n$ by the corresponding embedding $l^{A(x^*) \cup A_p^l} : \mathbb{R}^{n^{A(x^*)} + |A_p^l|} \to \mathbb{R}^n$. If A_p^l is a set of potentially active indices that can indeed be activated, i.e., if there is a curve γ of Pareto critical points starting in x^* with the active set $A(x^*) \cup A_p^l$, then by construction, v_1^l is the tangent vector of that curve. By analyzing the relationship between the derivative of γ at x^* and the conditions in Theorem 5.1.2, we obtain necessary conditions for A_p^l to be activated. This is done in the following lemma and the subsequent corollary.

Lemma 5.1.7. *Let $x^* \in \mathcal{P}_c \setminus \{0\}$ with $\nabla f(x^*) \neq 0$. Let $\gamma : (-1, 1) \to \mathbb{R}^n$ be a map such that*

(i) γ is continuously differentiable,

(ii) $\gamma(t) \in \mathcal{P}_c \quad \forall t \in [0, 1)$,

(iii) $\gamma(0) = x^*$ and

(iv) the active set $A(\gamma(t)) =: A_\gamma$ is constant $\forall t \in (0, 1)$.

Let $a \in A(x^*)$ and $w = \dot{\gamma}(0)$. Then, for all $j \in A_\gamma \setminus A(x^*)$, it holds

(a) either $w_j = 0$ or $\operatorname{sgn}(w_j) = -\operatorname{sgn}(\nabla f(x^*)_j)$.

Furthermore, if $A_p(x^*)$ is given by (5.21), we have for all $j \in A_p(x^*) \setminus A_\gamma$

(b) $\operatorname{sgn}(\nabla f(x^*)_j)(\nabla^2 f(x^*)w)_j \leq \operatorname{sgn}(\nabla f(x^*)_a)(\nabla^2 f(x^*)w)_a.$

Proof. First, since $\nabla f(x^*) \neq 0$, according to Theorem 5.1.2 and due to the continuity of ∇f and γ, there exists $\varepsilon > 0$ such that

$$\nabla f(\gamma(t))_j \neq 0 \quad \forall t \in [0, \varepsilon) \text{ and } j \in A(x^*) \cup A_p(x^*).$$

This implies that

$$\operatorname{sgn}(\nabla f(\gamma(t))_j) \text{ is constant} \quad \forall t \in [0, \varepsilon) \text{ and } j \in A(x^*) \cup A_p(x^*). \tag{5.24}$$

Furthermore, note that $A_\gamma \subseteq A(x^*) \cup A_p(x^*)$, i.e., (5.24) is especially true for $j \in A_\gamma$. We start by proving (a). To this end, assume that $j \in A_\gamma \setminus A(x^*)$. Due to (ii) and (iv), according to 2(a)i in Theorem 5.1.2, we have for all $t \in (0, 1)$

$$\operatorname{sgn}(\gamma_j(t)) = -\operatorname{sgn}(\nabla f(\gamma(t))_j), \tag{5.25}$$

if $\nabla f(\gamma(t))_j \neq 0$. By (5.24), this implies that $\operatorname{sgn}(\gamma_j(t))$ is constant for all $t \in (0, \varepsilon)$. Due to (iii), we have $\gamma_j(0) = x_j^* = 0$ and, thus,

$$w_j = \dot{\gamma}_j(0) = \lim_{t \to 0} \frac{\gamma_j(t) - \gamma_j(0)}{t} = \lim_{t \to 0} \frac{\gamma_j(t)}{t} = \lim_{t \downarrow 0} \frac{\gamma_j(t)}{t}.$$

If $w_j \neq 0$, since $\operatorname{sgn}(\gamma_j(t))$ is constant for all $t \in (0, \varepsilon)$, we have

$$\operatorname{sgn}(w_j) = \operatorname{sgn}(\gamma_j(t)) \quad \text{for } t \in (0, \varepsilon)$$

and, thus,

$$\operatorname{sgn}(w_j) \overset{(5.25)}{=} -\operatorname{sgn}(\nabla f(\gamma(t))_j) \overset{(5.24)}{=} -\operatorname{sgn}(\nabla f(x^*)_j) \quad \text{for } t \in (0, \varepsilon).$$

Now, we prove (b). Let $j \in A_p(x^*) \setminus A_\gamma$. From (ii) and 2b in Theorem 5.1.2, it follows that

$$\begin{aligned} &\operatorname{sgn}(\nabla f(\gamma(t))_j)\nabla f(\gamma(t))_j = |\nabla f(\gamma(t))_j| \\ &\leq |\nabla f(\gamma(t))_a| = \operatorname{sgn}(\nabla f(\gamma(t))_a)\nabla f(\gamma(t))_a \end{aligned} \tag{5.26}$$

for all $t \in [0,1)$. As $j \in A_p(x^*)$, we have $|\nabla f(x^*)_j| = |\nabla f(x^*)_a|$ and thus, $\operatorname{sgn}(\nabla f(\gamma(0))_j)\nabla f(\gamma(0))_j = \operatorname{sgn}(\nabla f(\gamma(0))_a)\nabla f(\gamma(0))_a$. Hence, from (5.26), we can conclude

$$\begin{aligned} &\frac{\operatorname{sgn}(\nabla f(\gamma(t))_j)\nabla f(\gamma(t))_j}{t} - \frac{\operatorname{sgn}(\nabla f(\gamma(0))_j)\nabla f(\gamma(0))_j}{t} \\ \leq\ &\frac{\operatorname{sgn}(\nabla f(\gamma(t))_a)\nabla f(\gamma(t))_a}{t} - \frac{\operatorname{sgn}(\nabla f(\gamma(0))_a)\nabla f(\gamma(0))_a}{t} \end{aligned} \tag{5.27}$$

for all $t \in [0,1)$. Recall that $\operatorname{sgn}(\nabla f(\gamma(t))_j)$ and $\operatorname{sgn}(\nabla f(\gamma(t))_a)$ are constant for $t \in [0,\varepsilon)$. Thus, by (5.27), we have

$$\operatorname{sgn}(\nabla f(\gamma(0))_j)\frac{\nabla f(\gamma(t))_j - \nabla f(\gamma(0))_j}{t} \leq \operatorname{sgn}(\nabla f(\gamma(0))_a)\frac{\nabla f(\gamma(t))_a - \nabla f(\gamma(0))_a}{t}$$

for all $t \in [0,\varepsilon)$. If we consider the limit $t \downarrow 0$ on both sides of (5.27), we get

$$\operatorname{sgn}(\nabla f(\gamma(0))_j)(\nabla^2 f(\gamma(0))\dot\gamma(0))_j \leq \operatorname{sgn}(\nabla f(\gamma(0))_a)(\nabla^2 f(\gamma(0))\dot\gamma(0))_a.$$

Substituting $\gamma(0) = x^*$ and $\dot\gamma(0) = w$ yields

$$\operatorname{sgn}(\nabla f(x^*)_j)(\nabla^2 f(x^*)w)_j \leq \operatorname{sgn}(\nabla f(x^*)_a)(\nabla^2 f(x^*)w)_a.$$

□

Since we do not know what the Pareto critical set looks like, we do not know whether such a curve γ for the activation structure $A(x^*) \cup A_p^l$ exists. Nevertheless, the existence of $\gamma_{|(0,1)}$ follows from the smooth structure of (5.8). Thus, we can use the derived properties of such a curve γ as necessary conditions to prove whether a direction v_1^l (given by (5.23)) is admissible. Note that the existence of γ implies a certain type of regularity of the smooth parts when they reach a kink. For example, roughly speaking, continuous differentiability of γ on an open neighborhood of $t = 0$ implies that the "curvature" of $\mathcal{P}_c$ is bounded. By the construction in (5.23), w is (up to scaling) equal to the direction v_1^l for the index set $A_p^l = A_\gamma \setminus A(x^*)$.

Corollary 5.1.8. *Let $x^* \in \mathcal{P}_c \setminus \{0\}$ with $\nabla f(x^*) \neq 0$, let γ be a curve as in Lemma 5.1.7 and let $a \in A(x^*)$. Furthermore, let v_1^l be the vector computed via (5.23) for the index set $A_p^l = A_\gamma \setminus A(x^*)$. Then, for $w = \sigma v_1^l$ with $\sigma \in \{-1,+1\}$,*

(a) either $w_j = 0$ or $\operatorname{sgn}(w_j) = -\operatorname{sgn}(\nabla f(x^*)_j) \quad \forall j \in A_p^l$, and

(b) $\operatorname{sgn}(\nabla f(x^*)_j)(\nabla^2 f(x^*)w)_j \leq -\sigma(1 + |\nabla f(x)_a|)^2 \quad \forall j \in A_p(x^*) \setminus A_p^l$.

Remark 5.1.9. *According to Corollary 5.1.8(a), for all possible activation structures $A_p^l \subseteq A_p(x^*)$, $A_p^l \neq \emptyset$, there either exists only one valid direction ($+v_1^l$ or $-v_1^l$) or the tangent vector w of the corresponding curve γ has to satisfy $w_j = 0$ for all $j \in A_p(x^*)$, i.e., γ has to be tangential to a vector with activation structure $A(x^*)$, which is unlikely. In addition, in that situation, both directions $+v_1^l$ or $-v_1^l$ have to satisfy the inequality in Corollary 5.1.8(b), i.e., it has to hold $\operatorname{sgn}(\nabla f(x^*)_j)(\nabla^2 f(x^*)v_1^l)_j = -(1 + |\nabla f(x)_a|)^2$ for all $j \in A_p(x^*) \setminus A_p^l$.*

Using the conditions from Corollary 5.1.8, we can eliminate certain possible activation structures and determine the correct sign of the directions corresponding to the remaining structures. These remaining directions can then be interpreted as the initial predictor steps for the new activation structures. Except for the direction we came from, we have to proceed with the continuation method in each of the remaining directions. In Examples 5.1.10 and 5.1.11, we show how Corollary 5.1.8 can be applied.

Example 5.1.10. *Consider* (MOP-ℓ_1) *with*

$$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + (x_3 - 1)^2.$$

Figure 5.2a shows the Pareto critical set for this problem. Consider the Pareto critical point $x^* = (1, 0, 0)^\top$. *Since* $\nabla f(x^*) = (-2, -2, -2)^\top$, *the set of potentially active indices is given by* $A_p(x^*) = \{2, 3\}$, *cf.* (5.21). *According to* (5.23), *we get the following possible directions:*

$$v_1^0 = \begin{pmatrix} 9/2 \\ 0 \\ 0 \end{pmatrix}, v_1^1 = \begin{pmatrix} 9/2 \\ 9/2 \\ 0 \end{pmatrix}, v_1^2 = \begin{pmatrix} 9/2 \\ 0 \\ 9/2 \end{pmatrix}, v_1^3 = \begin{pmatrix} 9/2 \\ 9/2 \\ 9/2 \end{pmatrix},$$

corresponding to $A_p^0 = \emptyset$, $A_p^1 = \{2\}$, $A_p^2 = \{3\}$, *and* $A_p^3 = \{2, 3\}$. *After checking Condition (a) in Corollary 5.1.8, we know that only*

$$v_1^0, -v_1^0, v_1^1, v_1^2 \text{ and } v_1^3$$

remain as suitable directions. By checking the inequality in (b) in Corollary 5.1.8, the set of admissible directions can be reduced to

$$-v_1^0 \text{ and } v_1^3.$$

Assuming that we started in the origin, we have to walk in direction v_1^3, *i.e., indices* 2 *and* 3 *become active (and positive).*

Example 5.1.11. *Consider* (MOP-ℓ_1) *with*

$$f(x) = (x_1 - 2)^2 - x_1(x_2 - 1)^2 + 2x_1x_3 + 2x_2^2x_1.$$

The relevant part of $\mathcal{P}_c$ *for this problem is shown in Figure 5.2b. Consider the Pareto critical point* $x^* = (1.25, 0, 0)^\top$. *Since* $\nabla f(x^*) = (-2.5, 2.5, 2.5)^\top$, *the set of potentially active indices is given by* $A_p(x^*) = \{2, 3\}$. *According to* (5.23), *we get as possible directions:*

$$v_1^0 = \begin{pmatrix} 49/8 \\ 0 \\ 0 \end{pmatrix}, v_1^1 = \begin{pmatrix} 441/8 \\ -49 \\ 0 \end{pmatrix}, v_1^2 = v_1^3 = \begin{pmatrix} -49/8 \\ 0 \\ 49/4 \end{pmatrix},$$

corresponding to $A_p^0 = \emptyset$, $A_p^1 = \{2\}$, $A_p^2 = \{3\}$, *and* $A_p^3 = \{2, 3\}$. *After checking the Conditions (a) and (b) in Corollary 5.1.8, only* $-v_1^0$, $-v_1^2$ *and* $-v_1^3$ *remain as*

admissible directions. Thus, assuming that we started in the origin, we have to proceed with the continuation into the directions $-v_1^2$ and $-v_1^3$. After a predictor step along the direction $-v_1^3$, the result of the subsequent corrector step is x^*, i.e., the continuation in this direction terminates immediately. Therefore, the only remaining direction is $-v_1^2$. Thus, only $j = 3$ is activated.

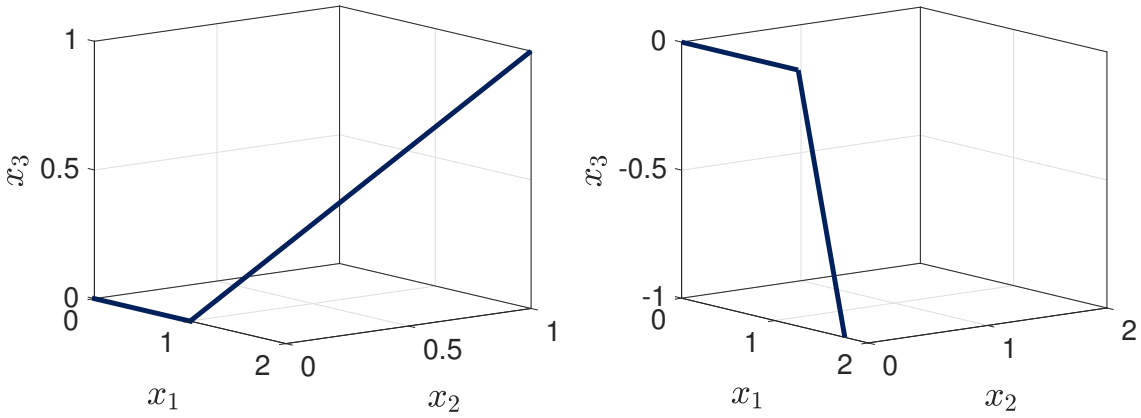


Figure 5.2: (a) $\mathcal{P}_c$ for the MOP in Example 5.1.10. (b) Part of $\mathcal{P}_c$ for the MOP in Example 5.1.11.

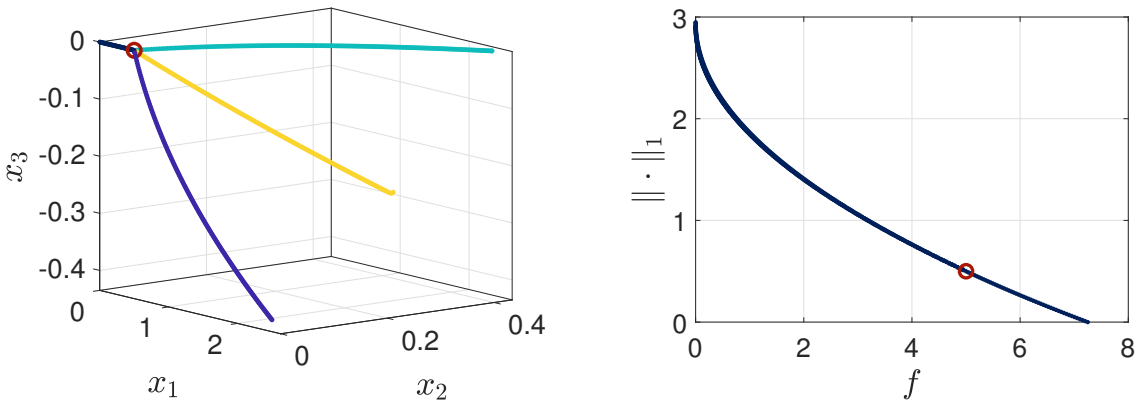
Typically, only one direction, excluding the old direction one comes from, remains after applying Corollary 5.1.8. If this is not the case, there is usually only one direction left after the first correction step, cf. Example 5.1.11. However, there are some situations where the continuation can be continued in more than one direction, i.e., the Pareto critical set splits. The following example shows such a situation.

Example 5.1.12. *Consider* (MOP-ℓ_1) *with*

$$f(x) = (x_1 - 2.5)^2 + ((x_2 + 1)^2 + (x_3 - 1)^2 - 3)^2.$$

The Pareto critical set $\mathcal{P}_c$ and its image are presented in Figure 5.3. We see that in the point $x^* = (0.5, 0.0, 0.0)^\top$, the Pareto critical set splits into three parts, visualized in different colors. In the blue part, only the indices 1 and 3 are active. In the turquoise part, the active set consists of the indices 2 and 3, whereas all indices are active in the yellow part. Obviously, this is caused by the symmetry in x_2 and x_3, i.e., by the fact that $f((x_1, x_2, x_3)^\top) = f((x_1, -x_3, -x_2)^\top)$. Note that all three parts have the same image, cf. Figure 5.3b.

If no direction satisfies the conditions in Corollary 5.1.8, we have reached the end of a connected component of $\mathcal{P}_c$. In the general smooth case, assuming that the weighted Hessian matrix $\sum_{i=1}^k \alpha_i \nabla^2 f_i(x)$ in (2.23) is regular, which implies that DH has full rank, a connected component of the Pareto critical set can only end in a critical point of one of the objective functions [GPD19]. For (MOP-ℓ_1), we have to distinguish between two cases. If all indices are active, the end of the component can only be reached if $\nabla f(x) = 0$ as both objectives are smooth in this part, and



(a) (b)

Figure 5.3: (a) $\mathcal{P}_c$ for the MOP in Example 5.1.12. In the red point, $\mathcal{P}_c$ splits into the blue, turquoise and yellow part. (b) The image of $\mathcal{P}_c$, with the image of the point where $\mathcal{P}_c$ splits marked in red.

the gradient of the ℓ_1-norm cannot become zero. Otherwise, $\nabla f(x) = 0$ means that all inactive indices are potentially active, i.e., $A_p(x) = \{1, \dots, n\} \setminus A(x)$. Besides the fact that the set $A_p(x)$ may be extremely large, Corollary 5.1.8 is not applicable. This situation is discussed in more detail in the following remark.

Remark 5.1.13. *Assume that we are in a point x^* with $\nabla f(x^*) = 0$ and $A_p(x^*) \neq \emptyset$, i.e., there are inactive indices. In the case that (a part of) the Pareto critical set continues in the current activation structure, such a point is likely missed during the continuation due to the discretization. If $\mathcal{P}_c$ does not continue along the current activation structure, the corrector will end up in x^*. In this case, we would have to consider every possible combination of the signs of potentially active indices for every possible activation structure A_p^l since we cannot predict the sign of x_j with $j \in A_p(x^*)$ via the sign of the gradient. Although the requirements of Corollary 5.1.8 are not satisfied, similar conditions can be derived based on the directional derivative in direction v_1^l to ensure that the conditions derived in Theorem 5.1.2 are satisfied. First of all, $\operatorname{sgn}((\nabla^2 f(x^*)w)_j) = -\operatorname{sgn}(x_j^*)$ has to hold for all $j \in A(x)$. Furthermore, for the newly active indices $j \in A_p^l$, we get $\operatorname{sgn}((\nabla^2 f(x^*)w)_j) = -\operatorname{sgn}(w_j)$ (or $w_j = 0$). As last condition, replacing Corollary 5.1.8(b), $|(\nabla^2 f(x^*)w)_j| \leq |(\nabla^2 f(x^*)w)_a|$ has to be satisfied for all indices $j \in A_p(x^*) \setminus A_p^l$, i.e., for all indices staying inactive. Although these conditions would be easy to check, the number of inactive indices tends to be large (since we are looking for sparse solutions), resulting in a vast number of directions that have to be computed and for which the conditions need to be checked. Therefore, for this situation, a different strategy would be required. Nevertheless, it seems quite unlikely that a critical point of the objective f is of such a nature that the activation structure has to change in it. Hence, in practice, it is reasonable to stop the continuation if a point x^* with $\nabla f(x^*) = 0$ is reached, and compute a new starting point x_{start} for the continuation method.*

As discussed in the previous remark, in the case that $\nabla f(x) = 0$ is satisfied on a part with constant activation structure, this point is probably missed by the continuation

method, i.e., if the Pareto critical set would split in this point, we probably would not recognize the other parts of the Pareto critical set. This is obviously not only the case for $\nabla f(x) = 0$ but can happen in general if $\mathcal{P}_c$ splits up and one part remains in the old activation structure. But based on Corollary 5.1.8 and Remark 5.1.9, one can show that such points are unlikely to exist in practice, as discussed in the following remark.

