Flat Hybrid Automata for Automation and Control of Hybrid Systems

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Frederik Zahn

Tag der mündlichen Prüfung:	10.11.2025
Hauptreferent:	Prof. Dr. Veit Hagenmeyer
Korreferent:	Prof. Dr.-Ing. Jörg Raisch

Acknowledgment

The research and the writing for this thesis was both exciting and challenging. It feels like I started this endeavor in a different world, so many things have changed in its course, and so have I. I am infinitely grateful for the people who have given me company, and for those who have come into my life.

First and foremost I want to thank Veit Hagenmeyer for giving me his trust, his insight numerous fruitful discussions, his warm and positive presence and his support. I also thank Tobias Kleinert, who developed the idea for this work together with Veit Hagenmeyer, and who was a reliable, curious and enthusiastic collaborator. Furthemore, I would like to thank Jörg Raisch for his great commitment as external reviewer, his valuable feedback, thoughtful questions and discussion, and for ensuring that the final examination took place despite challenging circumstances.

I also want to thank Tillmann Mühlpfordt and Timm Faulwasser, who opened the doors to academic research for me. They taught me many things, held me to high standard and gave me confidence. Tillmann was a great mentor and is a dear friend.

The Institute of Automation and Applied Informatics I many great colleagues and collaborators. I want to thank Lutz Gröll for the challenging discussions, his unrivaled rigor and eye for detail. You taught me many things in your classes and in our interactions. I also want to mention all the members of my group, who have accompanied me through all these fun and difficult moments, and who are friends more than co-workers. Thank you Dorina, Rebecca, Janik, Yanlin, Xinliang, Ben and Adrian, you are great people! I also want to thank Bernadette for all her support and joyful help. I am also grateful to Ralf Mikut for his guidance in many discussions and his thoughtful advice.

I want to express immense gratitude for the help I received in finalizing this thesis. I am extraordinarily grateful to Tessina Scholl for her precise and detailed proof reading and feedback. I also thank my colleagues Janik, Dorina, Yanlin and Rebecca for final proof reading, for being patient and giving me the freedom and time I needed.

Finally, I would like to thank the people closest to me. I thank my parents and my sisters, who always support me and give me the confidence and love I needed. I also have to thank my wonderful wife, Mélanie, who has been by my side through all the highs and the lows. She shows me tough love or tender love, and she always knows when I need the one or the other. During these years we also welcomed our three great children in this world. Matthieu, Louise and Pauline, you remind me everyday what life is about and you give me endless joy and wonder.

Abstract

Differential flatness and hybrid systems are established concepts in the automation and control of dynamic systems. The system property of differential flatness allows for greatly simplified control of linear and non-linear systems. Hybrid systems, on the other hand, are a highly expressive modeling approach due to the combination of discrete and continuous variables. Within this area, Hybrid Automata (HA) are considered the most fundamental model class, as they can represent almost all other hybrid system models. However, the combination of the expressiveness of HA with the simplification offered by differential flatness has not yet been studied and formalized. In this thesis, we therefore address the questions of how the concept of flatness can be applied to HA and what possibilities and limitations this approach entails.

In the first part of the thesis, we present the fundamental theory and define the relevant objects and concepts. First, we extend the concept of differential flatness to apply it to finite automata. To this end, we use the definition of flatness from behavioral system theory, which is based on equivalence relations of trajectories. Fundamental to the flatness of automata are a strongly connected automaton graph and the controllability of events. Based on these considerations, we propose a definition of flatness for HA, which we refer to as flat hybrid automata (FHA). The definition comprises three fundamental properties: flatness of continuous dynamics, a strongly connected automaton graph, and the reachability of guard sets. These properties ensure that all paths through the automaton can be realized.

The reachability of guard sets is a property that is not easy to demonstrate. We therefore investigate both theoretical properties and numerical methods to determine the reachability of switching conditions. We use the linearized representation of flat dynamics to derive sufficient conditions for reachability. We use optimization algorithms for the numerical determination of reachability.

In the last part of the thesis, we apply the results to a total of four example systems. We present examples of hybrid system models from electrical engineering, mechatronics, and process engineering. For these systems, we demonstrate the modeling as FHA in detail and check their flatness properties. We then apply feedforward control and optimized trajectory planning to these systems. In doing so, we examine the specific properties and possibilities of the presented FHA model with respect to these control approaches.

Deutsche Kurzfassung

Differenzielle Flachheit und hybride Systeme sind etablierte Konzepte in der Automatisierung und Regelung dynamischer Systeme. Die Systemeigenschaft der differenziellen Flachheit erlaubt eine stark vereinfachte Steuerung linearer und nichtlinearer Systeme. Hybride Systeme sind dagegen ein Modellierungsansatz, welcher durch die Verbindung diskreter und kontinuierlicher Größen große Ausdrucksstärke besitzt. Als grundlegendste Modellklasse gelten in diesem Bereich Hybride Automaten (HA), welche nahezu alle anderen hybriden Systemmodelle darstellen können. Die Verbindung der Ausdrucksstärke hybrider Systemmodelle mit der Vereinfachung durch differenzielle Flachheit ist die grundlegende Idee der vorliegenden Arbeit. Wir widmen uns in dieser Arbeit also den Fragen, wie das Konzept der Flachheit auf hybride Automaten angewandt werden kann und welche Möglichkeiten und Grenzen dieser Ansatz mit sich bringt.