Remark 5.1.14. *During the normal continuation, a point x^* where the Pareto set splits into two (or more) parts could only be missed if one part of the Pareto critical set remains in the old activation structure. According to Corollary 5.1.8 this would only be possible if both directions for the activation structure $A(x^*)$, i.e., for $A_p^0 = \emptyset$, are allowed. According to Corollary 5.1.8(b), for the direction v_1^0 computed via (5.23) and the appropriate embedding $l^A : \mathbb{R}^{n^A(x^*)} \to \mathbb{R}^n$ and projection $p^A : \mathbb{R}^n \to \mathbb{R}^{n^A(x^*)}$ as in Corollary 5.1.4, we have*

$$\begin{aligned} &\operatorname{sgn}(\nabla f(x^*)_j)(\nabla^2 f(x^*)v_1^0)_j = -(1 + |\nabla f(x^*)_a|)^2 \ \forall j \in A_p(x^*) \\ \Leftrightarrow\ &\left(\nabla^2 f(x^*) \cdot l^A \left((\nabla^2 f(x^*)_{|A(x^*)})^{-1} s\right)\right)_j = -\operatorname{sgn}(\nabla f(x^*)_j) \quad \forall j \in A_p(x^*), \end{aligned} \tag{5.28}$$

where $s = \operatorname{sgn}(p^A(x^*))$. Now, assume that the conditions in Corollary 5.1.8 are satisfied for a curve γ with activation structure $A_\gamma = A(x^*) \cup \bar{A}_p$, where $\bar{A}_p \subseteq A_p(x^*)$ and $\bar{A}_p \neq \emptyset$, i.e., we assume that the Pareto set splits and it exists an additional part with different activation structure. Since $\nabla^2 f(x^*)_{|A(x^*)}$ and $\nabla^2 f(x^*)_{|A_\gamma}$ are invertible by assumption, we can use the Schur-complement to compute the new direction $\bar{v}_1^\gamma$ according to (5.23) for the activation structure A_γ. Without loss of generality, we assume that the indices are sorted in such a way that

$$\nabla^2 f(x^*)_{|A_\gamma} = \begin{pmatrix} H_A & C^\top \\ C & D \end{pmatrix},$$

where $H_A = \nabla^2 f(x^*)_{|A(x^*)}$. Before we derive an expression for $\bar{v}_1^\gamma$, we note that (5.28) has to be satisfied in particular for $j \in \bar{A}_p$, i.e., it has to hold

$$s^* = \begin{pmatrix} C & D \end{pmatrix} \begin{pmatrix} H_A^{-1} s \\ 0 \end{pmatrix} = C H_A^{-1} s,$$

where $s^* = -\operatorname{sgn}(\nabla f(x^*)_{|\bar{A}_p})$. Thus, by using the Schur-complement and (5.23), we get

$$\begin{aligned} &\frac{\bar{v}_1^\gamma}{(1 + |\nabla f(x^*)_a|)^2} = \left(\nabla^2 f(x^*)_{|A_\gamma}\right)^{-1} \begin{pmatrix} s \\ s^* \end{pmatrix} \\ &= \begin{pmatrix} H_A^{-1} + H_A^{-1} C^\top (D - C H_A^{-1} C^\top)^{-1} C H_A^{-1} & -H_A^{-1} C^\top (D - C H_A^{-1} C^\top)^{-1} \\ -(D - C H_A^{-1} C^\top)^{-1} C H_A^{-1} & (D - C H_A^{-1} C^\top)^{-1} \end{pmatrix} \begin{pmatrix} s \\ s^* \end{pmatrix} \\ &= \begin{pmatrix} H_A^{-1} s + H_A^{-1} C^\top (D - C H_A^{-1} C^\top)^{-1} (s^* - s^*) \\ (D - C H_A^{-1} C^\top)^{-1} (-s^* + s^*) \end{pmatrix} \\ &= \begin{pmatrix} H_A^{-1} s \\ 0 \end{pmatrix}. \end{aligned}$$

According to (5.23), the tangent vector $\bar{v}_1^0$ for the activation structure $A(x^*)$ is given by $\bar{v}_1^0 = (1 + |\nabla f(x^*)_a|)^{-2} H_A^{-1} s$. Thus, we have $\bar{v}_1^\gamma = (\bar{v}_1^0, 0)^\top$. Since $\bar{v}_1^\gamma$ is the tangent vector of γ in x^*, this implies that the curve γ has to be tangential to the part of the Pareto critical set with activation structure $A(x^*)$. Although it is difficult to prove, we believe that this is an unlikely scenario. Therefore, the likelihood of missing a point where the activation structure changes is generally small as long as the respective Hessian matrices are invertible.

5.1.5 The Algorithm

By combining the predictor, the corrector, and the strategies to activate or deactivate indices, we are able to compute the connected component of the Pareto critical set that contains our starting point x_{start}. A more detailed version of Algorithm 4 is given in Algorithm 5 to sum up the presented results.

Algorithm 5 Continuation for (MOP-ℓ_1).

Input: $f : \mathbb{R}^n \to \mathbb{R} \in C^2$, $x_{\text{start}} \in \mathbb{R}^n$ satisfying (2.13)
Output: Discretization of the Pareto critical set $\mathcal{P}_{\text{PC}}$

$\mathcal{P}_{\text{PC}} = \emptyset$, $x^0 = x_{\text{start}}$, $A = \{j \in \{1, \dots, n\} : x_j^0 \neq 0\}$
while End of $\mathcal{P}_c$ component has not been reached **do**
  $\mathcal{P}_{\text{PC}} = \mathcal{P}_{\text{PC}} \cup \{x^0\}$
  if $A_p(x^0) = \emptyset$ (cf. (5.21)) **then**
    Predictor step:
      Compute direction $v_1 = l^A(\bar{v}_1)$ using (P),
      Determine sign of v_1 using (5.18),
      Compute the step size along $\pm v_1$ using (5.19),
      Determine $x^p = x^0 + h \cdot (\pm v_1)$.
    Corrector step:
      Compute $x^c \in \mathcal{P}_c$ by solving (C) with initial value x^p.
    $x^0 = x^c$
  else
    (De-)Activate entries:
      Calculate v^l for all $A_p^l \subseteq A_p(x^0)$ using (5.23),
      Reduce number of directions using Corollary 5.1.8,
      Proceed with the corrector step for all remaining directions with $A = A_p^l$.
  end if
end while

As already mentioned at the beginning of this section, the most intuitive choice for the starting point would be $x_{\text{start}} = 0$ since it is Pareto critical and even Pareto optimal as the global minimum of the ℓ_1-norm. According to Theorem 5.1.2, the first index that has to be activated in $x_{\text{start}} = 0$ is the one that corresponds to the largest absolute value of the gradient. The sign of the gradient entry furthermore tells us whether we have to proceed in the positive or negative direction. If there are multiple maxima, we can apply the strategy described in Section 5.1.4 to decide which indices should be activated. Obviously, any other Pareto critical point can be

chosen as a starting point, as well. In this case, running the continuation method in both directions may be useful, i.e., in the very first predictor step, the method is proceeded for v_1 and $-v_1$.

Often, the Pareto critical set consists of more than one connected component. Hence, multiple starting values (at least one for each connected component) are usually required. One possible approach would be to directly perform a *multi-start*, i.e., to create (more or less randomly) multiple Pareto critical points to start from. In practice, it may be more beneficial to compute an appropriate new starting value when one connected component is completely computed. To this end, methods acting in the objective space can be useful. Usually, we are interested in a new solution with a smaller value of f and a larger ℓ_1-norm or vice versa (depending on whether we reached the lower or the upper end of the image of the component). To this end, the ε-constraint method (cf. Section 2.3.2) can be used or, alternatively, the so-called *reference point methods* (cf. [Ehr05]) may be utilized, where an optimal solution is computed that is near an appropriate chosen *reference point* in the objective space.

Lastly, it is important to note that the continuation method only computes points that are Pareto *critical*, but not necessarily Pareto *optimal*. To verify whether the computed points are really optimal, in theory, a consecutive *non-dominance test* [Sch03] can be applied to the approximation of the Pareto critical set.

5.2 Numerical Results

Now we want to evaluate the presented continuation method by means of some numerical examples. We begin with two toy examples to illustrate the basic behavior, where the second one is of higher dimension ($n = 1000$). Afterwards, we compute the Pareto critical set for an example motivated by the SINDy approach mentioned in the introduction of this chapter [BPK16]. Finally, the use of our method in the context of NN training is studied in more detail.

5.2.1 Toy Examples

As a first illustrating example, we consider a low-dimensional (nonconvex) problem.

Example 5.2.1. *Consider* (MOP-ℓ_1) *where the first objective is given by*

$$f(x) = \left(x_1 - \tfrac{1}{4}\right)^2 + \left(x_2 - \tfrac{1}{2}\right)^2 + (x_3 - 1)^4 - \tfrac{1}{2}\left(x_3 - \tfrac{1}{4}\right)^3.$$

The result of the continuation method (which coincides with the analytic solution) is presented in Figure 5.4. We can observe that indices are activated and deactivated multiple times. Points where the activation structure changes are $x^1 = (0, 0, 0.375)^\top$, $x^2 = (0, 0.25, \frac{83}{151})^\top$, $x^3 \approx (0, 0.25, 0.81)^\top$, $x^4 \approx (0, 0, 1.07)^\top$, $x^5 \approx (0, 0, 1.93)^\top$, *as well as* $x^6 \approx (0, 0.25, 2.014)^\top$. *Note that with the standard weighted sum method, i.e., by solving* (5.1)*, only the convex parts of* $\mathcal{P}_c$ *can be computed. In particular,*

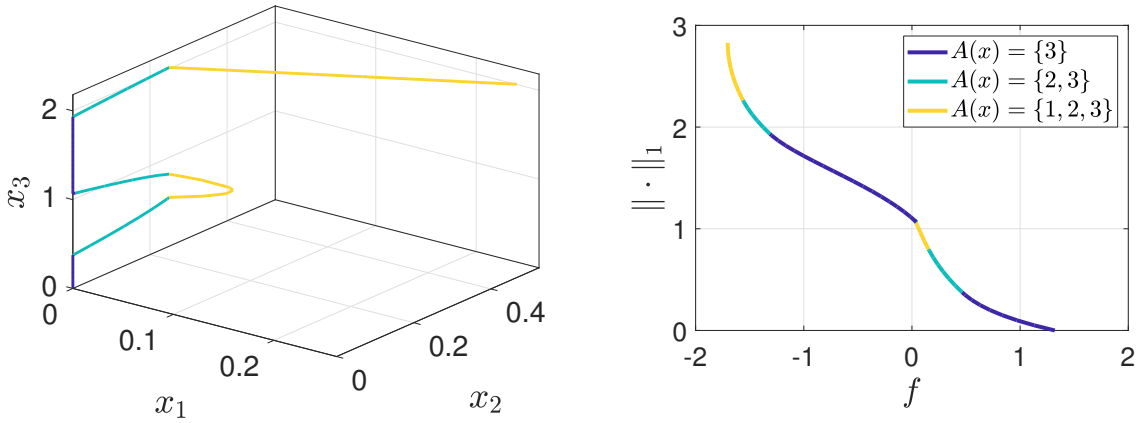


Figure 5.4: (a) $\mathcal{P}_c$ for the MOP in Example 5.2.1, where the various activation structures are differently colored. (b) Image of $\mathcal{P}_c$ with the corresponding coloring.

a large part where only $j = 3$ is active would be missed, which again highlights the necessity for a method that is able to handle a nonconvex function f.

In the next example, we want to study the applicability of the continuation method to higher dimensional problems. Therefore, we consider a (relatively simple) polynomial function f in the space $\mathbb{R}^{1000}$, motivated by Example 1 in [SDD05].

Example 5.2.2. *Consider* (MOP-ℓ_1) *with*

$$f(x) = (x_1 - a_1)^4 + \sum_{i=2}^{n} (x_i - a_i)^2,$$

where $a_1 = 1$ and a_i, $i \in \{2, \dots, n\}$, are randomly chosen values in the interval $[-1, 1]$. Figure 5.5 shows the Pareto front resulting from the continuation method for $n = 1000$ with step size $\tau = 0.25$ (which coincides with the analytical solution). Note that the computation of $\mathcal{P}_c$ does not suffer from the "curse of dimensionality" (with respect to n), since the dimension of the Pareto set depends on the number of objective functions k and is thus still one.

5.2.2 SINDy

In the next example, we consider the SINDy approach [BPK16], already briefly mentioned in the introduction of this chapter. Similar to the surrogate models dealt with in Chapters 3 and 4, it aims at building an approximation of a dynamical system given by

$$\dot{\mathbf{y}}(t) = g(\mathbf{y}(t)),$$

where $\mathbf{y} : [t_0, t_f] \to Y \subseteq \mathbb{R}^{n_y}$ is the state function and $g : Y \to Y$ defines the system dynamics. More precisely, the idea of SINDy is to represent g via a dictionary of c ansatz functions (denoted by $\Theta : Y \to \mathbb{R}^c$) with the corresponding coefficients $X \in \mathbb{R}^{c \times n_y}$:

$$g^r(\mathbf{y}(t)) = X^\top \Theta(\mathbf{y}(t)).$$

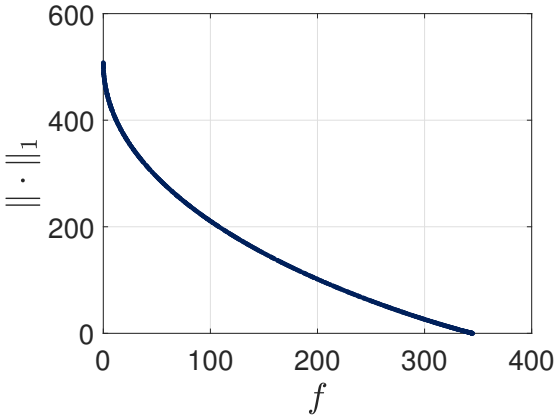


Figure 5.5: Pareto front for the MOP in Example 5.2.2.

The entries of Θ are arbitrary linear and nonlinear functions such as sine and cosine or polynomials. For the SINDy approach, we assume that N (potentially noisy) training data points of the form $((y^1, \dot{y}^1), ..., (y^N, \dot{y}^N))$ are given, i.e., we assume that we know (or can approximate) the derivative of the state y. Using these data points, the objective is to identify a sparse representation of g in terms of Θ, and to obtain a robust and interpretable dynamical system from data this way. The corresponding main objective is thus the following least squares term:

$$f(X) = \frac{1}{n_y c N} \sum_{i=1}^{N} \left\| X^\top \Theta(y^i) - \dot{y}^i \right\|_2^2 . \tag{5.29}$$

As this is a convex function, the corresponding MOP is also convex and can thus be addressed using the standard penalization approach, i.e., the weighted sum method $\min_X f(X) + \lambda \|X\|_1$ (where X is transformed to one large weight vector). However, the solution to this problem may be highly sensitive with respect to λ, which makes choosing an appropriate λ difficult.

As a concrete example, we consider again the Lorenz system (without an additional control input) with parameters $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$, for which the dynamical system possesses a chaotic solution:

$$\dot{\mathbf{y}} = \begin{pmatrix} \sigma \cdot (\mathbf{y_2} - \mathbf{y_1}) \\ \mathbf{y_1} \cdot (\rho - \mathbf{y_3}) - \mathbf{y_2} \\ \mathbf{y_1} \cdot \mathbf{y_2} - \beta \cdot \mathbf{y_3} \end{pmatrix} .$$

We collect $N = 10\,000$ training data points of a single long-term simulation to which we add white noise. The dictionary Θ consists of polynomial terms up to order 3 and thus $c = 20$ terms per dimension, i.e., $X \in \mathbb{R}^{20\times 3}$.

The result obtained by the new continuation method for the corresponding MOP, i.e., for (MOP-ℓ_1) where f is given by (5.29), is shown in Figure 5.6. We see that, although the objectives are convex, choosing the weight λ in a penalization approach can be very challenging. Large parts of the Pareto front are almost parallel to either the horizontal or vertical axis, such that a slight change in the penalty weight leads

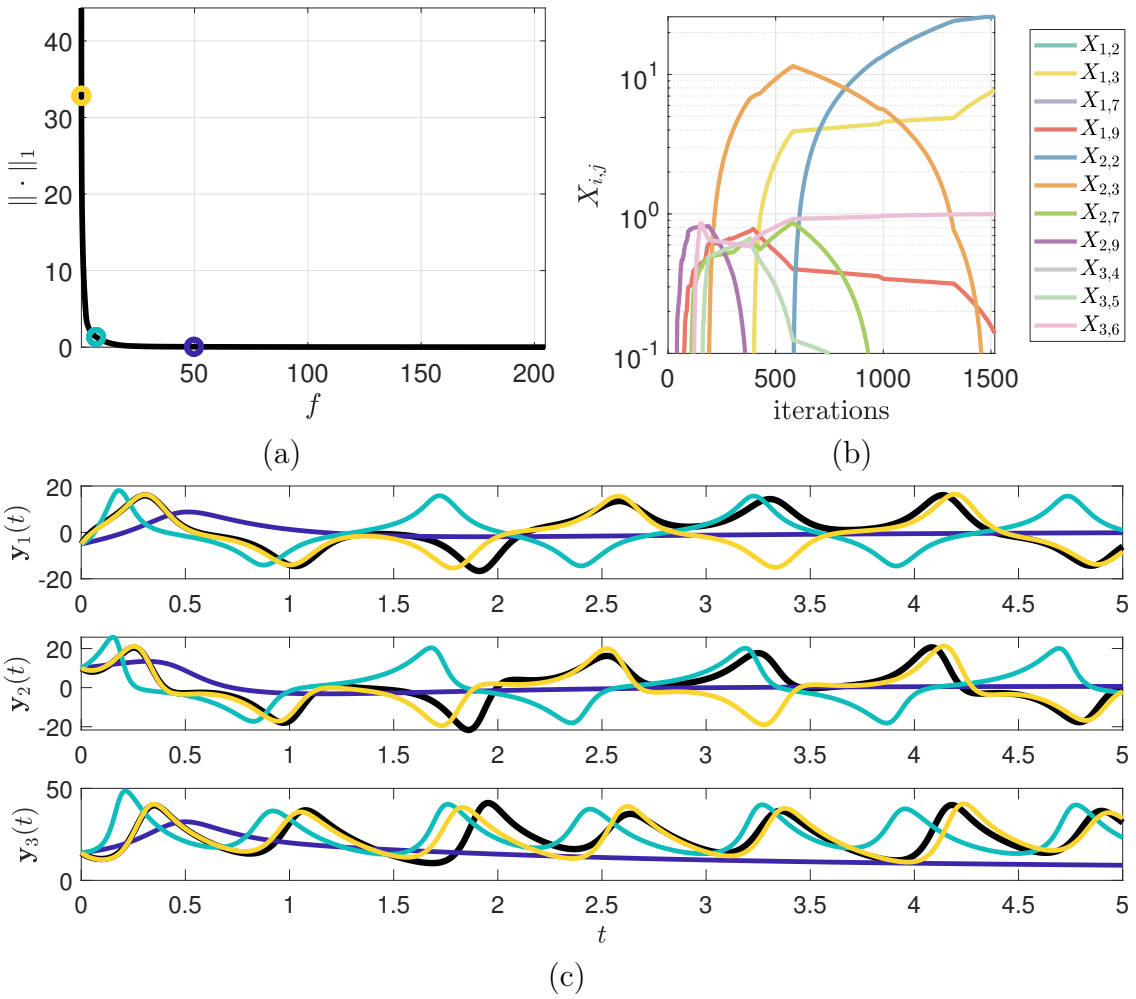


Figure 5.6: (a) Convex Pareto front for the SINDy approach (f of (MOP-ℓ_1) is given by (5.29)) applied to the Lorenz system. (b) Evolution of all $X_{i,j}$ that were greater than 0.5 in at least one iteration step. (c) Trajectory of the Lorenz system (black) and the resulting trajectories of the approximated solutions, where the corresponding point on the Pareto front is marked in (a).

to a significant deviation in the image space. In particular, for most of the weights λ, the solution of the penalization approach is close to the so-called *knee point*. A knee point is a point at which a small decrease in one objective function leads to a large increase in the other objective [Bra+04], which in the example here is (approximately) the turquoise point. In many applications, knee points represent desirable solutions when choosing a point from the Pareto set. However, the solution belonging to the yellow point is clearly better than the solution corresponding to the turquoise point, i.e., here, the knee point is not the desired solution.