In der vorliegenden Arbeit stellen wir zunächst die existierenden grundlegenden Theorien dar und definieren die relevanten Objekte und Konzepte. Im Hauptteil der Arbeit erweitern wir das Konzept der differenziellen Flachheit auf endliche Automaten. Dabei nutzen wir die Flachheitsdefinition aus der sogenannten Behavioral Systemtheorie, welche sich auf Äquivalenzbeziehungen von Trajektorien stützt. Aus dieser Perspektive kommen wir zu einer Flachheitsdefinition für Automaten, für welche ein stark zusammenhängender Automatengraph und Steuerbarkeit grundlegende Eigenschaften des Automaten sind. Aufbauend auf diesen Überlegungen kommen wir zu einer Flachheitsdefinition für HA, welche wir als Flache Hybride Automaten (FHA) bezeichnen. Die Definition umfasst drei grundlegende Eigenschaften: Flachheit der kontinuierlichen Dynamiken, ein stark zusammenhängender Automatengraph und die Erreichbarkeit von Schaltbedingungen. Durch diese Eigenschaften wird sichergestellt, dass alle Pfade durch den Automaten realisiert werden können.

Unter diesen Eigenschaften ist speziell die Erreichbarkeit der Schaltbedingungen nicht einfach zu zeigen. Wir untersuchen deshalb sowohl theoretische Kriterien als auch numerische Verfahren, um die Erreichbarkeit von Schaltbedingungen zu bestimmen. Dabei nutzen wir die linearisierte Darstellung der flachen Dynamiken zur Herleitung hinreichender Bedingungen für Erreichbarkeit. Für die numerische Bestimmung der Erreichbarkeit nutzen wir Optimierungsalgorithmen.

Im letzten Teil der Arbeit wenden wir die Ergebnisse auf vier Beispiele schaltender Systeme aus der Elektrotechnik, der Mechatronik und der Prozesstechnik an. Für diese Systeme zeigen wir detailliert die Modellierung als FHA und überprüfen deren Flachheitseigenschaften. Anschließend wenden wir Feedforward Steuerung und optimierte Trajektorienplanung auf diese Systeme an. Dabei untersuchen wir die spezifischen Eigenschaften und Möglichkeiten des vorgestellten FHA Modells.

Inhaltsverzeichnis

Acknowledgment . . . . **i**

Abstract . . . . **iii**

Deutsche Kurzfassung . . . . **v**

Notation . . . . **xi**

1 Introduction **1**
1.1 Related Work 3
1.2 Content and Contributions 5

2 Continuous Dynamical Systems and Differential Flatness . . **11**
2.1 Reachability and Controllability 14
2.2 Differential Flatness 16
2.2.1 Trajectories and Inversion 18
2.2.2 Equivalence to Linear System 19
2.2.3 The Behavioral Systems Theory Perspective 21
2.2.4 Computing Flatness 23

3 Hybrid Automata **25**
3.1 Elements of Hybrid Automata 26
3.1.1 Automaton 26
3.1.2 Transitions 27
3.1.3 Inputs 29
3.2 Hybrid Automaton Model 29
3.3 Solutions and Trajectories 33
3.4 Reachability and Controllability 35

4 Flat Hybrid Automata **39**
4.1 Automaton Flatness 39
4.2 Flatness of Hybrid Automata 43
4.2.1 Automaton 44
4.2.2 Continuous Dynamics 44
4.2.3 Switching Dynamics 46
4.3 Flat Hybrid Automata 47
4.4 Flat Hybrid Automaton Design 48
4.4.1 Connectedness of Automaton Graph 49
4.4.2 Flatness of Flow Set 50
4.5 Flat Hybrid Automaton Reachability and Controllability 51
4.6 Summary and Discussion 52

5 Reachability of Guard Sets **55**
5.1 Steady State Transitions 58
5.2 Non-Steady Initial and Final States 61
5.3 Numerical Reachability Computation 64
5.3.1 Bézier Curves 65
5.3.2 Exact Discretization 69
5.3.3 Example 72
5.3.4 Discussion of the Results 75
5.4 Summary and Discussion 77

6 FHA Modeling and Control **79**
6.1 Modeling of Systems as FHA 79
6.1.1 Electrical DC Switched Circuit 80
6.1.2 Tank Example 87
6.1.3 1 Degree-of-Freedom Robotic Thrower 93
6.2 FHA Inversion, Feedforward and Control 99
6.2.1 Application to Tank Example 101
6.2.2 Application to Transmission Line Switching 103
6.3 Summary and Discussion 110

7 Optimization-Based Trajectory Generation and Control for FHA **113**
7.1 Optimization Problem 116

7.2 Decoupled Path and Trajectory Computation 118
7.2.1 Example: 1-DoF Robot Thrower 120
7.2.2 Discussion 124
7.3 Integrated Path and Trajectory Computation 125
7.4 Example: Dynamic Optimal Transmission Switching 125
7.5 Summary and Discussion 132

8 Summary and Outlook 135

List of Publications 145

Bibliography 147

Notation

List of Symbols

Numbers and Functions

$\mathbb{N}$	natural numbers
$\mathbb{Z}$	integers
$\mathbb{R}, \mathbb{R}_{\geq 0}$	real numbers, non-negative real numbers
$\mathbb{R}^n$	n-dimensional Euclidean space
$\mathcal{T}$	time domain
$PC(\mathcal{T}, \mathbb{R}^n)$	piecewise continuous $\mathbb{R}^n$-valued functions on $\mathcal{T}$
$AC(\mathcal{T}, \mathbb{R}^n)$	absolutely continuous piecewise-continuous $\mathbb{R}^n$-valued functions on $\mathcal{T}$
C^γ	functions with $\gamma \in \mathbb{N}$ continuous time derivatives
$A > 0$	element wise relation for matrix A, i.e., each entry of A is greater than zero

Hybrid System Notation

x	continuous state
u	continuous input

x^0, x^f	initial and final states
l	location or vertex of an automaton
v	discrete input
e	transition or edge of an automaton
x^-, x^+	states before and after a transition
$L_\mathsf{e}(x^-)$	reset function, mapping states x^- to x^+ for transition e
$\mathcal{G}^x_\mathsf{e}$	guard set of transition e caused by the continuous state x
$\mathcal{G}^\mathsf{v}_\mathsf{e}$	guard set of transition e caused by the discrete input v
$\textbf{Arrive}_\mathsf{l}$	set of arrival states of location l
Flow	set of continuous evolution of x in location l
$(\mathsf{l}^j)_j$	short form notation of a finite ordered sequence with $j \in 0, \ldots, n$ for some $n \in \mathbb{N}$

Acronyms

HA	Hybrid Automaton or Hybrid Automata
ODE	Ordinary Differential Equation
FHA	Flat Hybrid Automaton or Flat Hybrid Automata
MLD	Mixed Logical Dynamical (system)
DC	Direct Current
AC	Alternating Current
HS	Hybrid System
MPC	Model Predictive Control

MIP	Mixed Integer Program
MILP	Mixed Integer Linear Program
MICP	Mixed Integer Convex Program
MINLP	Mixed Integer Non-linear Program
MIQP	Mixed Integer Quadratic Program
LP	Linear Program
QP	Quadratic Program
NLP	Non-linear Program
OTS	Optimal Transmission Switching
OPF	Optimal Power Flow

1 Introduction

In practice, the dynamics of many engineered systems can be described as hybrid dynamics, characterized by a mix of discrete and continuous variables. The hybrid nature can originate from physical phenomena such as switching elements, discontinuities, or changes in operation mode. Furthermore, many automated processes or tasks can be modeled as a series of continuous subparts, which can be described separately and concatenated to form different versions of the process. Additionally, the ever-rising interconnection in modern large-scale systems, for example, electrical grids or automated production sites, also results in systems that consist of many different model types with changing configurations. Inside the diverse landscape of hybrid system models, Hybrid Automata (HA) are a model class with enormous modeling power from the combination of continuous Ordinary Differential Equations (ODE) and finite automata. This combination could be expected to make them the predominant model type in automation applications. However, HA currently find themselves in a challenging position with respect to automation and control: Although they are a powerful model class with an important body of literature to their name, they remain largely irrelevant for actual applications. While HA have the potential to model a large variety of systems and dynamics, their capability to combine both paradigms comes with a cost: The resulting models and dynamics can be quite complex, and the mathematical tools and control approaches from each domain cannot be easily applied, if they can be applied at all.

The situation of HA is comparable to the class of non-linear ODE before the 1990s, when the model class was well established with elaborate mathematical theory, but with little significance in industrial applications. Today, however, non-linear ODE play an important role in many automation and control applications. One of the significant breakthroughs in this field was the introduction of differential flatness by Fliess and his co-authors in a series of seminal papers (Fliess et al. 1995, 1999a). Trajectories of differentially flat systems can be expressed in terms of their so-called flat outputs. The corresponding inputs, as well as the system states, can be directly computed from the flat output trajectories. Furthermore, feasible trajectories of differentially flat systems can be found without integrating the system dynamics. Differential flatness theory looks at dynamical systems in terms of their trajectories, and it allows for steering systems by feedforward. Important use cases are offline trajectory computations (Greco et al. 2022), model predictive control (Faulwasser et al. 2014, Greeff and Schoellig 2018) and feedforward linearization (Hagenmeyer and Delaleau 2003a,b).

Flatness-based control has also made an impact in many industrial applications, mainly in the form of two-degree-of-freedom control (Hagenmeyer and Zeitz 2004). In the literature, we can find examples of these applications either in advanced experimental settings or in even real industrial environments. These include control of clutches (Horn et al. 2003, Mrochen and Sawodny 2019), electrical drives (Henke et al. 2014, Stumper et al. 2015) and batch reactors (Hagenmeyer and Nohr 2008). In a series of articles by Hofmann and Raisch (2012, 2013), control of a batch crystallization process based on flatness theory is proposed, although the system at hand is not flat. Further examples of applications of flatness-based control are a hydraulic actuator (Bindel et al. 2000), a pumped storage power plant (Treuer et al. 2011), liquid metal pouring (Noda et al. 2011), a gantry crane (Petit et al. 2001), an elastomer actuator (Scherer et al. 2020), and control of an electrolyzer (Keller et al. 2025). Another important field using the two-degree-of-freedom control is robotics, where inverse dynamics and inverse kinematics are the basis of most control schemes Spong et al. (2020). Although the inverse dynamics used in robotics have not been derived from flatness theory historically, they are an expression of the differential flatness of the systems at hand.