5.2.3 Neural Network

Now we come to the main motivation for developing the presented continuation method – the training of neural networks. We consider a fully connected FNN,

recall Section 2.2.2. Thus, the equations of the NN are given by

$$\begin{aligned} h_0 &= a, \\ h_j &= \sigma_h\left(W_j^\top h_{j-1} + w_j^0\right) \quad \text{for } j \in \{1, \dots, K\}, \\ b &= \sigma_o(\underbrace{W_{K+1}^\top h_K + w_{K+1}^0}_{=h_{K+1}}), \end{aligned}$$

where K is the number of hidden layers, and σ_h and σ_o are the activation functions for the hidden layers and the output layer, respectively. The parameters that can be optimized are the weights $W_j \in \mathbb{R}^{n_{j-1} \times n_j}$ and the biases $w_j^0 \in \mathbb{R}^{n_j}$, where n_j is the number of neurons of the j-th layer. Accordingly, n_0 and n_{K+1} are equal to the input and output dimension, respectively. For simplicity of notation, we combine these into a single variable $x = ((W_1, w_1^0) \dots, (W_{K+1}, w_{K+1}^0))$ that we identify with a vector in $\mathbb{R}^n$, where n is the number of (trainable) parameters of the NN. In this concrete example, we consider an FNN with $K = 2$ hidden layers to solve the classification task of the well-known Iris data set [Fis36], cf. Section 2.2.1. Each hidden layer consists of two neurons, and the activation function for the hidden layers is given by $\sigma_h = \tanh$. For the output layer, we choose the softmax function, i.e.,

$$\sigma_o(h_{K+1})_i = \frac{\exp(h_{K+1})_i}{\sum_{j=1}^{n_{K+1}} \exp(h_{K+1})_j} \quad \text{for } i \in \{1, \dots, n_{K+1}\}. \tag{5.30}$$

Note that the output dimension n_{K+1} is equal to the number of possible classes in the output set $\mathcal{B}$ of the data set, i.e., $n_{K+1} = 3$, such that the total number of weights is $n = 25$ $[= (4 \cdot 2 + 2) + (2 \cdot 2 + 2) + (2 \cdot 3 + 3)]$. Since we consider a classification task, the loss function is chosen as the mean of the cross-entropy loss, cf. Equation (2.6). Figure 5.7 shows the Pareto front obtained via the continuation method applied to three different starting values, i.e., we computed three connected components of the Pareto critical set and applied a non-dominance test afterwards. As an initial starting value, we chose $x_{\mathsf{start}} = 0$ as the global minimum of the ℓ_1-norm. After computing the corresponding connected component, the ε-constraint method was applied to find a new suitable starting value. This was repeated after the computation of the second connected component. As (constant) step size, $\tau = 0.1$ was chosen. Note that the single connected components do not appear as distinct connected components in the objective space since their images intersect.

To compare the result to the classical training via SGD applied to the regularized objective $f(x) + \lambda\|x\|_1$, several solutions were computed using the Adam algorithm, cf. Section 2.2.2. To this end, random initial conditions, as well as varying weights λ, were used. Furthermore, a fixed number of 5000 epochs was chosen. (Note that this way, the influence of early stopping to avoid overfitting is left out.) As expected, too large values of λ lead to the calculation of solutions close to $x = 0$. These cause a large training error since they cannot represent any information of the training data. We see that the Adam optimizer usually ends up in points close to the calculated Pareto front, but the solutions cluster in two places, and Pareto optimal points corresponding to the middle part of the Pareto front are not obtained. This

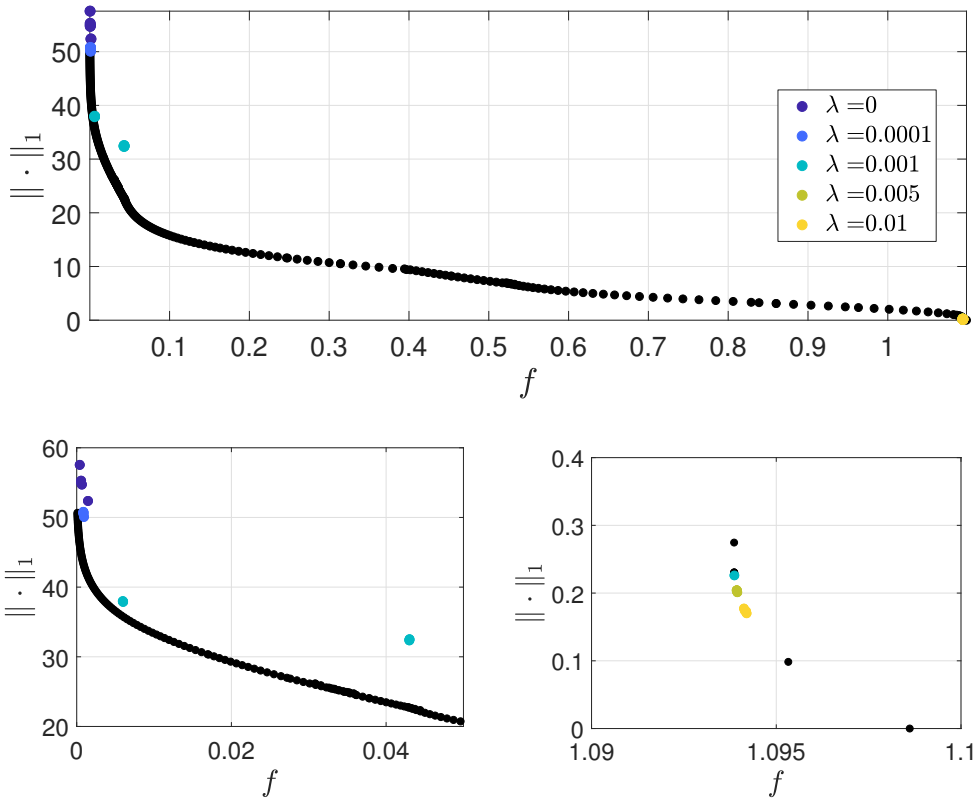


Figure 5.7: Pareto front computed via continuation for the NN training on the Iris data set (black) in (a). The colored dots show solutions obtained via the Adam algorithm with different weights for the ℓ_1-penalty term. These cluster in two places, cf. the two zoomed-in plots at the bottom.

is probably caused by the fact that large parts of the Pareto front are nonconvex, as marked in Figure 5.8. Furthermore, some parts of the convex segments of the Pareto front have a very low curvature, which implies that the subset of weights λ that need to be chosen to find these segments is small, i.e., it is difficult to find a suitable λ and to compute the solutions in a numerically stable way. Moreover, for $\lambda = 10^{-3}$, we obtain solutions in both clusters, depending on the initial condition. This is probably caused by the randomness induced by the batch-wise learning in the Adam optimizer or by the choice of the initial weights. This shows again the difficulties of the standard penalty term approach and the advantage of the continuation method, which still provides good compromise solutions in this case.

However, when training NNs, one is not directly interested in the training error but instead would like to find solutions that do not lead to overfitting. To assess this, we need to consider the test error on unseen test data, cf. Section 2.2.1. The test error for the solutions computed via continuation and the Adam algorithm are shown in Figure 5.9. While both approaches are capable of finding solutions with similar low test errors, we see that the continuation method finds a solution with a significantly smaller ℓ_1-norm. Moreover, since the test error of the Adam solutions for $\lambda = 10^{-3}$ varies strongly, the continuation method is more robust.

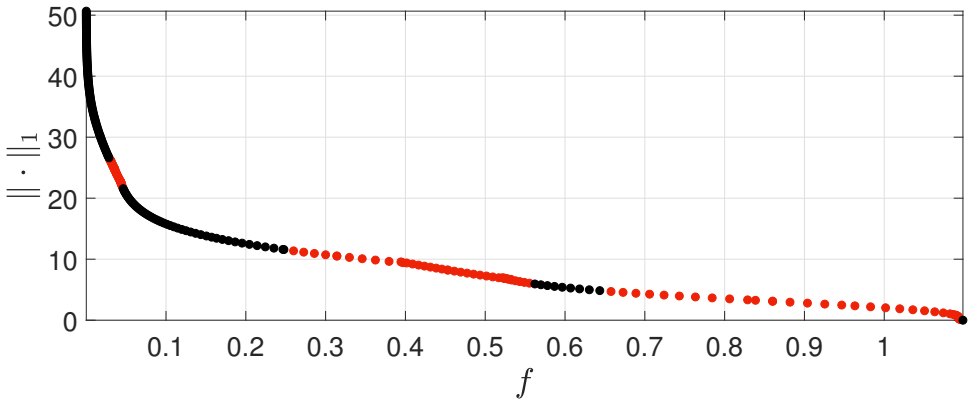


Figure 5.8: Pareto front computed via continuation for the NN training on the Iris data set where the nonconvex parts are marked in red.

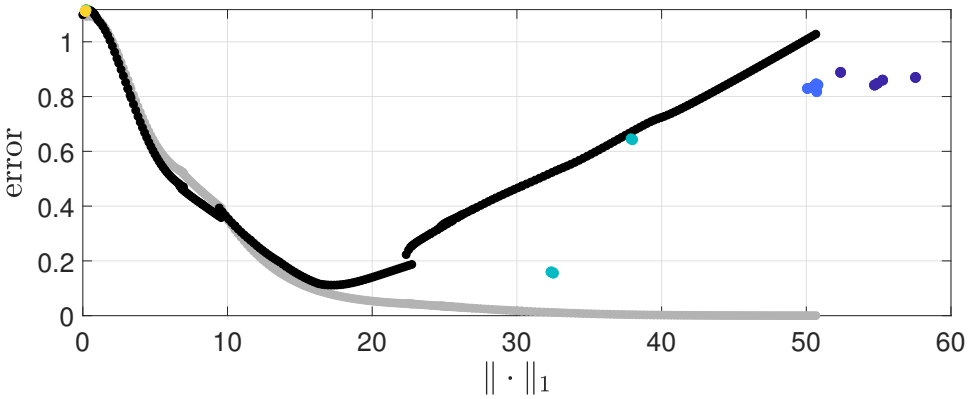


Figure 5.9: Test error (black) and training error (grey) against ℓ_1-norm for the Pareto optimal points computed via the continuation method (and the subsequent non-dominance test). The colored dots represent the test error of the corresponding Adam solutions shown in Figure 5.7.

Furthermore, in terms of the runtime, the continuation method is not necessarily slower than the penalty approach solved via Adam since the Adam solver has to be started multiple times to derive good solutions. Nevertheless, the presented continuation method lacks scalability to higher dimensions. How this issue might be overcome is discussed in the subsequent section. In addition, note that in a real training process, validation data has to be used to choose the best solution. The performance of the derived models can then be evaluated based on unseen test data.

5.3 Towards High-Dimensional Problems

Apart from the polynomial toy example, the examples presented in the last section were of relatively small dimension. As briefly mentioned at the end of that section, the proposed method does not scale well to larger dimensions. Especially in the training of NNs, we encounter some difficulties mainly caused by the numerous

symmetries in the networks, which also turn up in greater numbers as the dimension increases. It is well-known that symmetries are an issue for continuation methods in general [AG90]. It is also common knowledge in the area of machine learning that it is important to pay attention to them. There, the main issue is that once a symmetry occurs in the parameters, it cannot be eliminated since the corresponding gradient entries stay equal [GB10]. Typically, this is taken into account by an appropriate initialization that avoids (unwanted) symmetries in the standard training algorithms.

The main issues we face when dealing with higher dimensional problems in the presented continuation method are the following:

1. Although we only need to consider the reduced Hessian matrix, i.e., the rows and columns corresponding to nonzero entries of the current activation structure, the computation of the reduced Hessian and its inverse become very expensive with an increasing number of active indices n^A (which usually also increases if the problem dimension n increases).

2. The necessary conditions for activating indices, cf. Corollary 5.1.8, often yield a unique direction in which to continue. Unfortunately, there are also cases where several directions remain, and the Pareto critical set splits into multiple branches, cf. Example 5.1.12. In particular, this is the case if f possesses symmetries, as it is often the case for NNs. With an increasing dimension, such situations are more likely to occur, and it becomes more expensive to compute all Pareto critical branches.

3. The reduced Hessian matrix may become singular in certain situations. This is particularly an issue when there are multiple potentially active indices, and one has to prove which directions are suitable (with the necessary conditions from Corollary 5.1.8). Similar to the previous point, singular Hessian matrices can be caused by symmetries which, especially in the case of neural networks, occur much more frequently if the problem is of high dimension.

4. With an increasing dimension of the variable x, the ε-constraint method, which we use to find a new component, becomes more expensive and sometimes numerically unstable.

There are different possibilities to overcome these issues. The most obvious is to use approximations of the Hessian instead of the exact one to address Issue 1. For instance, during the optimization in the corrector step, a BFGS or SR1 update, cf. [NW06], can be used. The final approximation of the Hessian can then also be used in the subsequent predictor step. Furthermore, we can directly compute the inverse by a rank-1-update. First experiments suggest that the SR1 update may be more suitable in our case. A similar approach (using the BFGS update rule) is followed in [Mar14; MS17].

To handle Issue 2, we can apply a greedy strategy, i.e., we proceed only in the first suitable direction and ignore the others. This is particularly reasonable for splitting caused by symmetries where the branches of $\mathcal{P}_c$ correspond to the same part of the Pareto front and thus lead to equivalent well-suited solutions. If we consider neural

networks, due to the symmetric graph structure, we often face symmetries of the form that $f(\nu, 0) = f(\beta\nu, (1-\beta)\nu)$ for all $\beta \in [0,1]$, $\nu \in \mathbb{R}$, i.e., $f(\beta\nu, (1-\beta)\nu)$ has the same value for all $\beta \in [0,1]$. Furthermore, the value of the ℓ_1-norm is also constant, i.e., $\left\|(\beta\nu, (1-\beta)\nu)^\top\right\|_1 = |\nu|$ for all $\beta \in [0,1]$. Hence, the Pareto critical set is not one-dimensional where this symmetry condition holds. If both indices are activated, from the symmetry of f as above, it follows that the Hessian matrix is singular. By activating only one of the indices, the reduced Hessian matrix is only one-dimensional and may be invertible (unless it is zero). Thus, Issue 3 is at least partially addressed. Nevertheless, it is an open problem how to continue if it is necessary to consider an activation structure for which the reduced Hessian matrix is not regular.

The most efficient way to solve Issue 4 is to compute only the connected component that is relevant for the considered application by computing an appropriate x_{start}. For instance, in the case of NNs, a scalarization method in combination with an efficient training algorithm can be used, such as the weighted sum solved with Adam. However, it remains open how to ensure that a suitable solution is found. Alternatively, additional knowledge (e.g., regarding the influence of bias neurons on the number of connected components in an NN) may help to find suitable transitions to neighboring connected components, but this is left as a topic for future work.

In the following example, we apply the above suggestions to train a larger NN to learn an adaption of the well-known MNIST data set [LCB10].

Example 5.3.1 (NN for reduced MNIST)**.** *As in Section 5.2.3, we consider an FNN and want to solve a classification task. Here, we consider the MNIST data set, which consists of images with* 28×28 *pixels showing handwritten digits between* 0 *and* 9*. Without adaptions, this would result in a vast number of parameters. (Even for one hidden layer and one neuron, we already would have* 805 *parameters.) Therefore, we only take images labeled with "*3*" or "*6*" and reduce the number of pixels to* 6×6 *by using bilinear interpolation, see, for instance, [GW08a]. We consider an NN with* $K = 2$ *hidden layers consisting of* 4 *neurons each. This results in* $n = 173$ *parameters to train. This time, we choose the softplus activation function for the hidden layers, i.e.,* $h(\cdot) = \log(\exp(\cdot) + 1)$ *(applied elementwise to the input vector). For the output layer, we use again the softmax function, cf.* (5.30)*.*

Figure 5.10a shows the Pareto front obtained via the continuation method when starting in $x_{\text{start}} = 0$ *with* $\tau = 0.25$*. Similar to Section 5.2.3, we computed three components and used a non-dominance test to obtain the results. The shown Adam solutions are obtained with* 500 *epochs.*

We observe again that the Adam solutions cluster in two places, and solutions in both clusters are derived for $\lambda = 0.0005$*. Furthermore, the solutions are significantly less sparse than those obtained via the continuation. If we assume that the Adam solver converged to Pareto optimal solutions, this part is missing in the solution derived by the continuation method. To compute this part, an appropriate additional starting value would be required. However, as can be seen in Example 5.3.1, the effect on both training and test error is negligible. Thus, these solutions are not of practical*

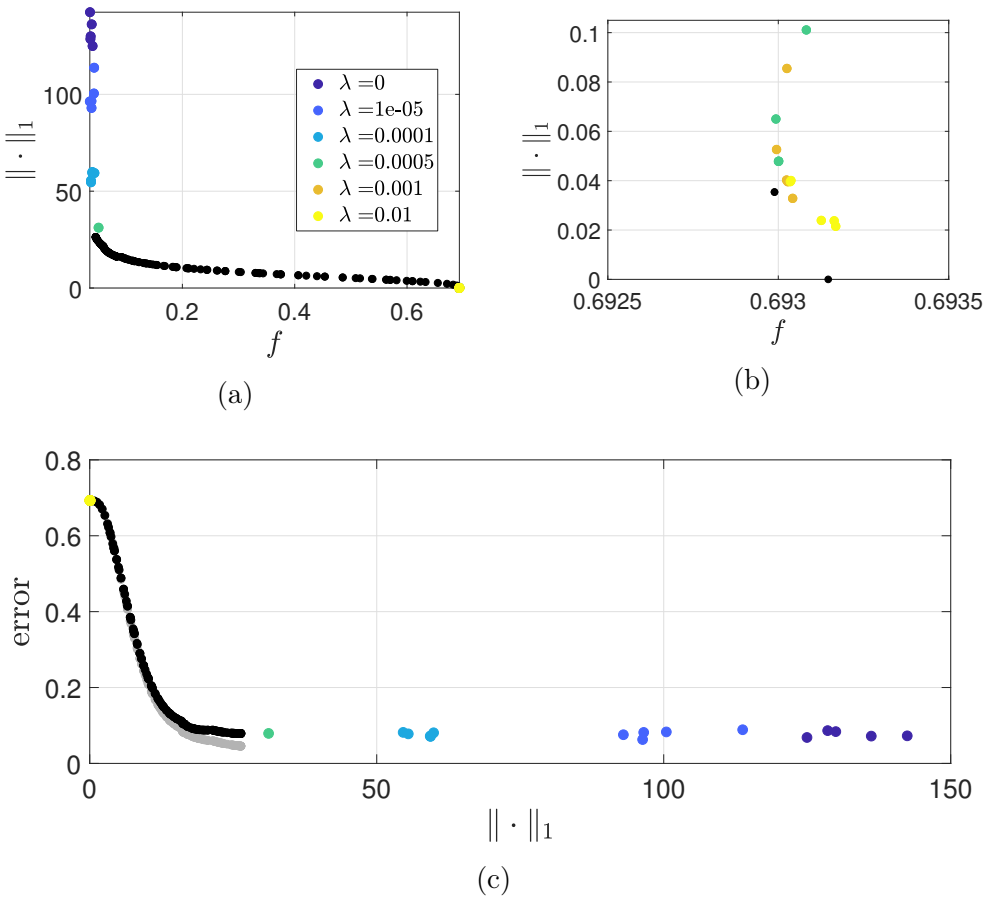


Figure 5.10: Results of Example 5.3.1. (a) Pareto front for the NN training via continuation on the reduced MNIST data set (black). The colored dots show solutions obtained via Adam with different weights for the ℓ_1-term. (b) Zoom-in on the bottom right of (a). (c) Test error (black) and training error (grey) against the ℓ_1-norm for Pareto critical points computed via continuation. The colored dots show the test error of the corresponding Adam solutions.

relevance since the solutions derived by the continuation method are equally good. Furthermore, note that the test error does not increase with a higher value of the ℓ_1-norm, i.e., no overfitting can be observed. This is probably caused by the small size of the model, which already avoids overfitting.

Although the results are promising, modern NNs have millions of parameters that have to be trained. Problems of such a size cannot be solved efficiently with the presented method, even when applying the suggested adaptations. Nevertheless, the proposed method is a very efficient solver for smaller problems and, in the case of NNs, indicates that it is helpful and important to integrate knowledge from multiobjective optimization into NN training methods. Hence, other methods inspired by multiobjective optimization should be developed to improve the training algorithms.

5.4 Generalization to Piecewise Differentiable Regularization Terms

Up to this point, this chapter was concerned with problems where the ℓ_1-norm is used as the regularization term. Beyond that, many other functions are used in practice as regularization terms, for instance, various norms like the already mentioned ℓ_2-norm or the maximum norm. As the ℓ_1-norm, these regularization terms are often nonsmooth. Thus, to generalize the previous results of this chapter, we now consider the problem

$$\min_{x\in\mathbb{R}^n} f(x) + \lambda g(x), \tag{5.31}$$

where $\lambda \in [0, \infty)$, f is at least twice continuously differentiable and $g : \mathbb{R}^n \to \mathbb{R}$ is nonsmooth. Some examples of such nonsmooth regularization terms g are given in Table 5.1. These examples have in common that they are not merely nonsmooth but

Table 5.1: An overview of applications utilizing nonsmooth regularization terms. The optimized variable is always x.