In that light, the observation that HA are of limited relevance in industrial applications, together with their inherent complexity, raises an obvious question: Can we apply the concept of differential flatness to HA to reduce their complexity and to make them more usable in practice? Answering this question is at the core of the present thesis.

1.1 Related Work

The introduction of hybrid systems to the scientific literature is often credited to Witsenhausen and his article Witsenhausen (1966). The topic gained significant interest in the 1990s, leading to a high number of publications and important results, and reaching its peak in the early 2000s. In this period, various hybrid system models were formalized and analyzed, in the hope of unifying the theory of discrete dynamical systems, predominant in computer science, and of continuous dynamical systems from automation and control. Combining discrete and continuous variables and dynamics can lead to many different system structures, depending on the presence and nature of inputs, linearity, and the type and structure of the transitions. The variety of different system definitions and formalisms in the literature is quite extensive. A few rather well-defined classes include Mixed Logical Dynamical (MLD) systems (Bemporad and Morari 1999), piecewise affine systems (Bemporad et al. 2000), and timed automata (Alur and Dill 1994). These formalisms are all restricted to linear continuous dynamics, either in discrete or continuous time. Another important class is the class of switched systems, also called switching systems. Under this term, multiple different system types are discussed in the literature, mainly with linear continuous dynamics and without continuous inputs. One of the most general hybrid system model classes is hybrid automata (Alur et al. 1993, Branicky et al. 1998), which can model all of those mentioned above. For an overview of these system classes, we refer to the collection published by Lunze and Lamnabhi-Lagarrigue (2009). An even more general view on hybrid systems was formalized as hybrid inclusions (Goebel et al. 2009). This framework is arguably the most abstract way of describing hybrid systems, requiring intricate mathematical tools.

Not only are there many different hybrid system models, but there are also different research directions investigating hybrid systems. We provide a concise overview of key topics in the automation and control domain. One group of research topics is grounded in a control and dynamical system theory perspective, aiming at mathematical characterizations of solution concepts (Sanfelice et al. 2007), stability (Goebel et al. 2009), observability and observer design (Bemporad et al. 2000, Bernard and Sanfelice 2024, 2022), controllability (Lemch et al. 2000, Bemporad et al. 2000, van Schuppen 1998), and other properties. Although these results are important contribution to hybrid system theory, actual applications of these results in the literature or in practice are rare.

In the field of computer science, the currently biggest field of hybrid system research is hybrid system verification. In this field, hybrid systems are also called cyber-physical systems (Platzer 2018). Two important types of approaches in this field are methods for computing reachable sets (Althoff et al. 2010, 2021) and verifying safe operation and robustness of hybrid systems, see e.g. Platzer (2010). These methods mainly consider autonomous or controlled hybrid systems without inputs.

Although applications of hybrid system models and techniques are not standard in industrial settings, there are some application examples in the literature. While a comprehensive list of hybrid system examples is beyond the scope of this work, we provide a few examples of HA applications. Examples of HA models in the literature include applications in mobile robots (Egerstedt 2000, Egerstedt and Hu 2002, Buss and Schlegl 2000, Nenchev et al. 2015) and dexterous manipulation (Xu and Li 2008). HA models are also reported in applications to power electronics (Sreekumar and Agarwal 2008, Sira-Ramirez and Silva-Ortigoza 2002) and electrical grids (Susuki et al. 2012, Fourlas et al. 2004). Furthermore, HA modeling and control are applied to the modeling of chemical processes (Kleinert and Lunze 2002, Engell et al. 2000, Raisch and Moor 2005).

Control approaches for HA documented in the literature often rely on discretizations or discrete abstractions of the hybrid dynamics. These have been used, for example, for supervisory control (Koutsoukos et al. 2000, Raisch and O'Young 1998, Moor and Raisch 1999) and for optimization schemes based on dynamic programming (Hedlund and Rantzer 2002, Rungger and Stursberg 2011). The discrete abstractions allow for tractable computation of solutions, but suffer from the curse of dimensionality, and they inevitably lose information about the continuous dynamics. Other approaches are based on optimal control theory, see e.g. Tomlin et al. (2000), Lygeros et al. (1997), Shaikh and Caines (2003). These face difficulties in computation, because the solutions to the problems are generally hard to obtain. An exception is the subclass of linear discrete-time models, for which Mixed-Integer Linear Programs (MILP) can be formulated, as demonstrated by Bemporad and Morari (1999), which has even found its way into industrial applications (see Torrisi and Bemporad (2004) and references therein).

Combining differential flatness and hybrid systems is not an entirely new idea. Differential flatness for switched systems with linear discrete-time dynamics is formalized in Millerioux and Daafouz (2007, 2009), and the authors successfully apply feedforward in examples. There are also applications of flatness to specific hybrid system applications in Sreenath et al. (2013) and Rouchon and Sira-Ramirez (2003). The work by Sreenath et al. (2013) applies flatness to model and control a drone with a cable-suspended load. Although the authors call their model a flat hybrid system, they do not provide any definition of such a system. In Rouchon and Sira-Ramirez (2003), the authors apply differential flatness to a walking toy, demonstrating the potential of feedforward to hybrid dynamics. All these works show promising combinations of hybrid systems and differential flatness. However, none of them formalizes flatness for the system class of HA.

1.2 Content and Contributions

The basic idea of the present thesis is fairly simple: We apply the concept of differential flatness to the system class of HA. Our aim is to investigate how and to what extent differential flatness theory can be extended to this system class. To this end, we take a trajectory perspective on HA: Starting from properties of their constituent parts, we want to construct a usable notion of the set of feasible hybrid trajectories. The approach we present in this thesis is aimed at systems that are designed in a way that either their continuous dynamics are flat, or that can be modeled approximately as flat systems. Thus, we are only interested in systems with inputs. Furthermore, our approach is distinctly focused on engineering systems, i.e., systems that are purposefully designed to have specific properties. Therefore, we take two complementary perspectives: We present the conditions and properties for flatness of HA, and describe their modeling and design implications, both in theory and through examples. The main contributions of this work are

- **Definitions of flatness for automata and HA:** We define flat HA as a class of hybrid systems, based on a set of properties of their constituent parts. This includes a definition of flatness for automata and a careful analysis of the reachability of transitions.
- **Tests and conditions to assess flatness of HA:** We analyze how to determine whether a given HA is flat according to our definition. A significant advantage of our definition is that the properties can be assessed separately for the continuous dynamics and the automaton, making the computation tractable.
- **Application of the theory to multiple examples:** We apply the modeling approach and the definition of flat HA to several examples from different engineering domains. Thereby, we demonstrate the modeling of systems as FHA.
- **Approaches for optimization-based trajectory generation and feedforward for flat HA:** We apply trajectory computation based on numerical optimization and feedforward to FHA examples. We also analyze the complexity of optimization-based trajectory computation for different types of trajectories, i.e., with different types of dynamics, constraints, and with or without path computations on the graph.

Parts of the present thesis are already published elsewhere. Other parts of the material have not been published yet. We indicate in the main text which parts have been published, except in Chapters 2 and 3, which introduce notation and definitions.

In the following, we give an overview of the contents of each chapter.

Chapter 2 – Continuous Dynamical Systems and Differential Flatness

The first part of the chapter introduces and defines the necessary concepts and notations from continuous dynamical systems theory. We define the basic continuous-time ordinary differential equation model used, along with notions of solutions and trajectories. Furthermore, we define the properties of reachability and controllability. In the second part, we give an introduction to differential flatness theory and an overview of important properties of flat systems. We also introduce the definition of flatness in the behavioral systems theory framework, which is necessary for defining flatness for automata and HA.

Chapter 3 – Hybrid Automata

In the hybrid system literature, different versions of HA models exist that vary in their elements and definition. Therefore, we begin the chapter by explaining the different elements and introducing the necessary notation. We define a HA model that we use in the subsequent developments, and we discuss and define notions of trajectories, reachability, and controllability.

Chapter 4 – Flat Hybrid Automata

The main part of this chapter is the definition of flatness for HA. Because of the mixed discrete and continuous dynamics, we cannot simply apply the classic definitions of flatness to HA. Therefore, we develop our definition by looking at the constituent parts of the HA and finding appropriate conditions for flatness. In a first step, we define flatness for automata based on the behavioral definition of flatness. Furthermore, we examine the continuous dynamics and transitions.

Based on the results of our investigations, we then synthesize a definition of flatness for HA based on three properties. We call the resulting model class Flat Hybrid Automata (FHA). At the end of the chapter, we evaluate the implications of our flatness definition for reachability and controllability.

Chapter 5 – Reachability of Guard Sets

One property of FHA is that all guard sets triggering transitions have to be reachable. This property guarantees reachability of all locations, but can be difficult to assess. However, the flatness property of the continuous dynamics allows for significant simplifications in this assessment. In this chapter, we therefore discuss the problem and develop necessary and sufficient conditions to guarantee reachability. These conditions only apply under certain assumptions, which are not true for all possible FHA. This motivates the second part of the chapter, in which we present two numerical approaches to evaluate reachability based on lightweight optimization problems. While the first approach is based on Bézier curve representations of the flat output trajectories, the second approach is based on exact discretization. We demonstrate the effectiveness of the approaches for an academic example.

Chapter 6 – FHA Modeling and Control

In this chapter, we apply our theory to four examples from three different engineering domains: We present two switched electrical network examples from electrical engineering, a tank example from process engineering, and an example with a robotic manipulator from mechatronics. For these examples, we explain in detail how the systems can be modeled as HA, and show their flatness property. Furthermore, we investigate feedforward and inversion for FHA in general, and demonstrate feedforward control for two of the examples.

Chapter 7 – Optimization-based Modeling and Control for FHA

Trajectory computation frequently relies on numerical optimization methods. For hybrid trajectories, these computations are often computationally expensive because of the combination of continuous and discrete decision variables. In this chapter, we show how the flatness property can simplify these computations for FHA. More specifically, the flatness property of FHA enables the separation of the discrete variables and the continuous variables for certain tasks, which we demonstrate both in theory and for the robotic manipulator example from the previous chapter. Furthermore, we discuss the possible simplifications that FHA can offer, even if both variable types are included, and we describe the limitations. We finish the chapter with a final example for transmission line switching in electrical grids.