Application	Objective	Regularization term
Sparse regression [HTF09; Tib96]	$\|b - Ax\|_2^2$	sparsity $\|x\|_1$
NN Training [Bis06; GBC16]	loss function	e.g., sparsity $\|x\|_1$ (or $\|x\|_2^2$)
SVM [Bis06; HTF09] (see also Section 5.4.2)	loss function	hinge loss $\sum_{i=1}^N \max(0, 1 - b^i(w^\top a^i + w^0))$ $x = (w, w^0)$; (a^i, b^i): data points
Image denoising [Cha04]	$\|y - x\|_2^2$ y: noisy image	total variation $\sum_{i=1}^{n-1} \|x_{i+1} - x_i\|$
Penalty method [BKM14; NW06] (constrained optimization)	obj. function	feasibility $\sum_{i=1}^p \max(c_i(x), 0)$ c_i: constraints, $i \in \{1, \dots, p\}$

(at least) piecewise twice continuously differentiable. Thus, it seems reasonable to restrict ourselves to such functions, i.e., we assume $g \in PC^2$, where PC^r is formally defined as follows, cf. [Sch12].

Definition 5.4.1. *Let $U \subseteq \mathbb{R}^n$ be open. Let $g : U \to \mathbb{R}$ be continuous and $g_i : U \to \mathbb{R}$, $i \in \{1, \dots, k\}$, be a set of r-times continuously differentiable (or C^r) functions for $r \in \mathbb{N} \cup \{\infty\}$. If $g(x) \in \{g_1(x), \dots, g_k(x)\}$ for all $x \in U$, then g is* piecewise r-times differentiable *(or a PC^r-function). In this case, $\{g_1, \dots, g_k\}$ is called a* set of selection functions *of g. Moreover, we denote by*

$$I(x) := \{i \in \{1, \dots, k\} : g(x) = g_i(x)\}$$

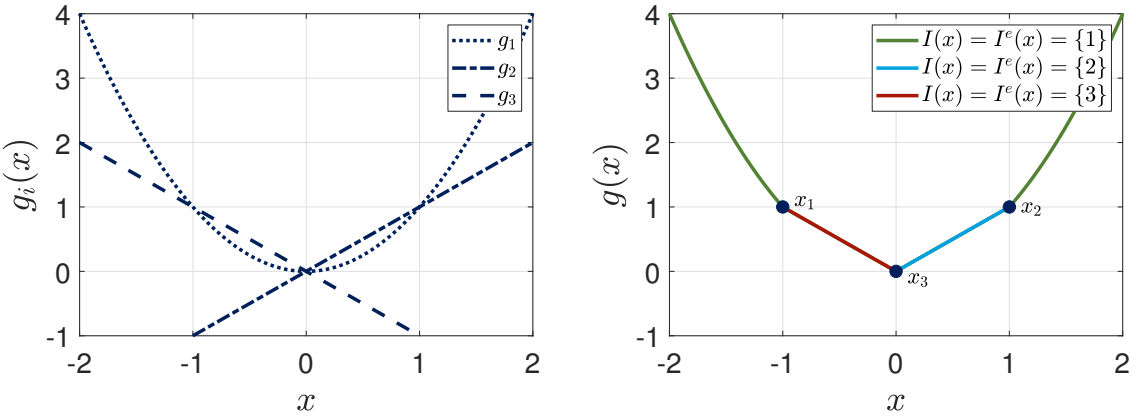


Figure 5.11: Graphs of g_i, $i \in \{1,2,3\}$, and g from Example 5.4.2. The (essentially) active sets of the marked points are given by $I(x_1) = I^e(x_1) = \{1,3\}$, $I(x_2) = I^e(x_2) = \{1,2\}$ and $I^e(x_3) = \{2,3\} \subsetneq I(x_3) = \{1,2,3\}$.

the active set in $x \in U$ *and by*

$$I^e(x) := \{i \in \{1,\dots,k\} : x \in \operatorname{cl}(\operatorname{int}(\{y \in U : g(y) = g_i(y)\}))\}$$

the essentially active set *at* $x \in U$.

Note that the term "active" is used differently than before when we solely considered the ℓ_1-norm. Here, it refers to one of the selection functions g_i (or its index) that has the same value as the function g, i.e., it currently represents g, whereas it referred to the nonzero entries of x (or the corresponding index) before. The following example briefly illustrates the previous definition.

Example 5.4.2. *Consider the function*

$$g : \mathbb{R} \to \mathbb{R}, \quad g(x) = \max(x^2, \|x\|_1) = \max(x^2, x, -x).$$

This function is a PC^∞-function with the set of selection functions $\{g_1, g_2, g_3\}$ where $g_1(x) = x^2$, $g_2(x) = x$ and $g_3(x) = -x$. The graphs and the (essentially) active sets are visualized in Figure 5.11. Note, in particular, that the active set in $x_3 = 0$ is given by $I(0) = \{1,2,3\}$, but the essentially active set is only the subset $I^e(0) = \{2,3\}$.

As shown in the previous example, PC^r-functions are an intuitive way to define piecewise differentiability for a multivariate function. Moreover, the subdifferential of such a function can be easily determined if the essentially active set is known. If $g : U \to \mathbb{R}$ is a PC^r-function with selection functions $\{g_1, ..., g_k\}$, then the subdifferential of g, cf. Definition 2.3.11, is given by

$$\partial g(x) = \mathbf{Conv}(\{\nabla g_i(x) : i \in I^e(x)\}) \quad \forall x \in U. \tag{5.32}$$

Since f and g may be nonconvex, similar to the considerations of the particular case of the ℓ_1-norm, we want to consider the MOP

$$\min_{x \in \mathbb{R}^n} \begin{pmatrix} f(x) \\ g(x) \end{pmatrix} \tag{5.33}$$

instead of (5.31) and analyze the Pareto critical set $\mathcal{P}_c$ of (5.33).

The structure of the regularization path, i.e., all solutions that can be derived by varying $\lambda \geq 0$ in (5.31), was analyzed for some special cases already by other authors. In [Efr+04; OPT00], it was shown that for sparse regression, the regularization path is piecewise linear, and a path-following method was proposed for its computation. Similar results were shown in [Has+04] for support-vector machines. In a more general setting in [RZ07], it was proven that if f is piecewise quadratic and g is piecewise linear, then the regularization path is always piecewise linear. In the case of the exact penalty method in constrained optimization, it was shown in [ZL15] that if the constrained problem is convex (and the equality constraints are affinely linear), then the regularization path is piecewise smooth.

This section aims at deriving the fundamentals for new continuation methods for Problem (5.33), similar to the one presented in the previous sections of this chapter where g was the ℓ_1-norm. To be more precise, conditions for the Pareto critical set to be locally smooth are presented. Parts of the Pareto critical set satisfying these assumptions can then be computed by continuation methods for smooth MOPs.

In Section 5.4.1, a rough overview of the theoretical results is given. Afterwards, in Section 5.4.2, the different types of kinks which occur by violating different assumptions are visualized using an SVM training example. More details, including the proofs of the mentioned results, can be found in [GBP22], to which the author of this thesis made significant contributions.

5.4.1 The Structure of $\mathcal{P}_c$

The basic idea to analyze the structure of $\mathcal{P}_c$ of (5.33), is to decompose $\mathcal{P}_c$ as follows

$$\begin{aligned}\mathcal{P}_c &\overset{(2.13)}{=} \{x \in \mathbb{R}^n : 0 \in \mathbf{Conv}(\{\nabla f(x)\} \cup \partial g(x))\} \\ &= \{x \in \mathbb{R}^n : 0 \in \mathbf{Conv}(\{\nabla f(x)\} \cup \{\nabla g_i(x) : i \in I^e(x)\})\} \\ &= \bigcup_{I \subseteq \{1,\dots,k\}} \mathcal{P}_c^I \cap \Omega^I,\end{aligned} \tag{5.34}$$

where

$$\begin{aligned}\mathcal{P}_c^I &:= \{x \in \mathbb{R}^n : 0 \in \mathbf{Conv}(\{\nabla f(x)\} \cup \{\nabla g_i(x) : i \in I\})\}, \\ \Omega^I &:= \{x \in \mathbb{R}^n : I^e(x) = I\},\end{aligned} \tag{5.35}$$

i.e., $\mathcal{P}_c^I$ is the Pareto critical set of the (smooth) MOP with the objective vector $(f, g_{i_1}, \dots g_{i_{|I|}})^\top$ (for $I = \{i_1, ..., i_{|I|}\}$) and Ω^I is the set of points in $\mathbb{R}^n$ in which precisely the selection functions with an index in I are essentially active. Thus, (5.34) expresses $\mathcal{P}_c$ as the union of Pareto critical sets of smooth MOPs that are intersected with the sets of points with constant essentially active sets. The following example illustrates the decomposition for a problem with the ℓ_1-norm.

Example 5.4.3. *Consider problem (5.33) with* $f : \mathbb{R}^2 \to \mathbb{R}$, $x \mapsto (x_1-2)^2+(x_2-1)^2$,

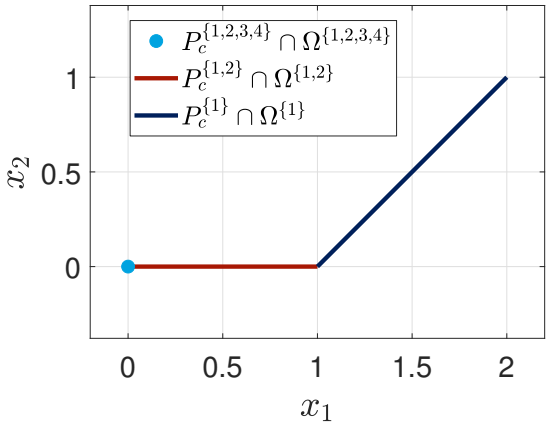


Figure 5.12: Decomposition of $\mathcal{P}_c$ for the MOP in Example 5.4.3 into the sets $\mathcal{P}_c^I \cap \Omega^I$ as in (5.34).

and

$$\begin{aligned}
g_1 &: \mathbb{R}^2 \to \mathbb{R}, \quad x \mapsto x_1 + x_2,\\
g_2 &: \mathbb{R}^2 \to \mathbb{R}, \quad x \mapsto x_1 - x_2,\\
g_3 &: \mathbb{R}^2 \to \mathbb{R}, \quad x \mapsto -x_1 + x_2,\\
g_4 &: \mathbb{R}^2 \to \mathbb{R}, \quad x \mapsto -x_1 - x_2,\\
g \ &: \mathbb{R}^2 \to \mathbb{R}, \quad x \mapsto \|x\|_1 = \max(\{g_1(x), g_2(x), g_3(x), g_4(x)\}).
\end{aligned}$$

It is possible to show that the Pareto critical (and in this case Pareto optimal) set is given by

$$\begin{aligned}
\mathcal{P}_c &= \{(0,0)^\top\} \cup ((0,1] \times \{0\}) \cup \{x \in \mathbb{R}^2 : x_1 \in (1,2], x_2 = x_1 - 1\}\\
&= (\mathcal{P}_c^{\{1,2,3,4\}} \cap \Omega^{\{1,2,3,4\}}) \cup (\mathcal{P}_c^{\{1,2\}} \cap \Omega^{\{1,2\}}) \cup (\mathcal{P}_c^{\{1\}} \cap \Omega^{\{1\}}).
\end{aligned}$$

Figure 5.12 shows the decomposition of $\mathcal{P}_c$ into the sets $\mathcal{P}_c^I \cap \Omega^I$ as in (5.34).

Based on this decomposition, five assumptions can be derived that ensure that the Pareto critical set locally around a critical point $x^0 \in \mathcal{P}_c$ is smooth. These assumptions are summarized in Table 5.2 and are roughly explained throughout this section. The essential idea is to allow for the application of the level set theorem, cf. [Lee12, Theorem 5.12], to a smooth function H whose projected zero level set coincides locally with $\mathcal{P}_c$, precisely as done for the case of the ℓ_1-norm, cf. Equation (5.13).

The first assumption is mainly influenced by the choice of the concrete selection functions to represent g. The choice can usually be adapted such that Assumption A1 is satisfied. More precisely, A1(i) ensures that all selection functions are indeed relevant for the representation of g (in the considered neighborhood U of x^0) since each selection function is at least active in x^0. Note that no additional selection functions can be active in U if it is chosen small enough. The Assumption A1(ii) ensures that it does not matter if we consider the active or the essentially active set in U, which allows for an easier representation of Ω^I. Finally, A1(iii) makes sure

Table 5.2: An overview of the five assumptions required to have a smooth structure of the Pareto critical set $\mathcal{P}_c$ of (5.33) around $x^0 \in \mathcal{P}_c$. For the definition of aff and affdim, see Definitions 5.4.4 and 5.4.5.

Let $x^0 \in \mathcal{P}_c$.	
A1	There is an open nbd. $U \ni x^0$ and a set of sel. fct. $\{g_1, \dots, g_k\}$ of $g\|_U$ with *(i)* $I(x^0) = \{1, \dots, k\}$, *(ii)* $I^e(x) = I(x) \quad \forall x \in U$, *(iii)* $\operatorname{affdim}(\operatorname{aff}(\{\nabla g_i(x) : i \in \{1, \dots, k\}\}))$ $= \operatorname{affdim}(\operatorname{aff}(\{\nabla g_i(x^0) : i \in \{1, \dots, k\}\})) \ \forall x \in U$.
A2	It holds $\nabla f(x^0) \notin \operatorname{aff}(\partial g(x^0))$.
A3	Let $\{g_1, \dots, g_k\}$ be a set of selection functions of g. It exists $\{i_1, \dots, i_r\} \subseteq \{1, \dots, k\}$ and $\alpha^0 \in \mathbb{R}$, $\beta^0 \in \mathbb{R}^r$ with $\alpha^0 + \sum_{j=1}^r \beta_j^0 = 1$ such that *(i)* $r = \operatorname{affdim}(\operatorname{aff}(\{\nabla f(x^0)\} \cup \partial g(x^0)))$, *(ii)* $\{\nabla f(x^0)\} \cup \{\nabla g_i(x^0) : i \in \{i_1, \dots, i_r\}\}$ aff. ind., *(iii)* $\alpha^0 \nabla f(x^0) + \sum_{j=1}^r \beta_j^0 \nabla g_{i_j}(x^0) = 0$, *(iv)* $\alpha^0 > 0$, $(\beta^0)_j > 0 \ \forall j \in \{1, ..., r\}$.
A4	Assume that A1, A2 and A3 hold and let H be defined as in Lemma 5.4.8. *(a)* $\operatorname{rk}(DH(x, \alpha, \beta)) = n + r \quad \forall (x, \alpha, \beta) \in H^{-1}(0)$ **or** *(b)* $\operatorname{rk}(DH(x, \alpha, \beta))$ is constant $\quad \forall (x, \alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}^{>0} \times (\mathbb{R}^{>0})^r$.
A5	Let $\{g_1, \dots, g_k\}$ be a set of selection functions of g. There is an open neighborhood $U \ni x^0$ with $I^e(x) = I^e(x^0) \quad \forall x \in \mathcal{P}_c \cap U$.

that the representation of $\partial g(x^0)$ via the gradients of our selection functions has the same *affine dimension* on the whole set U. For completeness, a formal definition of the affine dimension is given here. To this end, we first introduce the terms *affine hull* and *affine space*, see, for instance, [Gal11].

Definition 5.4.4.

(a) Let $k \in \mathbb{N}$ and $a^i \in \mathbb{R}^n$, $i \in \{1, \dots, k\}$. Let $\lambda \in \mathbb{R}^k$ with $\sum_{i=1}^k \lambda_i = 1$. Then $\sum_{i=1}^k \lambda_i a^i$ is an affine combination of $\{a^1, \dots, a^k\}$.

(b) Let $E \subseteq \mathbb{R}^n$. Then $\operatorname{aff}(E)$ is the set of all affine combinations of elements of E, called the affine hull of E. *Formally,*

$$\operatorname{aff}(E) := \left\{ \sum_{i=1}^k \lambda_i a^i : k \in \mathbb{N}, a^i \in E, \lambda_i \in \mathbb{R}, i \in \{1, \dots, k\}, \sum_{i=1}^k \lambda_i = 1 \right\}.$$

(c) Let $E \subseteq \mathbb{R}^n$. If $\operatorname{aff}(E) = E$, then E is called an affine space.

Affine spaces can be thought of as linear spaces that do not pass through the origin but have been shifted by a certain value. More precisely, if E is an affine space and $a' \in E$, then the set

$$E - a' = \{a - a' : a \in E\} =: V \tag{5.36}$$

is a linear subspace of $\mathbb{R}^n$. Note that V does not depend on the choice of a'. This allows for the definition of *affine bases* (also known as *affine frames*) and *affine independence*. Moreover, we can assign a dimension to an affine space.

Definition 5.4.5. *Let $k \in \mathbb{N}$ and let E be an affine space with corresponding linear subspace V (cf. (5.36)). Let $\{a^1, \dots, a^k\} \subseteq E$.*

(a) Then $\{a^1, \dots, a^k\}$ is called affinely independent *if $\{a^i - a^j : i \in \{1, \dots, k\} \setminus \{j\}\}$ is linearly independent for some $j \in \{1, \dots, k\}$.*

(b) Then $\{a^1, \dots, a^k\}$ is called an affine basis of E *if $\{a^i - a^j : i \in \{1, \dots, k\} \setminus \{j\}\}$ is a basis of V for some $j \in \{1, \dots, k\}$.*

(c) The (affine) dimension *of E is the dimension of V, denoted by* $\operatorname{affdim}(E)$.

For our analysis, we consider the affine hull of the subdifferential of g, later also combined with the gradient of f. In this context, the affine dimension may be seen as a more general concept than the rank of the Jacobian of a smooth function and takes into account, for instance, that it makes no difference for the "regularity" of the function g if there are two identical functions in the set of selection functions. Thus, roughly speaking, A1(iii) can be interpreted as a constant rank condition in the nonsmooth case.

Assumption A2 ensures that the gradient of f is not in the affine hull of the subdifferential of g. It is possible to show that, in this case, the coefficient of the gradient of f of the convex combination in (2.14) is unique. Although a Pareto critical point x^0 with $\nabla f(x^0) \in \operatorname{aff}(\partial g(x^0))$ may not necessarily cause nonsmoothness of $\mathcal{P}_c$, we need to make this assumption to avoid an irregularity in the augmented space of $\mathcal{P}_c$ and the corresponding KKT multipliers, cf. Lemma 5.4.8. Note that in the case of g being the ℓ_1-norm, Assumption A2 always holds and the KKT multipliers are unique, cf. Remark 5.1.3. Moreover, according to the discussion in Remark 5.1.5, the non-uniqueness of the multipliers indicates a kink in the Pareto front.

Assumption A3 is the first assumption that is directly related to kinks in the Pareto critical set. Essentially, it ensures that there is a choice of r selection functions such that $\mathbf{Conv}(\{\nabla f(x^0)\} \cup \partial g(x^0))$ is spanned by $\nabla g_{i_1}(x^0), \dots, g_{i_r}(x^0)$ and $\nabla f(x^0)$, and, at the same time, there is a vanishing convex combination of these gradients. We can give a necessary condition for Assumption A3 based on the *relative interior* of $\mathbf{Conv}(\{\nabla f(x^0)\} \cup \partial g(x^0))$. In particular, this necessary condition, given in the following lemma, is independent of the choice of selection functions.

Definition 5.4.6. *Let $A \subseteq \mathbb{R}^n$ and let $\operatorname{aff}(A)$ be endowed with the subspace topology of $\mathbb{R}^n$. Then the* relative interior of A, *denoted by* $\operatorname{ri}(A)$, *is the interior of A in*

$\operatorname{aff}(A)$, *i.e.*,

$$\operatorname{ri}(A) := \{x \in A : \exists U \subseteq \mathbb{R}^n \text{ open with } x \in U \text{ and } U \cap \operatorname{aff}(A) \subseteq A\}.$$

Lemma 5.4.7. *Let $x^0 \in \mathcal{P}_c$. If there is a set of selection functions such that Assumption A3 holds, then*

$$0 \in \operatorname{ri}(\mathbf{Conv}(\{\nabla f(x^0)\} \cup \partial g(x^0))).$$

Proof. See [GBP22, Appendix A.2]. □

Using the Assumptions A1, A2 and A3, it is possible to show that $\mathcal{P}_c^I \cap \Omega^I$ is the projection of a level set from a higher dimensional space onto the variable space $\mathbb{R}^n$.