Chapter 8 – Summary and Outlook

The thesis is concluded with a summary, including a discussion of open problems and future research directions.

2 Continuous Dynamical Systems and Differential Flatness

Continuous dynamical systems in the form of ordinary differential equations are at the basis of this thesis in two important ways: First, they are one of the two basic ingredients of HA. We thus have to properly define the necessary notation and concepts as a basis for our developments. Second, the concept of differential flatness was developed for these systems, and they remain the principal domain of its application. Our work extends its area of application to HA, and, therefore, we need a good understanding of the definition and implications of differential flatness. In the following, we describe our notation for the continuous dynamics, and we define the necessary properties and concepts.

We consider continuous-time time-invariant systems with inputs in state-space representation described by[1]

$$\dot{x} = f(x, u), \quad x(0) = x^0 \tag{2.1}$$

and we consider piecewise continuous input functions

$$u \in PC(\mathcal{T}, \mathbb{R}^m) \tag{2.2}$$

on a time interval $\mathcal{T} = [0, t^f) \subseteq \mathbb{R}_{\geq 0}$. The state vector $x(t) \in \mathbb{R}^n$ and the input vector $u(t) \in \mathbb{R}^m$ are the system variables at time t, $\dot{x}$ denotes the derivative

[1] Contrary to many publications in control theory, we do not introduce a system output y because we only consider the states and the flat output for our approach. Thus, we assume that the state of the system is known and can be measured.

with respect to time, i.e., $\frac{\mathrm{d}x}{\mathrm{d}t}$. An initial condition is denoted by $x(0) = x^0$. The dynamics of the system is defined by the vector field $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. In the present work, we are mainly interested in local properties of ODEs in a given set $\mathcal{X} \subseteq \mathbb{R}^n$ and for a given set of inputs $\mathcal{U} \subseteq \mathbb{R}^m$. We thus restrict our investigation to states $x(t) \in \mathcal{X}$ and inputs $u(t) \in \mathcal{U}$. To ensure well-posedness, i.e., existence and uniqueness of solutions as well as their continuous dependence on parameters and initial data, of (2.1) with piecewise continuous input functions $u(\cdot)$, we assume that f is locally Lipschitz in both arguments (Khalil 2002).

We define a trajectory of (2.1) on an interval $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$ as a pair of functions[2] (x^*, u^*) with absolutely continuous $x^* \in AC(\mathcal{T}, \mathbb{R}^n)$ and piecewise continuous $u^* \in PC(\mathcal{T}, \mathbb{R}^m)$ that satisfy

$$x^*(t) = x(0) + \int_0^t f(x^*(\tau), u^*(\tau))\mathrm{d}\tau, \quad \forall t \in \mathcal{T}. \tag{2.3}$$

We use the superscripts $\cdot^0$ and $\cdot^f$ to denote initial and final states, respectively. To characterize trajectories of system (2.1) in a concise notation, we introduce the map

$$x^*(t) = \phi_f(t, x^0, u^*), \quad \forall t \in \mathcal{T}, \tag{2.4}$$

where $\phi_f : \mathcal{T} \times \mathbb{R}^n \times PC(\mathcal{T}, \mathbb{R}^m) \to \mathbb{R}^n$.

Linear time-invariant systems with finite dimension and with inputs are a very important subclass of systems in the systems and control literature. An ODE (2.1) with a linear vector field can be put into a standard form

$$\dot{x} = Ax + Bu, \tag{2.5}$$

with $x(t) \in \mathbb{R}^n, A \in \mathbb{R}^{n\times n}, u(t) \in \mathbb{R}^m$, and, $B \in \mathbb{R}^{n\times m}$.

[2] On some occasions, we also call a function x^* a trajectory, assuming that the corresponding inputs exist and can be computed.

A special case of a linear system is a chain of integrators, sometimes also called *trivial system* (Fliess et al. (1999a)). Such a system consists of m independent integrator chains of length $\kappa_i \in \mathbb{N}_{\geq 0}$

$$\zeta_i^{(\kappa_i)} = u_i, \qquad i = 1, \ldots, m, \tag{2.6}$$

i.e., for the m scalar variables ζ_i, there is no relation between their derivatives of any order. This type of system plays an important role in differential flatness theory. A classic example of such a system is the dynamic equation of a moving particle, derived from Newton's second law: In the one-dimensional setting, the acceleration $\ddot{x}$ of a particle with mass m is given by the force acting on it. Considering the force divided by the mass as the input, we obtain

$$\ddot{x} = \frac{1}{m} F = u. \tag{2.7}$$

The particle can follow any two times continuously differentiable trajectory x^* consistent with an initial condition x^0. We will get back to such systems in the section on differential flatness.

For unconstrained linear systems, properties are global, whereas properties of non-linear systems can only be studied locally in many cases. This fact is one of the reasons for the difference in complexity of these two system classes. Another reason is that any properties of linear systems can be analyzed in terms of the matrices A and B, whereas non-linear systems can consist of a large variety of functions. For non-linear systems, the analysis is generally much more involved and mostly restricted to local properties. In the following section, we define some of these properties, which are important for our application of differential flatness to HA.