Lemma 5.4.8. *Let $x^0 \in \mathcal{P}_c$. Let $U \subseteq \mathbb{R}^n$ be an open neighborhood of x^0 and let $\{g_1, \dots, g_k\}$ be a set of selection functions of $g|_U$ satisfying Assumption A1. In addition, assume that Assumption A2 holds and that there exists an index set $\{i_1, \dots, i_r\} \subseteq \{1, \dots, k\}$ such that Assumption A3 is satisfied. Then there is an open neighborhood $U' \subseteq U$ of x^0 such that*

$$\mathcal{P}_c^{\{1,\dots,k\}} \cap \Omega^{\{1,\dots,k\}} \cap U' = \operatorname{pr}_x(H^{-1}(0)) \cap U', \tag{5.37}$$

where $\operatorname{pr}_x : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^r \to \mathbb{R}^n$ is the projection onto the first n components and

$$H : \mathbb{R}^n \times \mathbb{R}^{>0} \times (\mathbb{R}^{>0})^r \to \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^{r-1}, (x, \alpha, \beta) \mapsto \begin{pmatrix} \alpha \nabla f(x) + \sum_{j=1}^r \beta_j \nabla g_{i_j}(x) \\ \alpha + \sum_{j=1}^r \beta_j - 1 \\ (g_{i_j}(x) - g_{i_1}(x))_{j \in \{2,\dots,r\}} \end{pmatrix}.$$

Proof. See [GBP22, Proof of Lemma 6]. □

So far, we have only exploited the fact that f is continuously differentiable and g is PC^1. This implies that the map H in the previous lemma is at least continuous. If H is actually continuously differentiable, the level set theorem can be used to analyze the structure of its level sets on the right-hand side of (5.37). This is done in the following theorem, which can be seen as an analogue to Theorem 2.3.17.

Theorem 5.4.9. *In the setting of Lemma 5.4.8 it holds:*

(a) If $DH(x, \alpha, \beta)$ has full rank for all $(x, \alpha, \beta) \in H^{-1}(0)$, then $H^{-1}(0)$ is a 1-dimensional submanifold of $\mathbb{R}^n \times \mathbb{R}^{>0} \times (\mathbb{R}^{>0})^r$.

(b) If $DH(x, \alpha, \beta)$ has constant rank $m \in \mathbb{N}$ for all $(x, \alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}^{>0} \times (\mathbb{R}^{>0})^r$, then $H^{-1}(0)$ is an $(n+r+1-m)$-dimensional submanifold of $\mathbb{R}^n \times \mathbb{R}^{>0} \times (\mathbb{R}^{>0})^r$.

In both cases, the tangent space of $H^{-1}(0)$ is given by

$$T_{(x,\alpha,\beta)}(H^{-1}(0)) = \ker(DH(x, \alpha, \beta)). \tag{5.38}$$

Proof. See [GBP22, Proof of Theorem 2]. The proof is essentially based on the level set theorem, cf. [Lee12]. □

The additional requirements of Theorem 5.4.9 are summarized in Assumption A4. Note that violating Assumption A4 is not necessarily caused by the nonsmoothness of g and can also happen for smooth objective functions, see, for instance, Example 1 in [GPD19].

Theorem 5.4.9 considers the structure of $\mathcal{P}_c^I \cap \Omega^I$, i.e., the structure of parts where the (essentially) active set does not change. The only nonsmooth points in $\mathcal{P}_c$, not addressed so far, may arise by taking their union, i.e., at points where the set of (essentially) active selection functions changes. These points are excluded by Assumption A5. Unfortunately, in contrast to Assumptions A1 to A4, Assumptions A5 is only an a posteriori condition, i.e., we already have to know $\mathcal{P}_c$ around x^0 to be able to check if Assumption A5 holds. Although the considered examples suggest that typically Assumption A3 or A2 do not hold when Assumption A5 is violated, so far, it could not be proven that Assumption A5 may be already covered by the other four assumptions.

In summary, the main result of this section is that if we have a Pareto critical point x^0 for which Assumptions A1 to A5 hold, the Pareto critical set $\mathcal{P}_c$ is the projection of a certain manifold around x^0 by Theorem 5.4.9. This allows us to formulate an abstract version of a continuation method for the computation of $\mathcal{P}_c$, similar to Algorithm 4, which is presented in Algorithm 6.

Algorithm 6 Rough draft of the continuation method.

Input: Step size $h > 0$, initial point $x_{\mathsf{start}} \in \mathcal{P}_c$ and $I \subseteq \{1, ..., k\}$ such that $x_{\mathsf{start}} \in \mathcal{P}_c^I \cap \Omega^I$

1: Initialize $i = 0$ and $x^0 = x_{\mathsf{start}}$.
2: Compute the projected tangent space of $\mathcal{P}_c^I \cap \Omega^I$ in x^i via Lemma 5.4.8 and Theorem 5.4.9 and choose a tangent vector v in the current direction of continuation. (Predictor step)
3: Compute a point x^{i+1} in $\mathcal{P}_c^I \cap \Omega^I$ close to $x^i + hv$. (Corrector step)
4: **if** the end of $\mathcal{P}_c^I \cap \Omega^I$ is detected **then**
5: Compute the endpoint $\bar{x}$ of $\mathcal{P}_c^I \cap \Omega^I$.
6: **for** all $I' \subseteq I^e(\bar{x})$, $I' \neq I$, $\mathcal{P}_c^{I'} \cap \Omega^{I'} \neq \emptyset$ **do**
7: Restart this method with $I = I'$ and some $x^1 \in \mathcal{P}_c^{I'} \cap \Omega^{I'}$ close to $\bar{x}$.
8: **end for**
9: **else**
10: Set $i = i + 1$ and go to step 2.
11: **end if**

Obviously, this algorithm cannot be implemented directly, and, depending on the concrete problem to solve, strategies to realize steps 3 to 6 have to be further investigated. Some considerations of practical issues are summarized in the following remark.

Remark 5.4.10. *(a) For the development of continuation methods, it is crucial to be able to detect nonsmooth points during the computation of $\mathcal{P}_c$. If the different sets $\mathcal{P}_c \cap \Omega^I$ are computed separately, then typically (but not necessarily), the nonsmooth points of the path are the endpoints of these sets (in case the path is "1-dimensional"). Thus, since continuation methods compute a pointwise approximation of the path, these endpoints roughly appear as points where the method fails to continue with the currently active set $I \subseteq \{1, ..., k\}$. To find the exact nonsmooth point, one could try to find the closest point where one of the Assumptions A1 to A5 is violated. While it is unclear how this can be done numerically in the general setting presented here, it is easier in specific applications, where more structure can be exploited, as done for the case of ℓ_1-regularization in the previous part of this chapter.*

(b) If Assumption A5 is violated in $x^0 \in \mathcal{P}_c$, then there are Pareto critical points arbitrarily close to x^0 with a different (essentially) active set $I' \neq I^e(x^0)$. Within a continuation method, it may be of interest to find I' to compute the direction in which $\mathcal{P}_c$ continues once the nonsmoothness in x^0 is detected. To this end, let $\{g_1, \dots, g_k\}$ be the set of selection functions that are all essentially active in x^0. While it is not possible to determine I' solely from the set $\mathbf{Conv}(\{\nabla f(x^0)\} \cup \partial g(x^0)) = \mathbf{Conv}(\{\nabla f(x^0)\} \cup \{\nabla g_1(x^0), \dots, \nabla g_k(x^0)\})$, we can at least determine all potential candidates for I' by finding all subsets $\{i_1, \dots, i_m\} \subseteq \{1, \dots, k\}$ with

$$0 \in \mathbf{Conv}(\{\nabla f(x^0)\} \cup \mathbf{Conv}(\{\nabla g_{i_1}(x^0), \dots, \nabla g_{i_m}(x^0)\})).$$

5.4.2 An Example - Support Vector Machines

Assume that a data set $D = \{(a^i, b^i) : a^i \in \mathbb{R}^l, b^i \in \{-1, 1\}, i \in \{1, \dots, N\}\}$ is given. Then, the goal of an SVM is to find parameters $w \in \mathbb{R}^l$ and $w^0 \in \mathbb{R}$ such that

$$\operatorname{sgn}(w^\top a^i + w^0) = b^i \quad \forall i \in \{1, \dots, N\}.$$

In other words, the goal is to find a hyperplane $\{d \in \mathbb{R}^l : w^\top d + w^0 = 0\}$ such that all a^i with $b^i = 1$ lie on one side and all a^i with $b^i = -1$ lie on the other side of the hyperplane, cf. [Bis06].

Since such a hyperplane is usually not unique, a second criterion is introduced. To get a robust classification, the hyperplane that has the largest distance to the given data points a^i, the so-called *margin*, is computed. As in the case of sparse optimization with respect to the ℓ_1-norm, this is typically done via regularization. More precisely, we consider Problem (5.31) with

$$f : \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}, \quad x = (w, w^0) \mapsto \frac{1}{2}\|w\|_2^2 \quad \text{and}$$

$$g : \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}, \quad x = (w, w^0) \mapsto \sum_{i=1}^{N} \max\{0, 1 - b^i(w^\top a^i + w^0)\}.$$

Often g is referred to as the *hinge loss*, cf. Table 5.1. Note that in the literature, the roles of f and g are typically reversed.

In theory, the most favorable hyperplane would be one with $g(w, w^0) = 0$ and $f(w, w^0)$ as small as possible. But in practice, when working with large and noisy data sets, a perfect separation might not be possible. Furthermore, an imperfect separation, where only a few points violate the separation, may be more desirable to get a more robust classifier that is not overfitted to the training data. The balance between the margin and the quality of the separation can be controlled via the parameter λ in (5.31). Since both objectives are convex, the regularized problem has the same solution (excluding the global minimum of g) as the MOP (5.33). The solution set of this problem class was already considered in earlier works. In [Has+04], it was shown that the set is 1-dimensional and piecewise linear up to certain degenerate points, and a path-following method was proposed that exploits this structure. It was conjectured (without proof) that the existence of these degenerate points is related to certain properties of the data points (a^i, b^i), like having duplicates of the same point in the data set or having multiple points with the same margin. In [OSY10], these degeneracies were analyzed further, and a modified path-following method was proposed, specifically taking degenerate data sets into account. Other methods for degenerate data sets were proposed in [Dai+13; SAG16; Wan+19]. In the following, we analyze how these degeneracies are related to the nonsmooth points we characterized before.

Obviously, f is twice continuously differentiable and g is PC^2 with selection functions

$$\left\{ x = (w, w^0) \mapsto \sum_{i \in I} 1 - b^i(w^\top a^i + w^0) : I \subseteq \{1, \dots, N\} \right\}.$$

Since f is quadratic and all selection functions are linear, it is possible to show that $\mathcal{P}_c$ is piecewise linear (but not necessarily 1-dimensional) up to points violating the Assumptions A1 to A5, cf. [GBP22, Remark A.3]. Due to the properties of g, the Assumption A1 always holds, cf. [GBP22, Remark A.5].

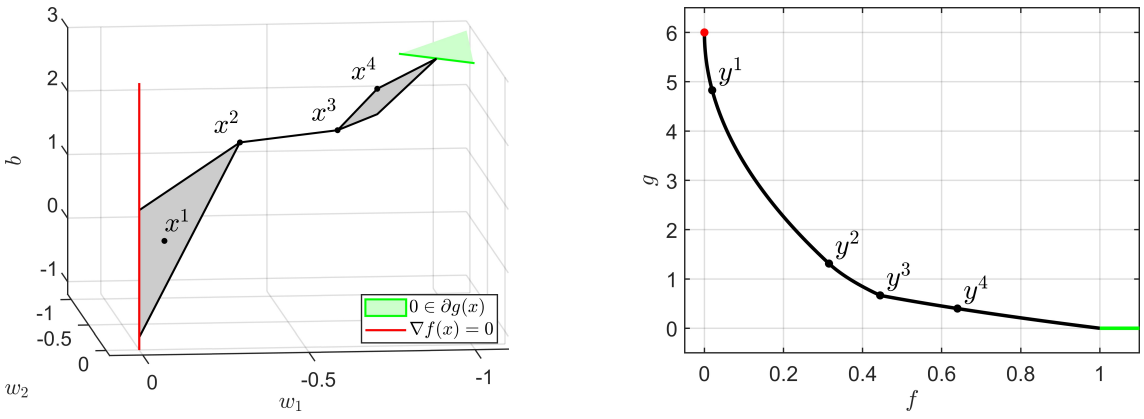
In the following, we consider the remaining Assumptions A2 to A5 for a concrete example. To be more precise, we consider Example 1 from [OSY10], which was specifically constructed to have a degenerated Pareto critical set (or regularization path).

Example 5.4.11. *Consider the data set*

$$D = \big\{((0.7, 0.3)^\top, 1), ((0.5, 0.5)^\top, 1), ((2, 2)^\top, -1), \\ ((1, 3)^\top, -1), ((0.75, 0.75)^\top, 1), ((1.75, 1.75)^\top, -1)\big\}.$$

The Pareto critical set for the MOP induced by this data set can be computed analytically and is shown in Figure 5.13a. In the following, we analyze the points x^1, x^2, x^3 and x^4 highlighted in Figure 5.13a.

There are two 2-dimensional parts of the Pareto critical set, and the point x^1 lies in one of them. It is possible to show that g is smooth around x^1 and easy to verify



(a) (b)

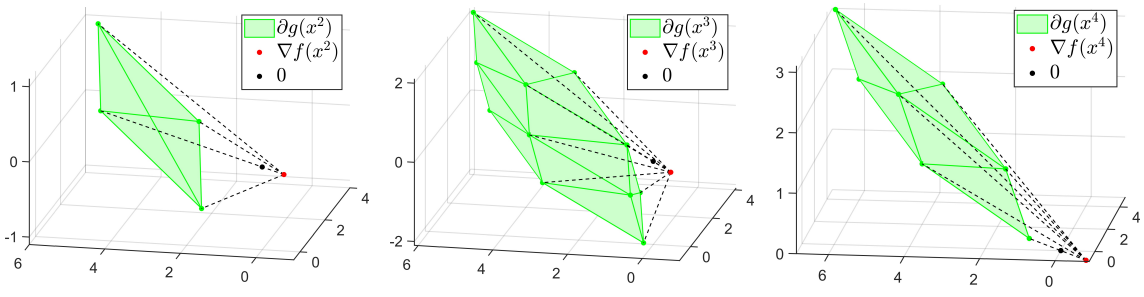
Figure 5.13: $\mathcal{P}_c$ for the MOP induced by the data set in Example 5.4.11 and the points $x^1 = \frac{1}{372}(-35, -65, 137)^\top$, $x^2 = \frac{1}{93}(-35, -65, 137)^\top$, $x^3 = \frac{1}{3}(-2, -2, 5)^\top$, and $x^4 = \frac{1}{5}(-4, -4, 11)^\top$. (b) Image of $\mathcal{P}_c$ with $y^i = (f(x^i), g(x^i))^\top$, $i \in \{1, \dots, 4\}$, and the same coloring as in (a).

that Assumptions A2, A3 and A5 are satisfied. Concerning Assumption A4, it holds $r = \operatorname{affdim}(\operatorname{aff}(\{\nabla f(x^1)\} \cup \partial g(x^1))) = 1$ *and*

$$DH(x, \alpha, \beta) = \begin{pmatrix} 2\alpha & 0 & 0 & -\frac{35}{372} & \frac{14}{5} \\ 0 & 2\alpha & 0 & -\frac{65}{372} & \frac{26}{5} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

with $\operatorname{rk}(DH(x, \alpha, \beta)) = 3$ *for all* $(x, \alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}^{>0} \times \mathbb{R}^{>0}$. *Thus, A4(b) holds which by Theorem 5.4.9 implies that* $\mathcal{P}_c$ *is the projection of an* $n+r+1-m = 3+1+1-3 = 2$ *dimensional manifold around* x^1, *as expected.*

In x^2, *we observe a kink of the Pareto critical set. The subdifferential of* g *in* x^2 *can be computed analytically and is shown in Figure 5.14a.*



(a) (b) (c)

Figure 5.14: Gradient of f, subdifferential of g and the (relative) boundary of the convex hull (dashed) in x^2, x^3, and x^4 in Example 5.4.11.

In this case, we have $\operatorname{affdim}(\operatorname{aff}(\partial g(x^2))) = 2$ *and* $\nabla f(x^2) \notin \operatorname{aff}(\partial g(x^2))$. *Hence, Assumption A2 is satisfied. But, according to Lemma 5.4.7, Assumption A3 has*

to be violated since zero lies on the relative boundary of $\mathbf{Conv}(\{\nabla f(x^2)\} \cup \partial g(x^2))$. *Furthermore, the active set changes in* x^2, *i.e., Assumption A5 is violated, as well.*

A different type of kink occurs at point x^3. *The corresponding subdifferential of* g *is shown in Figure 5.14b. As for* x^2, *Assumptions A3 and A5 are violated in* x^3. *But in contrast to* x^2 *we have* $\operatorname{affdim}(\operatorname{aff}(\partial g(x^2))) = 3$, *so* $\nabla f(x^2) \in \operatorname{aff}(\partial g(x^2)) = \mathbb{R}^3$ *trivially holds and Assumption A2 is violated additionally. This results in a kink in the Pareto front in the image of* x^3 *under the objective vector* (f, g), *as can be seen in Figure 5.13b.*

Finally, x^4 *marks a corner of one of the* 2*-dimensional parts of the Pareto critical set, and the corresponding subdifferential is shown in Figure 5.14c. As for* x^3, *Assumptions A2, A3 and A5 are violated in* x^4. *But unlike* x^3, *when we consider the image of* x^4 *in Figure 5.13b, we see that there is no kink in* y^4. *This suggests that the KKT multiplier of* f *may be unique even though Assumption A2 is violated.*

6 | Conclusion and Future Work

Motivated by the rapid development in the field of machine learning and the importance of controlling complex nonlinear dynamical systems, this thesis dealt with the use of data-based methods in the context of MPC and analyzed regularized optimization problems which are often encountered in the training of machine learning models. First, it was shown, using an example of a flow control problem, that it may be sufficient to consider only measurement data of sensors and a specific neural network architecture as a surrogate model to solve complex control problems via MPC. Subsequently, it was proven that multiple autonomous models can be used to build a surrogate model for a controlled system which may reduce the amount of data needed to train accurate models. In the second part of the thesis, regularized optimization problems, which often occur in the training of machine learning models to avoid overfitting, were studied from a multiobjective optimization perspective. This allows for a better understanding of the trade-off between the objectives and for developing new solution methods.

To conclude this thesis, this chapter summarizes the main contributions and provides an outlook on future research questions.

DeepMPC for Flow Control

In Chapter 3, an approach that embeds a deep recurrent neural network as a surrogate model in the MPC framework was presented and applied to the fluidic pinball. This is an academic flow control problem specifically designed to test new control algorithms in the area of fluid dynamics, as it behaves chaotically in specific regimes [Den+18; Den+19]. The results, presented in Section 3.2, are promising and proved (for this specific example) that it is possible to (a) learn the relevant dynamics only based on sensor data and (b) use a deep neural network as a surrogate model to achieve robust control of the system, even if the underlying system is chaotic. Furthermore, it turned out to be helpful to double the training data by exploiting the system's symmetry. In the last step, online learning was considered for the case of $Re = 100$, where the system does not behave chaotically but periodically. The resulting model also performed very well. Nevertheless, many open questions were raised, as discussed in Section 3.3.

Part of the discussion in Section 3.3 dealt with structural issues of the studied experiment, such as the concrete architecture of the NN or the initialization used, but also with the applicability to noisy measurement data, which would be relevant,

especially for more challenging real-world examples. By addressing these issues, a significant improvement can likely be achieved for the concrete problem of the fluidic pinball and flow control problems in general.

The question of the applicability of the presented or similar approaches based on machine learning models in real physical systems raises further problems. In the presented experiment, the "real" system was a computer simulation. In practice, the data must be taken from a real physical system, which is much more complex. On the one hand, it can be expensive to collect data. This aspect was partially addressed in Chapter 4. On the other hand, it may be inadmissible to collect data in every region of the state space, e.g., an accident of an autonomous vehicle or a setting on a power plant that may lead to its damage should not be tested in a real physical experiment. Here, the combination of simulations and physical experiments can be helpful. However, a simulation is, of course, based on a model and is, therefore, less accurate, which has to be taken into account. Another possibility to reduce the data needed to train an accurate model could be by incorporating active learning strategies, see, for instance, [TBM21]. Moreover, in terms of the real-world applicability of machine learning systems for control problems, the need for further development of online and transfer learning, although already known in the community, is an important insight from the experiments. In Section 3.2, online learning was only performed for $Re = 100$. It would be of interest to develop more robust online learning algorithms that are capable of improving the results for more complex or even chaotic systems. Transfer learning means that based on a model for one task, a model for another (related) task is learned. For instance, we might have trained a model for the system identification of the fluidic pinball for $Re = 100$ and would like to train a model for $Re = 120$. To this end, continuation methods may be used, similar to the ones from multiobjective optimization. If the system does not undergo a bifurcation, this likely works well. Moreover, the system identification at different Reynolds numbers can be seen as a multitask learning problem if we want to train a single model that can predict the system dynamics for different Reynolds numbers. The problem of multitask learning can also be addressed as a multiobjective optimization problem, cf. [SK18].