2.1 Reachability and Controllability

The concepts of reachability and controllability define a relation between the sets of the initial and final values in terms of feasible trajectories. There are different definitions for these concepts in the literature. The following definitions are based on the theory and concepts in Sontag (1998) and Levine (2009).

When we want to compute a trajectory from a state x^0 to a state x^f, we first want to make sure that x^f is reachable from x^0, meaning that a feasible trajectory between them exists. Whether a trajectory exists depends on the system dynamics and the sets $\mathcal{X}$ and $\mathcal{U}$. The corresponding property of the two states and the system is called reachability:[3]

Definition 1 (Reachability, (adapted from Levine (2009), Definition 4.5)) *For a system $\dot{x} = f(x, u)$, a state $x^f \in \mathcal{X}$ is reachable from $x^0 \in \mathcal{X}$, if there exist a $t^f \in \mathbb{R}_{>0}$ and a $u^*(\cdot)$, such that $\phi_f(t^f, x^0, u^*) = x^f$ and $\phi_f(t, x^0, u^*) \in \mathcal{X}$ for all $t \in [0, t^f]$.* □

Reachability is frequently studied in terms of *reachable sets*, see i.e. Levine (2009), Sontag (1998). The reachable set is defined as the set of all states that are reachable from an initial state x^0 in time t. In our work, we do not need to characterize the reachable set, but in the context of guard sets of HA, we want to compute whether a guard set can be reached. In this case, we only have to make sure that at least one point in a guard set can be reached from everywhere in the given set of initial states. Thus, we say that a set $\mathcal{X}^f \subseteq \mathcal{X}$ is reachable from $\mathcal{X}^0 \subseteq \mathcal{X}$, when for all $x^0 \in \mathcal{X}^0$, there is at least one reachable $x^f \in \mathcal{X}^f$. We use this relation in Chapter 5 to evaluate reachability of guard sets.

The property of reachability can be generalized to the system property controllability:

[3] In the literature, reachability is often defined as a system property, see e.g. Levine (2009), Sontag (1998). In our work, we are not interested in the system property, but the reachability of specific pairs of states or sets.

Definition 2 (Controllability) *A system $\dot{x} = f(x, u)$ is controllable if for all pairs $x^0, x^f \in \mathcal{X}$, the state x^f is reachable from x^0.* □

For non-linear systems, controllability can generally be studied only locally. For linear systems of the form (2.5), controllability can be proven globally, if the state and input space are unconstrained, i.e., $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{U} = \mathbb{R}^m$. In that setting, the rank of Kalman's controllability matrix reveals whether the system is controllable:

Theorem 1 (Controllability of linear systems, Sontag (1998), Theorem 3) *A linear system $\dot{x} = Ax + Bu$ is controllable, if the controllability matrix*

$$R(A, B) = [B, AB, A^2B, \dots, A^{n-1}B]$$

has rank n. □

The controllability property of linear systems is only valid for unconstrained systems, because it only guarantees the existence of trajectories without considering constraints. For constrained systems, we would additionally need the property that all trajectories stay in the constrained state space. We will discuss this problem in later sections.

Both controllability and reachability of non-linear systems can rarely be proven globally. There are local tests that can determine these properties in neighborhoods, but there are no global equivalences to the linear system characterizations. In the present thesis, we do not need these tests and properties. We refer to, e.g., Levine (2009) for an introduction and references on that topic.

For a general non-linear system, the knowledge of its reachable set does not imply knowledge about a corresponding feasible trajectory, other than that it exists. To find a trajectory, we still have to find a solution to the differential equation. Differential flatness goes beyond characterizing the reachable sets by parameterizing all feasible trajectories, as we see in the next section.

2.2 Differential Flatness

Differential flatness allows us to express the dynamics of a dynamical system in terms of a so-called flat output z. This flat output is defined as a function of the system state x and the input u, and possibly derivatives of the input. A classical definition of differential flatness for a continuous-time ODE, as defined by (2.1) is the following:

Definition 3 (Differential Flatness (Levine (2009), Definition 6.1)) *A system $\dot{x} = f(x, u)$ with $x \in \mathcal{X} \subseteq \mathbb{R}^n, u \in \mathcal{U} \subseteq \mathbb{R}^m$ is differentially flat if there exist two integers β and γ, and smooth functions F, Φ, Ψ with*

$$z = F(x, u, \dot{u}, \ddot{u}, \dots, u^{(\beta)}), \tag{2.8a}$$

$$x = \Phi(z, \dot{z}, \ddot{z}, \dots, z^{(\gamma-1)}), \tag{2.8b}$$

$$u = \Psi(z, \dot{z}, \ddot{z}, \dots, z^{(\gamma)}), \tag{2.8c}$$

which define the so-called flat output z in an open set $\mathcal{Z} \subseteq \mathbb{R}^m$ (2.8a). *The components of z are differentially independent.*[4] *The superscripts $(\beta), (\gamma)$ denote the β-th and the γ-th time derivative. F, Φ, Ψ are smooth functions at least in a suitably chosen open subset.* □

The most useful property of a differentially flat system can be directly seen in this definition: For any trajectory $z^*(t)$ of class $C^{\gamma-1}$, we can directly compute the states $x^*(t)$ and the corresponding inputs $u^*(t)$.