A more general point, also raised in Section 3.3, is the idea of combining existing system knowledge with machine learning techniques, called physics-informed machine learning [Kar+21]. This area was also briefly introduced in Section 2.1.3 and is a very active area of research.

Utilizing Autonomous Models for Model Predictive Control

In Chapter 4, a framework for the use of (data-based) surrogate models in MPC was presented, which is based on the idea of using multiple autonomous models that do not receive the control $u \in U$ as an input but only the (observed) system state $z \in Z$. To this end, a finite subset $V = \{u^1, \dots, u^m\}$ of the control set U is introduced, and for each control input u^i, $i \in \{1, \dots, m\}$, a separate surrogate model is trained that represents the system dynamics for the specific input. A similar idea was followed in [PK19; POR20] for surrogate models approximating the Koopman operator or generator. Here, we allowed for arbitrary surrogate models which may

act on measurement data given by an observable or on the full state. However, knowledge of an error bound on the approximation error was assumed to derive error bounds for the open-loop problem in Section 4.2. In Section 4.3, we demonstrated good control performance of the approach on a variety of dynamical systems using different control inputs, observations, and surrogate modeling techniques. Moreover, we discussed whether the presented approach is advantageous in the sense that it requires less data than training a full model with the same accuracy. To this end, additional numerical experiments were performed and presented in Section 4.4.

As mentioned before, the derivation of the error bounds for the open-loop problem assumes that the model error is known, cf. Assumption 4.2.7. Therefore, it would be necessary to derive (improved) error bounds for data-based methods like neural networks. Moreover, the derived error bounds are not tight and only concern the open-loop problem. Therefore, for error bounds that are useful in practice, the closed-loop problem should be considered, including stability and robustness issues. The stability and robustness issues are not specific to the presented framework but are generally important when machine learning is used to solve control problems [BS15; MRW21]. Furthermore, the issue of how much data is needed has only been addressed in some numerical experiments. A more detailed mathematical analysis needs to be carried out here. Another more practical issue is the choice of V, which is essential for good control performance, cf. Remark 4.2.2. Moreover, an additional issue arises if we consider V to be the set of control bounds, e.g., if the control space is given by $U = [-1, 1]^{n_u}$ and we choose V as the set consisting of the corners, i.e., $V = \{(-1, \dots, -1)^\top, (-1, \dots, -1, 1)^\top, \dots, (1, \dots, 1)^\top\} \subseteq \{-1, 1\}^{n_u}$. In this case, the number of autonomous systems to be learned increases exponentially, i.e., we have 2^{n_u} systems to approximate. For affine control systems, this problem can be avoided by allowing for extrapolation. This way, only one autonomous system corresponding to $u = 0$ and n_u autonomous systems corresponding to the Euclidean basis vectors for the control space ($u^1 = (1, 0, \dots, 0)^\top$, $u^2 = (0, 1, 0, \dots, 0)^\top$, etc.) have to be trained, resulting in the training of $n_u + 1$ systems. Thus, the complexity (in terms of the number of autonomous systems) is merely linear in the control dimension. This has already been considered in [POR20] for approximating the Koopman generator as a surrogate model. Finally, bang-bang control, resulting from the SUR algorithm, is generally not admissible for practical applications. Therefore, it may be of interest to pursue other interpolation approaches and include constraints on the smoothness of the control function in the control cost.

Treating ℓ_1-regularized Problems via Multiobjective Continuation

In the last part of the thesis, regularization problems were considered. These arise, for example, when machine learning models are trained and a regularization term is added to avoid overfitting, cf. Sections 2.2.1 and 2.2.2. Here, instead of using the sum of the loss function and the regularization term, the idea was to consider the biobjective MOP that has the loss function as the first objective and the regularization term as the second objective. More precisely, in Chapter 5, the biobjective MOP was considered where the first objective is a twice continuously differentiable and the second objective is the ℓ_1-norm, cf. (MOP-ℓ_1). From the multiobjective op-

timization point of view, the standard approach of adding the regularization term to the loss function is equivalent to the weighted sum method. However, this method cannot be used to compute all Pareto optimal solutions, so the idea of a continuation method was pursued. Since the ℓ_1-norm is not differentiable, existing continuation methods, cf. [Hil01; SDD05], cannot be used. Therefore, in Section 5.1, a novel continuation method was developed for this particular problem. The optimality conditions, derived in Section 5.1.1, show that the Pareto critical set $\mathcal{P}_c$ is piecewise smooth and thus allows continuation methods from the smooth case to be applied almost everywhere. Furthermore, at points where $\mathcal{P}_c$ is nonsmooth, the derived results allow for exploiting the structure of the set to proceed efficiently with the continuation method, cf. Section 5.1.4. In Section 5.2.3, the method was used to train a (small) NN. This numerical experiment showed that relevant solutions can be missed by the traditional penalty approach but can be computed by the continuation method. Finally, two extensions were discussed. First, Section 5.3 discussed how the method can be scaled to larger dimensions and the special problems that arise when dealing with NNs. Second, more general regularization problems were considered in Section 5.4, where the ℓ_1-norm was replaced by a piecewise twice continuously differentiable function. For these more general problems, the structure of the Pareto critical set was analyzed by deriving conditions that ensure that the Pareto critical set is locally smooth (A1 to A5). The results are mainly based on a partition of the Pareto critical set $\mathcal{P}_c$ into parts with constant essentially active sets, which allowed us to apply techniques from differential geometry to obtain structural results. In Section 5.4.2, the kinks occurring when (at least) one of the assumptions is violated were visualized by an example of SVMs.

For future work, based on the results presented in Section 5.4, continuation methods for MOPs of the form (5.33) can be developed, supplementing the continuation methods from Section 5.1 and [Hil01; RZ07; ZL15]. These may address other applications, such as image denoising via total variation. Although the presented results have laid the foundations of such methods, in that nonsmooth points can be characterized, and the tangent space can be computed on smooth sections, there is still a large gap to a concrete implementation. In particular, it is unclear how to compute a corrector step, as in Section 5.1.3, or how to detect kinks and find a new direction for the following predictor step, analogous to Section 5.1.4.

Concerning the concrete continuation method for the Problem (MOP-ℓ_1), it should be improved to scale to a higher dimension. This would be of particular interest for the training of NNs. A first step was done in Section 5.3 by approximating the (reduced) Hessian matrix and its inverse. Furthermore, especially in the context of NN training, more structural questions arise. First of all, the symmetries in NNs have to be understood and handled better, as these cause the splitting of the Pareto critical set and singular Hessian matrices. Moreover, local minima, which occur frequently in the loss function of NNs, usually cause the Pareto critical set to consist of multiple connected components. Since it is expensive to compute new starting points on other components, it may be an option to compute only a few starting points with different methods, for instance, by multiobjective stochastic gradient descent methods [LV21; MPD18], and start the continuation method from

there to derive more sparse and robust solutions.

To further generalize the presented results on the structure of Pareto critical sets, an arbitrary number of (nonsmooth) objectives may be considered, and constraints can be added. Likely, most of the results presented in Section 5.4 hold with minor adjustments if the first objective f would be merely piecewise differentiable. Furthermore, adding more objectives will lead to similar results but with a higher dimensional regularization path. Moreover, the continuation method for (MOP-ℓ_1) can probably be directly extended to an arbitrary number of additional smooth objectives similar to the Pareto explorer [Sch+19], which can handle MOPs with more than two objectives and equality constraints. This is also of practical interest. For instance, for the *elastic net regularization*, two regularization terms, the ℓ_1-norm and the squared ℓ_2-norm, are considered at the same time [ZH05].

To address the problem of NN training with regularization terms, other methods from multiobjective optimization can be utilized. To improve the efficiency, combining them with existing techniques from NN training, as done in [Rei+22], can be helpful. Besides the regularization problems considered here, utilizing techniques from multiobjective optimization in the area of machine learning is promising in general. For instance, the transfer and multitask learning mentioned before can be interpreted as multiobjective optimization problems, see, for example, [Lin+19; SK18], or [MDM20] for the application to a control problem.

List of Abbreviations

AD algorithmic differentiation 27

Adam adaptive moment estimation – a special SGD algorithm for NN training 29

BFGS Broyden-Fletcher-Goldfarb-Shanno update 50

BPTT backpropagation through time 30

CNN convolutional neural network 27

DDE delay differential equation 10

DeepMPC deep model predictive control 53

DMD dynamic mode decomposition 34

eDMD extended dynamic mode decomposition 34

ESN echo state network 32

FEM finite element method 10

FNN feed-forward neural network 25

KKT Karush-Kuhn-Tucker (condition) 39, 42

LSTM long short-term memory 30

MLC machine learning control 19

MOP multiobjective optimization problem 36

MPC model predictive control 14

NN neural network 25

ODE ordinary differential equation 10

PDE partial differential equation 10

POD proper orthogonal decomposition 18

QuaSiModO quantization, simulation, modeling and optimization 69

RBM restricted Boltzmann machine 57

RNN recurrent neural network 29

SGD stochastic gradient descent 28

SINDy sparse identification of nonlinear dynamics 126

SQP sequential quadratic programming 13

SR1 symmetric rank-one update 132

SUR sum-up-rounding 74

SVM support vector machine 143

Bibliography

[AG90] Eugene L. Allgower and Kurt Georg. *Numerical Continuation Methods*. Springer Berlin Heidelberg, 1990. DOI: 10.1007/978-3-642-61257-2 (pages 49, 50, 112, 114, 115, 132).

[AKM18] Hassan Arbabi, Milan Korda, and Igor Mezic. "A Data-Driven Koopman Model Predictive Control Framework for Nonlinear Partial Differential Equations". In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6409–6414. DOI: 10.1109/cdc.2018.8619720 (page 18).

[ALS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. "A Convergence Theory for Deep Learning via Over-Parameterization". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 242–252. URL: https://proceedings.mlr.press/v97/allen-zhu19a.html (page 29).

[AML12] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. AMLBook, 2012 (pages 20, 23, 24).

[AP98] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1998 (page 13).

[Arm+19] Luca Bugliari Armenio, Enrico Terzi, Marcello Farina, and Riccardo Scattolini. "Model Predictive Control Design for Dynamical Systems Learned by Echo State Networks". In: *IEEE Control Systems Letters* 3.4 (2019), pp. 1044–1049. DOI: 10.1109/LCSYS.2019.2920720 (pages 19, 33, 66, 94).

[AS12] Haldun Aytug and Serpil Sayın. "Exploring the trade-off between generalization and empirical errors in a one-norm SVM". In: *European Journal of Operational Research* 218.3 (2012), pp. 667–675. DOI: 10.1016/j.ejor.2011.11.037 (page 105).

[AS14] Ayşegül Aşkan and Serpil Sayın. "SVM classification for imbalanced data sets using a multiobjective optimization framework". In: *Annals of Operations Research* 216.1 (2014), pp. 191–203. DOI: 10.1007/s10479-012-1300-5 (page 105).

[Bar+10] Richard G. Baraniuk, Emmanuel Candes, Michael Elad, and Yi Ma. "Applications of Sparse Representation and Compressive Sensing [Scanning the Issue]". In: *Proceedings of the IEEE* 98.6 (2010), pp. 906–909. DOI: 10.1109/JPROC.2010.2047424 (page 103).

[BB21] Anas Barakat and Pascal Bianchi. "Convergence and Dynamical Behavior of the ADAM Algorithm for Nonconvex Stochastic Optimization". In: *SIAM Journal on Optimization* 31.1 (2021), pp. 244–274. DOI: 10.1137/19M1263443 (page 29).

[BBK18] Thomas Baumeister, Steven L. Brunton, and J. Nathan Kutz. "Deep learning and model predictive control for self-tuning mode-locked lasers". In: *Journal of the Optical Society of America B* 35.3 (2018), pp. 617–626. DOI: 10.1364/JOSAB.35.000617 (pages 3, 19, 54, 57).

[BCB05] Michel Bergmann, Laurent Cordier, and Jean-Pierre Brancher. "Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model". In: *Physics of Fluids* 17.9 (2005), pp. 097101-1–097101-21. DOI: 10.1063/1.2033624 (page 69).

[BCR09] Oliver Brdiczka, James L. Crowley, and Patrick Reignier. "Learning Situation Models in a Smart Home". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.1 (2009), pp. 56–63. DOI: 10.1109/TSMCB.2008.923526 (page 2).

[BD15] Richard E. Bellman and Stuart E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, 2015. DOI: 10.1515/9781400874651 (page 72).

[Bel10] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, 2010. DOI: 10.1515/9781400835386 (page 12).

[Ber+21] Julius Berner, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. *The Modern Mathematics of Deep Learning*. 2021. DOI: 10.48550/ARXIV.2105.04026 (page 25).

[BF09] Ewout van den Berg and Michael P. Friedlander. "Probing the Pareto Frontier for Basis Pursuit Solutions". In: *SIAM Journal on Scientific Computing* 31.2 (2009), pp. 890–912. DOI: 10.1137/080714488 (page 105).

[BF11] Ewout van den Berg and Michael P. Friedlander. "Sparse Optimization with Least-Squares Constraints". In: *SIAM Journal on Optimization* 21.4 (2011), pp. 1201–1229. DOI: 10.1137/100785028 (page 105).

[BGP22] Katharina Bieker, Bennet Gebken, and Sebastian Peitz. "On the Treatment of Optimization Problems With L1 Penalty Terms via Multiobjective Continuation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.11 (Nov. 2022), pp. 7797–7808. DOI: 10.1109/TPAMI.2021.3114962 (pages 6, 105).

[Bie+20] Katharina Bieker, Sebastian Peitz, Steven L. Brunton, J. Nathan Kutz, and Michael Dellnitz. "Deep model predictive flow control with limited sensor data and online learning". In: *Theoretical and Computational Fluid Dynamics* 34.4 (Mar. 2020), pp. 577–591. DOI: 10.1007/s00162-020-00520-4 (pages 6, 19, 54, 96).

[Bin+01] Thomas Binder et al. "Introduction to Model Based Optimization of Chemical Processes on Moving Horizons". In: *Online Optimization of Large Scale Systems*. Springer Berlin Heidelberg, 2001, pp. 295–339. DOI: 10.1007/978-3-662-04331-8_18 (pages 9, 12, 13).

[Bis+21] Bernd Bischl et al. *Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges*. 2021. DOI: 10.48550/ARXIV.2107.05847 (page 66).

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer New York, 2006 (pages 20, 25, 28, 135, 143).

[BK18] Andrew L. Beam and Isaac S. Kohane. "Big Data and Machine Learning in Health Care". In: *JAMA* 319.13 (2018), pp. 1317–1318. DOI: 10.1001/jama.2017.18391 (page 2).

[BK19] Steven L. Brunton and J. Nathan Kutz. *Data-driven science and engineering : machine learning, dynamical systems, and control*. Cambridge: Cambridge University Press, 2019. DOI: 10.1017/9781108380690 (pages 2, 17–19, 22).

[BKM14] Adil Bagirov, Napsu Karmitsa, and Marko M. Mäkelä. *Introduction to Nonsmooth Optimization*. Springer International Publishing, 2014. DOI: 10.1007/978-3-319-08114-4 (pages 41, 135).

[Bot99] Léon Bottou. "On-line learning and stochastic approximations". In: *On-line Learning in Neural Networks*. Cambridge University Press, 1999, pp. 9–42. URL: https://dl.acm.org/doi/10.5555/304710.304720 (page 28).

[BPK16] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113 (pages 2, 92, 103, 125, 126).

[Bra+04] Jürgen Branke, Kalyanmoy Deb, Henning Dierolf, and Matthias Osswald. "Finding Knees in Multi-objective Optimization". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 722–731. DOI: 10.1007/978-3-540-30217-9_73 (page 128).

[Bri+18] Bjoern Bringmann, Daniel Cremers, Felix Krahmer, and Michael Moeller. "The homotopy method revisited: Computing solution paths of ℓ_1-regularized problems". In: *Mathematics of Computation* 87.313 (2018), pp. 2343–2364. DOI: 10.1090/mcom/3287 (page 104).

[Bru+17] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, and J. Nathan Kutz. "Chaos as an intermittently forced linear system". In: *Nature Communications* 8.1 (2017). DOI: 10.1038/s41467-017-00030-8 (page 35).

[Bru+21] Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. *Modern Koopman Theory for Dynamical Systems*. 2021. DOI: 10.48550/ARXIV.2102.12086 (pages 2, 33, 36).

[BS15] Felix Berkenkamp and Angela P. Schoellig. "Safe and robust learning control with Gaussian processes". In: *2015 European Control Conference (ECC)*. 2015, pp. 2496–2501. DOI: 10.1109/ECC.2015.7330913 (page 149).

[BS21] Mrinal R. Bachute and Javed M. Subhedar. "Autonomous Driving Architectures: Insights of Machine Learning and Deep Learning Algo-

rithms". In: *Machine Learning with Applications* 6 (2021), p. 100164. DOI: 10.1016/j.mlwa.2021.100164 (page 2).

[But+18] Keith T. Butler, Daniel W. Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. "Machine learning for molecular and materials science". In: *Nature* 559.7715 (2018), pp. 547–555. DOI: 10.1038/s41586-018-0337-2 (page 2).

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441 (page 48).

[BW20] Dennis M Bushnell and Israel Wygnanski. "Flow Control Applications". In: *NASA/TM – 2020 – 220436* (2020). URL: https://ntrs.nasa.gov/citations/20200000702 (page 53).

[BZ03] Alfredo Bellen and Marino Zennaro. *Numerical Methods for Delay Differential Equations*. Oxford University Press, 2003. DOI: 10.1093/acprof:oso/9780198506546.001.0001 (page 10).

[CC21] Kushal Chakrabarti and Nikhil Chopra. "Generalized AdaGrad (G-AdaGrad) and Adam: A State-Space Perspective". In: *60th IEEE Conference on Decision and Control (CDC)*. 2021, pp. 1496–1501. DOI: 10.1109/CDC45484.2021.9682994 (page 29).

[CD94] Shaobing Chen and D. Donoho. "Basis pursuit". In: *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*. Vol. 1. IEEE, 1994, pp. 41–44. DOI: 10.1109/ACSSC.1994.471413 (page 104).

[Cha04] Antonin Chambolle. "An Algorithm for Total Variation Minimization and Applications". In: *Journal of Mathematical Imaging and Vision* 20 (2004), pp. 89–97. DOI: 10.1023/b:jmiv.0000011325.36760.1e (page 135).

[CHS20] Ashesh Chattopadhyay, Pedram Hassanzadeh, and Devika Subramanian. "Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network". In: *Nonlinear Processes in Geophysics* 27.3 (2020), pp. 373–389. DOI: 10.5194/npg-27-373-2020 (page 33).

[Cla90] Frank H. Clarke. *Optimization and Nonsmooth Analysis*. Society for Industrial and Applied Mathematics, 1990. DOI: 10.1137/1.9781611971309 (page 41).

[Coe06] Carlos A. Coello Coello. "Evolutionary multi-objective optimization: a historical view of the field". In: *IEEE Computational Intelligence Magazine* 1.1 (2006), pp. 28–36. DOI: 10.1109/MCI.2006.1597059 (page 51).

[Cor+21] Guy Y. Cornejo Maceda, Yiqing Li, François Lusseyran, Marek Morzyński, and Bernd R. Noack. "Stabilization of the fluidic pinball with gradient-enriched machine learning control". In: *Journal of Fluid Mechanics* 917 (2021), A42-1–A42-43. DOI: 10.1017/jfm.2021.301 (page 54).

[CR80] Charles R. Cutler and Brian L. Ramaker. "Dynamic matrix control – A computer control algorithm". In: *Joint Automatic Control Conference* 17 (1980), p. 72 (pages 2, 17).

[Dai+13] Jisheng Dai, Chunqi Chang, Fei Mai, Dean Zhao, and Weichao Xu. "On the SVMpath Singularity". In: *IEEE Transactions on Neural Networks and Learning Systems* 24.11 (2013), pp. 1736–1748. DOI: 10.1109/tnnls.2013.2262180 (page 144).

[Dar59] Charles Darwin. *On the Origin of Species by Means of Natural Selection.* or the Preservation of Favored Races in the Struggle for Life. Murray, 1859 (page 50).

[DBN17] Thomas Duriez, Steven L. Brunton, and Bernd R. Noack. *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence.* Springer International Publishing, 2017. DOI: 10.1007/978-3-319-40624-4 (pages 18, 19).

[Dea+91] Anil E. Deane, Ioannis G. Kevrekidis, George Em Karniadakis, and Steven A. Orszag. "Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders". In: *Physics of Fluids A: Fluid Dynamics* 3.10 (1991), pp. 2337–2354. DOI: 10.1063/1.857881 (page 96).

[Deb+02] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and Thirunavukarasu Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017 (page 51).

[Den+18] Nan Deng, Luc R. Pastur, Marek Morzyński, and Bernd R. Noack. "Route to Chaos in the Fluidic Pinball". In: *Proceedings of the ASME 2018 Fluids Engineering Division Summer Meeting.* American Society of Mechanical Engineers, 2018. DOI: 10.1115/FEDSM2018-83359 (pages 3, 53, 147).

[Den+19] Nan Deng, Bernd R. Noack, Marek Morzyński, and Luc R. Pastur. "Low-order model for successive bifurcations of the fluidic pinball". In: *Journal of Fluid Mechanics* 884 (2019). DOI: 10.1017/jfm.2019.959 (pages 53, 58–60, 147).

[DER95] Andreas Draeger, Sebastian Engell, and Horst Ranke. "Model predictive control using neural networks". In: *IEEE Control Systems Magazine* 15.5 (1995), pp. 61–66. DOI: 10.1109/37.466261 (page 19).

[DF21] Eva Dierkes and Kathrin Flaßkamp. "Learning Hamiltonian Systems considering System Symmetries in Neural Networks". In: *IFAC-PapersOnLine* 54.19 (2021). 7th IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control LHMNC 2021, pp. 210–216. DOI: 10.1016/j.ifacol.2021.11.080 (page 19).

[Don06] David L. Donoho. "For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution". In: *Communications on Pure and Applied Mathematics* 59.6 (2006), pp. 797–829. DOI: 10.1002/cpa.20132 (page 24).

[DT08] David L. Donoho and Yaakov Tsaig. "Fast Solution of ℓ_1-Norm Minimization Problems When the Solution May Be Sparse". In: *IEEE Transactions on Information Theory* 54.11 (2008), pp. 4789–4812. DOI: 10.1109/TIT.2008.929958 (page 104).

[Du+19] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. "Gradient descent finds global minima of deep neural networks". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*. Vol. 97. Proceedings of Machine Learning Research. PMLR. 2019, pp. 1675–1685. URL: http://proceedings.mlr.press/v97/du19c/du19c.pdf (page 29).

[E+20] Weinan E, Chao Ma, Stephan Wojtowytsch, and Lei Wu. *Towards a Mathematical Understanding of Neural Network-Based Machine Learning: what we know and what we don't*. 2020. DOI: 10.48550/ARXIV.2009.10713 (page 25).

[EFM10] Michael Elad, Mário A. T. Figueiredo, and Yi Ma. "On the Role of Sparse and Redundant Representations in Image Processing". In: *Proceedings of the IEEE* 98.6 (2010), pp. 972–982. DOI: 10.1109/JPROC.2009.2037655 (page 103).

[Efr+04] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. "Least angle regression". In: *The Annals of Statistics* 32.2 (2004), pp. 407–499. DOI: 10.1214/009053604000000067 (page 137).

[EG15] Lawrence Craig Evans and Ronald F. Gariepy. *Measure Theory and Fine Properties of Functions, Revised Edition*. Chapman and Hall/CRC, 2015. DOI: 10.1201/b18333 (page 41).

[Ehr05] Matthias Ehrgott. *Multicriteria optimization*. 2nd ed. Springer Berlin, 2005. DOI: 10.1007/3-540-27659-9 (pages 36, 37, 45, 125).

[EK12] Yonina C. Eldar and Gitta Kutyniok, eds. *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012. DOI: 10.1017/CBO9780511794308 (page 103).

[EMH19] Thomas Elsken, Jan H. Metzen, and Frank Hutter. "Neural Architecture Search: A Survey". In: *Journal of Machine Learning Research* 20.55 (2019), pp. 1–21. URL: http://jmlr.org/papers/v20/18-598.html (page 66).

[ES15] A.E. Eiben and Jim E. Smith. *Introduction to Evolutionary Computing*. Springer Berlin, Heidelberg, 2015. DOI: 10.1007/978-3-662-44874-8 (page 51).

[FDS09] Jörg Fliege, L. M. Graña Drummond, and Benar F. Svaiter. "Newton's Method for Multiobjective Optimization". In: *SIAM Journal on Optimization* 20.2 (2009), pp. 602–626. DOI: 10.1137/08071692x (page 47).

[Fer12] David A. Ferrucci. "Introduction to "This is Watson"". In: *IBM Journal of Research and Development* 56.3.4 (2012), 1:1–1:15. DOI: 10.1147/JRD.2012.2184356 (page 2).

[Fis36] Ronald A. Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of Eugenics* 7.2 (1936), pp. 179–188 (pages 21, 129).

[FLK19] Akhmad Faqih, Aldo Pratama Lianto, and Benyamin Kusumoputro. "Mackey-Glass Chaotic Time Series Prediction Using Modified RBF Neural Networks". In: *Proceedings of the 2nd International Conference on Software Engineering and Information Management*. ICSIM 2019. Association for Computing Machinery, 2019, pp. 7–11 (page 93).

[For+15] Michael G. Forbes, Rohit S. Patwardhan, Hamza Hamadah, and R. Bhushan Gopaluni. "Model Predictive Control in Industry: Challenges and Opportunities". In: *IFAC-PapersOnLine* 48.8 (2015). 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015, pp. 531–538. DOI: 10.1016/j.ifacol.2015.09.022 (pages 2, 17).

[FS00] Jörg Fliege and Benar F. Svaiter. "Steepest descent methods for multicriteria optimization". In: *Mathematical Methods of Operations Research (ZOR)* 51.3 (2000), pp. 479–494. DOI: 10.1007/s001860000043 (pages 46, 47).

[Gal11] Jean Gallier. *Geometric Methods and Applications*. Texts in applied mathematics. Springer New York, 2011. DOI: 10.1007/978-1-4419-9961-0 (pages 82, 139).

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010*. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html (pages 57, 67, 132).

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. URL: http://www.deeplearningbook.org (pages 2, 25, 28, 29, 135).

[GBP22] Bennet Gebken, Katharina Bieker, and Sebastian Peitz. "On the structure of regularization paths for piecewise differentiable regularization terms". In: *Journal of Global Optimization* (Sept. 2022). DOI: 10.1007/s10898-022-01223-2 (pages 7, 105, 137, 141, 142, 144).

[Geb22] Bennet Gebken. "Computation and analysis of Pareto critical sets in smooth and nonsmooth multiobjective optimization". PhD thesis. 2022. DOI: 10.17619/UNIPB/1-1327 (page 112).

[Ger11] Matthias Gerdts. *Optimal Control of ODEs and DAEs*. De Gruyter, 2011. DOI: 10.1515/9783110249996 (page 77).

[GK71] Ernest William Griffith and Kadaba S. P. Kumar. "On the observability of nonlinear systems: I". In: *Journal of Mathematical Analysis and Applications* 35.1 (1971), pp. 135–147. DOI: 10.1016/0022-247X(71)90241-1 (page 18).

[GM79] Leon Glass and Michael C. Mackey. "Pathological conditions resulting from instabilities in physiological control systems". In: *Annals of the New York Academy of Sciences* 316.1 (1979), pp. 214–235. DOI: 10.1111/j.1749-6632.1979.tb29471.x (page 93).

[GP17] Lars Grüne and Jürgen Pannek. *Nonlinear Model Predictive Control*. 2nd ed. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-46024-6 (pages 1, 2, 14, 15, 18).

[GP21] Bennet Gebken and Sebastian Peitz. "An Efficient Descent Method for Locally Lipschitz Multiobjective Optimization Problems". In: *Journal of Optimization Theory and Applications* 188.3 (2021), pp. 696–723. DOI: 10.1007/s10957-020-01803-w (page 48).

[GPD19] Bennet Gebken, Sebastian Peitz, and Michael Dellnitz. "On the hierarchical structure of Pareto critical sets". In: *Journal of Global Optimization* 73.4 (2019), pp. 891–913. DOI: 10.1007/s10898-019-00737-6 (pages 121, 142).

[GPT99] W. R. Graham, Jaume Peraire, and K. Y. Tang. "Optimal control of vortex shedding using low-order models. Part I – open-loop model development". In: *International Journal for Numerical Methods in Engineering* 44.7 (1999), pp. 945–972. DOI: 10.1002/(SICI)1097-0207(19990310)44:7<945::AID-NME537>3.0.CO;2-F (page 69).

[Gre+17] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232. DOI: 10.1109/TNNLS.2016.2582924 (page 31).

[Grö19] Thomas Hakon Grönwall. "Note on the Derivatives with Respect to a Parameter of the Solutions of a System of Differential Equations". In: *Annals of Mathematics* 20.4 (1919), pp. 292–296. DOI: 10.2307/1967124 (page 75).

[GSS15] Alireza Goudarzi, Alireza Shabani, and Darko Stefanovic. "Product reservoir computing: Time-series computation with multiplicative neurons". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8. DOI: 10.1109/ijcnn.2015.7280453 (page 93).

[GW08a] Rafael C. González and Richard E. Woods. *Digital image processing, 3rd Edition*. Pearson Education, 2008 (page 133).

[GW08b] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008. DOI: 10.1137/1.9780898717761 (pages 27, 28).

[Has+04] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. "The Entire Regularization Path for the Support Vector Machine". In: *Journal of Machine Learning Research* 5 (2004), pp. 1391–1415. URL: http://jmlr.org/papers/volume5/hastie04a/hastie04a.pdf (pages 137, 144).

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123 (page 66).

[Hes+18] Michael Hesse, Julia Timmermann, Eyke Hüllermeier, and Ansgar Trächtler. "A Reinforcement Learning Strategy for the Swing-Up of the Double Pendulum on a Cart". In: *Procedia Manufacturing* 24 (2018), pp. 15–20. DOI: 10.1016/j.promfg.2018.06.004 (pages 19, 22).

[Hil01] Claus Hillermeier. *Nonlinear Multiobjective Optimization*. Birkhäuser Basel, 2001. DOI: 10.1007/978-3-0348-8280-4 (pages 5, 48–50, 111–113, 115, 150).

[HL06] Márton Albert Hajnal and András Lőrincz. "Critical Echo State Networks". In: *Artificial Neural Networks – ICANN 2006*. Springer Berlin Heidelberg, 2006, pp. 658–667. DOI: 10.1007/11840817_69 (page 93).

[Hoi+21] Steven C.H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. "Online learning: A comprehensive survey". In: *Neurocomputing* 459 (2021), pp. 249–289. DOI: 10.1016/j.neucom.2021.04.112 (page 22).

[HS06] Geoffrey E. Hinton and Ruslan Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647 (page 57).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735 (pages 2, 30, 31).

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8 (page 26).

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. 2nd ed. Springer Series in Statistics. Springer New York, 2009. DOI: 10.1007/978-0-387-84858-7 (page 135).

[Igl+18] Ramon Iglesias et al. "Data-Driven Model Predictive Control of Autonomous Mobility-on-Demand Systems". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6019–6025. DOI: 10.1109/ICRA.2018.8460966 (pages 19, 66).

[Jae02] Herbert Jaeger. *Short term memory in echo state networks*. Vol. 152. German National Research Center for Information Technology, 2002. URL: https://www.ai.rug.nl/minds/uploads/STMEchoStatesTechRep.pdf (page 32).

[Jae10] Herbert Jaeger. "The "echo state" approach to analysing and training recurrent neural networks – with an erratum note". In: *GMD Technical Report* 148 (2010), pp. 1–47. URL: https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf (pages 2, 32, 33, 93).

[JH04] Herbert Jaeger and Harald Haas. "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication". In: *Science* 304.5667 (2004), pp. 78–80. DOI: 10.1126/science.1091277 (pages 2, 32).

[JJT07] Hrvoje Jasak, Aleksandar Jemcov, and Željko Tuković. "OpenFOAM : A C++ Library for Complex Physics Simulations". In: *International Workshop on Coupled Methods in Numerical Dynamics - CMND2007* (2007), pp. 47–66 (page 59).

[Jor+18] Jean P. Jordanou, Eduardo Camponogara, Eric Aislan Antonelo, and Marco Aurélio Schmitz Aguiar. "Nonlinear Model Predictive Control of an Oil Well with Echo State Networks". In: *IFAC-PapersOnLine* 51.8 (2018), pp. 13–18. DOI: 10.1016/j.ifacol.2018.06.348 (pages 19, 33, 66, 94).

[Kal60a] Rudolf E. Kalman. "On the general theory of control systems". In: *IFAC Proceedings Volumes* 1.1 (1960). 1st International IFAC Congress on

Automatic and Remote Control, Moscow, USSR, 1960, pp. 491–502. DOI: 10.1016/S1474-6670(17)70094-8 (page 18).

[Kal60b] Rudolf. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45. DOI: 10.1115/1.3662552 (page 19).

[Kar+21] George Em Karniadakis et al. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440. DOI: 10.1038/s42254-021-00314-5 (pages 19, 67, 148).

[Kar39] William Karush. "Minima of Functions of Several Variables with Inequalities as Side Conditions". MA thesis. Department of Mathematics, University of Chicago, 1939 (page 39).

[KB14] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980 (page 29).

[KBP13] Jens Kober, J. Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274. DOI: 10.1177/0278364913495721 (pages 19, 22).

[Kir+22] B. Ravi Kiran et al. "Deep Reinforcement Learning for Autonomous Driving: A Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 4909–4926. DOI: 10.1109/TITS.2021.3054625 (page 2).

[KKB18] Eurika Kaiser, J. Nathan Kutz, and Steven L. Brunton. "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2219 (2018), p. 20180335. DOI: 10.1098/rspa.2018.0335 (pages 69, 92).

[KKS16] Stefan Klus, Péter Koltai, and Christof Schütte. "On the numerical approximation of the Perron-Frobenius and Koopman operator". In: *Journal of Computational Dynamics* 3.1 (2016), pp. 51–79. DOI: 10.3934/jcd.2016003 (page 34).

[Klu+20] Stefan Klus et al. "Data-driven approximation of the Koopman generator: Model reduction, system identification, and control". In: *Physica D: Nonlinear Phenomena* 406 (2020), p. 132416. DOI: 10.1016/j.physd.2020.132416 (pages 35, 36).

[KM18a] Milan Korda and Igor Mezić. "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control". In: *Automatica* 93 (2018), pp. 149–160. DOI: 10.1016/j.automatica.2018.03.046 (pages 18, 35, 69).

[KM18b] Milan Korda and Igor Mezić. "On Convergence of Extended Dynamic Mode Decomposition to the Koopman Operator". In: *Journal of Nonlinear Science* 28.2 (2018), pp. 687–710. DOI: 10.1007/s00332-017-9423-0 (page 35).

[Koo31] Bernard O. Koopman. "Hamiltonian Systems and Transformation in Hilbert Space". In: *Proceedings of the National Academy of Sciences* 17.5 (1931), pp. 315–318. DOI: 10.1073/pnas.17.5.315 (pages 2, 33).

[KR17] Gábor Kiss and Gergely Röst. "Controlling Mackey–Glass chaos". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.11 (2017), p. 114321. DOI: 10.1063/1.5006922 (page 93).

[Kra85] Dieter Kraft. "On Converting Optimal Control Problems into Nonlinear Programming Problems". In: *Computational Mathematical Programming*. Springer Berlin Heidelberg, 1985, pp. 261–280. DOI: 10.1007/978-3-642-82450-0_9 (page 13).

[Kra94] Dieter Kraft. "Algorithm 733: TOMP–Fortran modules for optimal control calculations". In: *ACM Transactions on Mathematical Software* 20.3 (1994), pp. 262–281. DOI: 10.1145/192115.192124 (page 13).

[KT51] Harold W. Kuhn and Albert W. Tucker. "Nonlinear programming". In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Calif.: University of California Press, 1951, pp. 481–492 (page 39).

[KV99] Karl Kunisch and Stefan Volkwein. "Control of the Burgers Equation by a Reduced-Order Approach Using Proper Orthogonal Decomposition". In: *Journal of Optimization Theory and Applications* 102.2 (1999), pp. 345–371. DOI: 10.1023/a:1021732508059 (page 18).

[LCB10] Yann LeCun, Corinna Cortes, and Chris J. Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]* 2 (2010). URL: http://yann.lecun.com/exdb/mnist (page 133).

[Lee11] Jay H. Lee. "Model predictive control: Review of the three decades of development". In: *International Journal of Control, Automation and Systems* 9.3 (2011), pp. 415–424. DOI: 10.1007/s12555-011-0300-6 (pages 2, 17).

[Lee12] John M. Lee. *Introduction to Smooth Manifolds*. Springer New York, 2012. DOI: 10.1007/978-1-4419-9982-5 (pages 138, 142).

[LHO18] Zhixin Lu, Brian R. Hunt, and Edward Ott. "Attractor reconstruction by machine learning". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.6 (2018), p. 061104. DOI: 10.1063/1.5039508 (page 33).

[LHW12] Decai Li, Min Han, and Jun Wang. "Chaotic Time Series Prediction Based on a Novel Robust Echo State Network". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.5 (2012), pp. 787–799. DOI: 10.1109/TNNLS.2012.2188414 (page 33).

[Li+12] Hui Li, Xiaolei Su, Zongben Xu, and Qingfu Zhang. "MOEA/D with Iterative Thresholding Algorithm for Sparse Optimization Problems". In: *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature - PPSN XII*. Vol. 7492. Lecture Notes in Computer Science. Springer, 2012, pp. 93–101. DOI: 10.1007/978-3-642-32964-7_10 (page 105).

[Lib12] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2012. DOI: 10.1515/9781400842643 (pages 9–12).

[Lin+19] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. "Pareto Multi-Task Learning". In: *Advances in Neural Information Pro-*

cessing Systems. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/685bfde03eb646c27ed565881917c71c-Paper.pdf (page 151).

[LJ09] Mantas Lukosevicius and Herbert Jaeger. "Reservoir computing approaches to recurrent neural network training". In: *Computer Science Review* 3.3 (2009), pp. 127–149. DOI: 10.1016/j.cosrev.2009.03.005 (page 32).

[Lju98] Lennart Ljung. "System Identification". In: *Signal Analysis and Prediction*. Birkhäuser Boston, 1998, pp. 163–173. DOI: 10.1007/978-1-4612-1768-8_11 (page 18).

[LKS15] Ian Lenz, Ross A. Knepper, and Ashutosh Saxena. "DeepMPC: Learning Deep Latent Features for Model Predictive Control". In: *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*. 2015. DOI: 10.15607/RSS.2015.XI.012 (pages 3, 54, 57).

[Lóp+16] Carlos H. López-Caraballo et al. "Mackey-Glass noisy chaotic time series prediction by a swarm-optimized neural network". In: *Journal of Physics: Conference Series* 720 (2016), p. 012002. DOI: 10.1088/1742-6596/720/1/012002 (page 93).

[LT20] H. Lu and D. M. Tartakovsky. "Predictive Accuracy of Dynamic Mode Decomposition". In: *SIAM Journal on Scientific Computing* 42.3 (2020), pp. 1639–1662. DOI: 10.1137/19M1259948 (page 35).

[LV21] Suyun Liu and Luis Nunes Vicente. "The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning". In: *Annals of Operations Research* (2021). DOI: 10.1007/s10479-021-04033-z (pages 48, 150).

[LVV12] Frank L. Lewis, Draguna Vrabie, and Kyriakos G. Vamvoudakis. "Reinforcement Learning and Feedback Control: Using Natural Decision Methods to Design Optimal Adaptive Controllers". In: *IEEE Control Systems Magazine* 32.6 (2012), pp. 76–105. DOI: 10.1109/MCS.2012.2214134 (pages 19, 22).

[Man+18] Krithika Manohar, Bingni W. Brunton, J. Nathan Kutz, and Steven L. Brunton. "Data-Driven Sparse Sensor Placement for Reconstruction: Demonstrating the Benefits of Exploiting Known Patterns". In: *IEEE Control Systems Magazine* 38.3 (2018), pp. 63–86. DOI: 10.1109/MCS.2018.2810460 (page 53).

[Mar+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (pages 28, 59).

[Mar14] Adanay Martín. "Pareto Tracer: A Predictor Corrector Method for Multi-objective Optimization Problems". MA thesis. 2014 (page 132).

[MBS93] Thomas M. Martinets, Stanislav G. Berkovich, and Klaus J. Schulten. "'Neural-gas' network for vector quantization and its application to time-series prediction". In: *IEEE Transactions on Neural Networks* 4.4 (1993), pp. 558–569. DOI: 10.1109/72.238311 (page 93).

[MCW05] D. M. Malioutov, M. Cetin, and A. S. Willsky. "Homotopy continuation for sparse signal representation". In: *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* Vol. 5. 2005, v/733–v/736 Vol. 5. DOI: 10.1109/ICASSP.2005.1416408 (page 104).

[MDM20] Pingchuan Ma, Tao Du, and Wojciech Matusik. "Efficient Continuous Pareto Exploration in Multi-Task Learning". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020.* Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 6522–6531. URL: http://proceedings.mlr.press/v119/ma20a.html (page 151).

[MEK14] Marko M. Mäkelä, Ville-Pekka Eronen, and Napsu Karmitsa. "On Nonsmooth Multiobjective Optimality Conditions with Generalized Convexities". In: *Optimization in Science and Engineering: In Honor of the 60th Birthday of Panos M. Pardalos.* Springer New York, 2014, pp. 333–357. DOI: 10.1007/978-1-4939-0808-0_17 (pages 41, 42).

[MG77] Michael C. Mackey and Leon Glass. "Oscillation and Chaos in Physiological Control Systems". In: *Science* 197.4300 (1977), pp. 287–289. DOI: 10.1126/science.267326 (page 93).

[Mie98] Kaisa Miettinen. *Nonlinear multiobjective optimization.* 1st ed. Springer New York, 1998. DOI: 10.1007/978-1-4615-5563-6 (pages 36, 40–42).

[MK20] Paul Manns and Christian Kirches. "Improved regularity assumptions for partial outer convexification of mixed-integer PDE-constrained optimization problems". In: *ESAIM: Control, Optimisation and Calculus of Variations* 26 (2020), p. 32. DOI: 10.1051/cocv/2019016 (pages 77, 78).

[MKW15] Marko M. Mäkelä, Napsu Karmitsa, and Outi Wilppu. "Proximal Bundle Method for Nonsmooth and Nonconvex Multiobjective Optimization". In: *Computational Methods in Applied Sciences.* Springer International Publishing, 2015, pp. 191–204. DOI: 10.1007/978-3-319-23564-6_12 (page 48).

[MOO13] Alejandro Marquez, Jairo José Espinosa Oviedo, and Darci Odloak. "Model Reduction Using Proper Orthogonal Decomposition and Predictive Control of Distributed Reactor System". In: *Journal of Control Science and Engineering* 2013 (2013), pp. 1–19. DOI: 10.1155/2013/763165 (page 18).

[Mor+18] Jeremy Morton, Antony Jameson, Mykel J. Kochenderfer, and Freddie D. Witherden. "Deep Dynamical Modeling and Control of Unsteady Fluid Flows". In: *Advances in Neural Information Processing Systems.* Vol. 31. Curran Associates, Inc., 2018, pp. 9258–9268. URL: https://proceedings.neurips.cc/paper/2018/file/2b0aa0d9e30ea3a55fc271ced8364536-Paper.pdf (page 54).

[MPD18] Quentin Mercier, Fabrice Poirion, and Jean-Antoine Désidéri. "A stochastic multiple gradient descent algorithm". In: *European Journal of Operational Research* 271.3 (2018), pp. 808–817. DOI: 10.1016/j.ejor.2018.05.064 (pages 48, 150).

[MRW21] Ian R. Manchester, Max Revay, and Ruigang Wang. "Contraction-Based Methods for Stable Identification and Robust Machine Learning: a Tutorial". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021, pp. 2955–2962. DOI: 10.1109/CDC45484.2021.9683128 (page 149).

[MS17] Adanay Martín and Oliver Schütze. "Pareto Tracer: a predictor–corrector method for multi–objective optimization problems". In: *Engineering Optimization* 50.3 (2017), pp. 516–536. DOI: 10.1080/0305215x.2017.1327579 (pages 50, 113, 115, 132).

[MW17] Patrick L. McDermott and Christopher K. Wikle. "An Ensemble Quadratic Echo State Network for Nonlinear Spatio-Temporal Forecasting". In: *Stat* 6.1 (2017), pp. 315–330. DOI: 10.1002/sta4.160 (page 33).

[Nis97] Volker Nissen. *Einführung in Evolutionäre Algorithmen*. Vieweg+Teubner Verlag Wiesbaden, 1997. DOI: 10.1007/978-3-322-93861-9 (page 51).

[Noa+03] Bernd R. Noack, Konstantin Afanasiev, Marek Morzyński, Gilead Tadmor, and Frank Thiele. "A hierarchy of low-dimensional models for the transient and post-transient cylinder wake". In: *Journal of Fluid Mechanics* 497 (2003), pp. 335–363. DOI: 10.1017/s0022112003006694 (page 96).

[Noa+16] Bernd R. Noack, Witold Stankiewicz, Marek Morzyński, and Peter J. Schmid. "Recursive dynamic mode decomposition of transient and post-transient wake flows". In: *Journal of Fluid Mechanics* 809 (2016), pp. 843–872. DOI: 10.1017/jfm.2016.678 (pages 3, 53).

[Nüs+21] Feliks Nüske, Sebastian Peitz, Friedrich Philipp, Manuel Schaller, and Karl Worthmann. *Finite-data error bounds for Koopman-based prediction and control*. 2021. DOI: 10.48550/ARXIV.2108.07102 (pages 35, 36, 84).

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2nd ed. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. DOI: 10.1007/978-0-387-40065-5 (pages 13, 27, 47, 50, 57, 74, 132, 135).

[OO23] Sina Ober-Blöbaum and Christian Offen. "Variational learning of Euler–Lagrange dynamics from data". In: *Journal of Computational and Applied Mathematics* 421 (Mar. 2023), p. 114780 (page 19).

[OPT00] M. R. Osborne, B. Presnell, and B. A. Turlach. "A new approach to variable selection in least squares problems". In: *IMA Journal of Numerical Analysis* 20.3 (2000), pp. 389–403. DOI: 10.1093/imanum/20.3.389 (pages 104, 137).

[OSY10] Chong-Jin Ong, Shiyun Shao, and Jianbo Yang. "An Improved Algorithm for the Solution of the Regularization Path of Support Vector Machine". In: *IEEE Transactions on Neural Networks* 21.3 (2010), pp. 451–462. DOI: 10.1109/tnn.2009.2039000 (page 144).

[PA16] Alberto Padoan and Alessandro Astolfi. "A Note on Delay Coordinates for Locally Observable Analytic Systems". In: *IEEE Transactions on*

Automatic Control 61.5 (2016), pp. 1409–1412. DOI: 10.1109/TAC.2015.2478128 (page 18).

[Par06] V. Pareto. *Manuale di Economia Politica*. Piccola biblioteca scientifica. Societa Editrice, 1906 (page 36).

[Pas+19a] Luc R. Pastur, Nan Deng, Marek Morzyński, and Bernd R. Noack. "Reduced-Order Modeling of the Fluidic Pinball". In: *11th Chaotic Modeling and Simulation International Conference*. Springer International Publishing, 2019, pp. 205–213. DOI: 10.1007/978-3-030-15297-0_19 (pages 53, 58).

[Pas+19b] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (page 28).

[Pat+17] Jaideep Pathak, Zhixin Lu, Brian R. Hunt, Michelle Girvan, and Edward Ott. "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.12 (2017), p. 121102. DOI: 10.1063/1.5010300 (page 33).

[Pat+18] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. "Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach". In: *Physical Review Letters* 120.2 (2018), p. 24102. DOI: 10.1103/PhysRevLett.120.024102 (page 33).

[PB23] Sebastian Peitz and Katharina Bieker. "On the universal transformation of data-driven models to control systems". In: *Automatica* 149 (2023), p. 110840. DOI: 10.1016/j.automatica.2022.110840 (pages 6, 70).

[PBK16] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. "Dynamic Mode Decomposition with Control". In: *SIAM Journal on Applied Dynamical Systems* 15.1 (2016), pp. 142–161. DOI: 10.1137/15M1013857 (pages 35, 69).

[PBK18] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. "Generalizing Koopman Theory to Allow for Inputs and Control". In: *SIAM Journal on Applied Dynamical Systems* 17.1 (2018), pp. 909–930. DOI: 10.1137/16m1062296 (page 35).

[PH07] Mee Young Park and Trevor Hastie. "L1-regularization path algorithm for generalized linear models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.4 (2007), pp. 659–677. DOI: https://doi.org/10.1111/j.1467-9868.2007.00607.x (page 104).

[PK19] Sebastian Peitz and Stefan Klus. "Koopman operator-based model reduction for switched-system control of PDEs". In: *Automatica* 106 (2019), pp. 184–191. DOI: 10.1016/j.automatica.2019.05.016 (pages 4, 18, 36, 53, 69, 148).

[Plu+10] Mark D. Plumbley, Thomas Blumensath, Laurent Daudet, Rémi Gribonval, and Mike E. Davies. "Sparse Representations in Audio and Music: From Coding to Source Separation". In: *Proceedings of the*

IEEE 98.6 (2010), pp. 995–1005. DOI: 10.1109/JPROC.2009.2030345 (page 103).

[Pol64] Boris T. Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17. DOI: 10.1016/0041-5553(64)90137-5 (page 29).

[POR20] Sebastian Peitz, Samuel E. Otto, and Clarence W. Rowley. "Data-Driven Model Predictive Control using Interpolated Koopman Generators". In: *SIAM Journal on Applied Dynamical Systems* 19.3 (2020), pp. 2162–2193. DOI: 10.1137/20m1325678 (pages 4, 18, 36, 53, 58, 62, 69, 70, 148, 149).

[Pyt99] Radosław Pytlak. *Numerical Methods for Optimal Control Problems with State Constraints*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1999. DOI: 10.1007/BFb0097244 (page 13).

[Rai+20] Cédric Raibaudo, P. Zhong, Bernd R. Noack, and Robert J. Martinuzzi. "Machine learning strategies applied to the control of a fluidic pinball". In: *Physics of Fluids* 32.1 (2020), pp. 015108-1–015108-13. DOI: 10.1063/1.5127202 (page 54).

[Rao09] Anil V. Rao. "A survey of numerical methods for optimal control". In: *Advances in the Astronautical Sciences* 135.1 (2009), pp. 497–528 (page 12).

[Rei+22] Malena Reiners, Kathrin Klamroth, Fabian Heldmann, and Michael Stiglmayr. "Efficient and sparse neural networks by pruning weights in a multiobjective learning approach". In: *Computers & Operations Research* 141 (2022), p. 105676. DOI: 10.1016/j.cor.2021.105676 (pages 105, 151).

[Ric+78] J. Richalet, A. Rault, J.L. Testud, and J. Papon. "Model predictive heuristic control: Applications to industrial processes". In: *Automatica* 14.5 (1978), pp. 413–428. DOI: 10.1016/0005-1098(78)90001-8 (pages 2, 17).

[Rid+21] Steffen Ridderbusch, Christian Offen, Sina Ober-Blöbaum, and Paul Goulart. "Learning ODE Models with Qualitative Structure Using Gaussian Processes". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021, pp. 2896–2896. DOI: 10.1109/CDC45484.2021.9683426 (page 19).

[RKK18] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. "On the Convergence of Adam and Beyond". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=ryQu7f-RZ (page 29).

[RM21] Cédric Raibaudo and Robert J. Martinuzzi. "Unsteady actuation and feedback control of the experimental fluidic pinball using genetic programming". In: *Experiments in Fluids* 62.219 (2021), pp. 1–18. DOI: 10.1007/s00348-021-03309-1 (page 54).

[RM51] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586 (page 28).

[Ros04] Saharon Rosset. "Following Curved Regularized Optimization Solution Paths". In: *Advances in Neural Information Processing Systems*. Vol. 17. MIT Press, 2004, pp. 1153–1160. URL: https://proceedings.neurips.cc/paper/2004/hash/32b991e5d77ad140559ffb95522992d0-Abstract.html (page 104).

[RZ07] Saharon Rosset and Ji Zhu. "Piecewise linear regularized solution paths". In: *The Annals of Statistics* 35.3 (2007), pp. 1012–1030. DOI: 10.1214/009053606000001370 (pages 137, 150).

[RZH04] Saharon Rosset, Ji Zhu, and Trevor Hastie. "Boosting as a Regularized Path to a Maximum Margin Classifier". In: *Journal of Machine Learning Research* 5 (2004), pp. 941–973. URL: https://dl.acm.org/doi/10.5555/1005332.1016790 (page 24).

[Sag05] Sebastian Sager. "Numerical Methods for Mixed-Integer Optimal Control Problems". PhD thesis. 2005 (pages 73, 74).

[SAG16] Christopher G. Sentelle, Georgios C. Anagnostopoulos, and Michael Georgiopoulos. "A Simple Method for Solving the SVM Regularization Path for Semidefinite Kernels". In: *IEEE Transactions on Neural Networks and Learning Systems* 27.4 (2016), pp. 709–722. DOI: 10.1109/tnnls.2015.2427333 (page 144).

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (page 19).

[SB92] Oskar von Stryk and Roland Z. Bulirsch. "Direct and indirect methods for trajectory optimization". In: *Annals of Operations Research* 37.1 (1992), pp. 357–373. DOI: 10.1007/bf02071065 (page 13).

[SBD12] Sebastian Sager, Hans Georg Bock, and Moritz Diehl. "The integer approximation error in mixed-integer optimal control". In: *Mathematical Programming* 133.1-2 (2012), pp. 1–23. DOI: 10.1007/s10107-010-0405-3 (pages 4, 70, 73–79).

[SC63] E.N. Sarmin and L.A. Chudov. "On the stability of the numerical integration of systems of ordinary differential equations arising in the use of the straight line method". In: *USSR Computational Mathematics and Mathematical Physics* 3.6 (1963), pp. 1537–1543. DOI: 10.1016/0041-5553(63)90256-8 (page 10).

[Sch+19] Oliver Schütze, Oliver Cuate, Adanay Martín, Sebastian Peitz, and Michael Dellnitz. "Pareto Explorer: a global/local exploration tool for many-objective optimization problems". In: *Engineering Optimization* 52.5 (2019), pp. 832–855. DOI: 10.1080/0305215x.2019.1617286 (pages 50, 151).

[Sch+21] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. "Review on model predictive control: an engineering perspective". In: *The International Journal of Advanced Manufacturing Technology* 117.5–6 (2021), pp. 1327–1349. DOI: 10.1007/s00170-021-07682-3 (pages 2, 17).

[Sch03] O. Schütze. "A New Data Structure for the Nondominance Problem in Multi-objective Optimization". In: *Lecture Notes in Computer Science*.

Springer Berlin Heidelberg, 2003, pp. 509–518. DOI: 10.1007/3-540-36970-8_36 (page 125).

[Sch10] Peter J. Schmid. "Dynamic mode decomposition of numerical and experimental data". In: *Journal of Fluid Mechanics* 656 (2010), pp. 5–28. DOI: 10.1017/S0022112010001217 (page 34).

[Sch12] Stefan Scholtes. *Introduction to Piecewise Differentiable Equations*. Springer Briefs in Optimization. Springer New York, 2012. DOI: 10.1007/978-1-4614-4340-7 (page 135).

[SDD05] Oliver Schütze, Alessandro Dell'Aere, and Michael Dellnitz. "On Continuation Methods for the Numerical Treatment of Multi-Objective Optimization Problems". In: *Practical Approaches to Multi-Objective Optimization*. Dagstuhl Seminar Proceedings 04461. Internationales Begegnungs- und Forschungszentrum f"ur Informatik (IBFI), Schloss Dagstuhl, Germany, 2005. URL: https://drops.dagstuhl.de/opus/volltexte/2005/349/pdf/04461.SchuetzeOliver.Paper.349.pdf (pages 50, 126, 150).

[SDL11] Huan Su, Xiaohua Ding, and Wenxue Li. "Numerical bifurcation control of Mackey–Glass system". In: *Applied Mathematical Modelling* 35.7 (2011), pp. 3460–3472. DOI: 10.1016/j.apm.2011.01.009 (page 93).

[Set12] Burr Settles. "Active learning". In: *Synthesis lectures on artificial intelligence and machine learning* 6.1 (2012), pp. 1–114. DOI: 10.2200/S00429ED1V01Y201207AIM018 (page 22).

[SI18] Carlos Sánchez-Sánchez and Dario Izzo. "Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems". In: *Journal of Guidance, Control, and Dynamics* 41.5 (2018), pp. 1122–1135. DOI: 10.2514/1.G002357 (page 19).

[Sil+16] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961 (pages 2, 22, 25).

[Sil+17] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (2017), pp. 354–359. DOI: 10.1038/nature24270 (pages 2, 22, 25).

[ŠIM21] Marko Švec, Šandor Ileš, and Jadranko Matuško. "Model predictive control of vehicle dynamics based on the Koopman operator with extended dynamic mode decomposition". In: *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. Vol. 1. 2021, pp. 68–73. DOI: 10.1109/ICIT46573.2021.9453623 (page 18).

[SK18] Ozan Sener and Vladlen Koltun. "Multi-Task Learning as Multi-Objective Optimization". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/432aca3a1e345e339f35a30c8f65edce-Paper.pdf (pages 148, 151).

[SM19] Sebastian Scher and Gabriele Messori. "Generalization properties of feed-forward neural networks trained on Lorenz systems". In: *Nonlinear Processes in Geophysics* 26.4 (2019), pp. 381–399. DOI: 10.5194/npg-26-381-2019 (page 92).

[Smi93] John M. Smith. *The theory of evolution*. Cambridge University Press, 1993 (page 50).

[Sut+13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*. Vol. 28. Proceedings of Machine Learning Research. PMLR. 2013, pp. 1139–1147. URL: https://proceedings.mlr.press/v28/sutskever13.html (page 29).

[Tak81] Floris Takens. "Detecting strange attractors in turbulence". In: *Dynamical Systems and Turbulence, Warwick 1980*. Springer Berlin Heidelberg, 1981, pp. 366–381. DOI: 10.1007/BFb0091924 (page 18).

[TBM21] Annalisa T. Taylor, Thomas A. Berrueta, and Todd D. Murphey. "Active learning in robotics: A review of control principles". In: *Mechatronics* 77 (2021), p. 102576. DOI: 10.1016/j.mechatronics.2021.102576 (page 148).

[THR06] Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. "Modeling Human Motion Using Binary Latent Variables". In: *Advances in Neural Information Processing Systems*. Vol. 19. MIT Press, 2006, pp. 1345–1352. URL: https://proceedings.neurips.cc/paper/2006/hash/1091660f3dff84fd648efe31391c5524-Abstract.html (page 57).

[Tib11] Robert Tibshirani. "Regression shrinkage and selection via the lasso: a retrospective". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.3 (2011), pp. 273–282. DOI: 10.1111/j.1467-9868.2011.00771.x (page 104).

[Tib96] Robert Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x (pages 104, 135).

[Tu+14] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. "On Dynamic Mode Decomposition: Theory and Applications". In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421. DOI: 10.3934/jcd.2014.1.391 (page 34).

[VBL13] D. Vidaurre, C. Bielza, and P. Larrañaga. "A survey of L_1 regression". In: *International Statistical Review* 81.3 (2013), pp. 361–387. DOI: 10.1111/insr.12023 (page 103).

[Vla+18] Pantelis R. Vlachas, Wonmin Byeon, Zhong Y. Wan, Themistoklis P. Sapsis, and Petros Koumoutsakos. "Data-driven forecasting of high-dimensional chaotic systems with long short-Term memory networks". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2213 (2018). DOI: 10.1098/rspa.2017.0844 (page 31).

[Vol11] Stefan Volkwein. "Model reduction using proper orthogonal decomposition". In: *Lecture Notes* (2011) (page 84).

[Wan+19] Binyu Wang, Lei Zhou, Zheng Cao, and Jisheng Dai. "Ridge-Adding Approach for SVMpath Singularities". In: *IEEE Access* 7 (2019), pp. 47728–47736. DOI: 10.1109/access.2019.2909297 (page 144).

[Waż63] Tadeusz Ważewski. "On an optimal control problem". In: *Differential equations and their applications* (1963), pp. 229–242 (page 79).

[WB95] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. Tech. rep. USA, 1995 (page 19).

[WKR15] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. "A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition". In: *Journal of Nonlinear Science* 25.6 (2015), pp. 1307–1346. DOI: 10.1007/s00332-015-9258-5 (page 34).

[WRP19] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. *A Survey on Neural Architecture Search*. 2019. DOI: 10.48550/arXiv.1905.01392 (page 66).

[ZH05] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x (page 151).

[Zha+14] Junsheng Zhao, Yongmin Li, Xingjiang Yu, and Xingfang Zhang. "Levenberg-Marquardt Algorithm for Mackey-Glass Chaotic Time Series Prediction". In: *Discrete Dynamics in Nature and Society* 2014 (2014), pp. 1–6. DOI: 10.1155/2014/193758 (page 93).

[Zha17] Lei Zhang. "Artificial neural networks model design of Lorenz chaotic system for EEG pattern recognition and prediction". In: *2017 IEEE Life Sciences Conference (LSC)*. 2017, pp. 39–42. DOI: 10.1109/LSC.2017.8268138 (page 92).

[Zho+11] Aimin Zhou et al. "Multiobjective evolutionary algorithms: A survey of the state of the art". In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 32–49. DOI: 10.1016/j.swevo.2011.03.001 (page 51).

[ZL15] Hua Zhou and Kenneth Lange. "Path following in the exact penalty method of convex programming". In: *Computational Optimization and Applications* 61.3 (2015), pp. 609–634. DOI: 10.1007/s10589-015-9732-x (pages 137, 150).

[ZY04] P. Zhao and B. Yu. *Boosted lasso*. Tech. rep. University of California, Berkeley, 2004 (page 104).