Definition 3 expresses the flatness property in terms of the functions F, Φ, Ψ. It is a common way to define flatness, but there are several other definitions. Originally, differential flatness was defined in a differential algebraic setting in Fliess and Glad (1993), Fliess et al. (1995). Subsequently, an equivalent definition in a differential geometric framework was given in Fliess et al. (1999a). Both

[4] The differential independence of the flat outputs can also be replaced by the condition that the dimension of the flat output has to be the same as the number of independent inputs m (Fliess et al. 1999b).

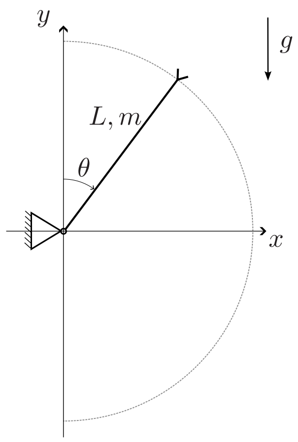


Abbildung 2.1: Schema of a planar one-degree-of-freedom robotic manipulator.

formulations are built on specific concepts from their respective fields. As we do not need these concepts for our work, we refer the interested reader to the sources mentioned above.

The flat output of a given differentially flat system is not unique. Therefore, it is essential to state the flat output explicitly. For many non-linear flat systems, flatness is not a global property (see, for example, Fliess et al. (1999a)). One reason for that is the possible presence of singularities in the maps (2.8). We thus often say that a system is locally flat in a certain set, which can be either expressed in the flat output $z, \dot{z}, \ddot{z}, \ldots$, or in the state variable x. For a linear system (2.5), differential flatness is equivalent to controllability: A linear system is differentially flat if and only if it is controllable (Levine (2009), Corollary 6.1).

Example 1 *We apply the flatness definition to a simplified planar one-degree-of-freedom robotic manipulator depicted in Figure 2.1. We model the dynamics of the manipulator input $u = \tau$ and state $x = \theta$ as*

$$L^2 m\ddot{\theta} = mgL\sin(\theta) - \tau. \tag{2.9}$$

The input is the actuator torque τ, and the other variables are gravity g, length L, mass m and angle θ. We assume that all the mass of the manipulator is concentrated at its end. These non-linear dynamics are differentially flat with flat output $z = \theta$. Thus, the function (2.8b) *is simply* $\Phi(z) = z$*, and* (2.8c) *is*

$$\Psi(z, \ddot{z}) = mL(g\sin(z) - L\ddot{z}). \tag{2.10}$$

In the robotics literature, these relations are well-known under the term inverse dynamics, see Spong et al. (2020).

2.2.1 Trajectories and Inversion

In systems and control theory, the inversion of a controlled system is a central concept. For a continuous-time dynamical system in state-space form (2.1), we can calculate a solution x^* for a given initial value x^0 and a given input u^*. This formulation of a system is captured in (2.4), $x^*(\cdot) = \phi_f(\cdot, x^0, u^*)$. The inversion of the system is a mapping, which computes the input u^* for a given trajectory x^* with initial value x^0.[5] For both expressions ϕ_f and its inversion to u^*, the system model generally does not provide explicit representations. Computing x^* from $x(0)$ and u^* is often numerically straightforward using appropriate numerical integration schemes. For linear systems and for some simple non-linear systems, explicit solutions can be computed by hand or by numerical algebra solvers. However, for a general system (2.1), no universal method exists to compute u^*.

For a differentially flat system $\dot{x} = f(x, u)$ and any trajectory $z^* \in C^{\gamma-1}([t^0, t^f], \mathcal{Z})$ with $\mathcal{Z}$ and γ from Definition 3 with consistent initial condition

$$x(t^0) = \Phi(z^*(t^0), \dot{z}^*(t^0), \ddot{z}^*(t^0), \dots, z^{*(\gamma-1)}(t^0)), \tag{2.11}$$

[5] Note that this is not an inverse mapping in the mathematical sense, because the dimensions of the domain and the co-domain are different.

the inputs u^* and the states x^* can be explicitly calculated from Equations (2.8b) and (2.8c) without integrating differential equations, see (Fliess et al. (1995), Hagenmeyer and Delaleau (2003a)). Thus, differentially flat systems are invertible by definition with respect to their flat output and its derivatives. Studying systems in terms of their flat output restricts properties of potential trajectories z^*: According to Equations (2.8), they have to be of class $C^{\gamma-1}$.

Example 1 (continued) *We go back to the robotic manipulator example. Given an initial condition $\theta^0, \ddot{\theta}^0$, we can compute feasible trajectories and the inversion to compute the corresponding inputs:*

$$\tau^* = mL(g\sin(\theta^*) - L\ddot{\theta}^*). \tag{2.12}$$

We can directly see that any two-time differentiable function θ^* with $\theta^*(0) = \theta^0$, $\ddot{\theta}^*(0) = \ddot{\theta}^0$ is a feasible trajectory, to which we can compute the corresponding inputs.

2.2.2 Equivalence to Linear System

The dynamics of a differentially flat system are equivalent to those of a trivial system in terms of trajectories: For every differentially flat system, there is a pure integrator system (2.6) whose trajectories are also trajectories of the corresponding flat system. This equivalence is commonly established by a coordinate transformation and with one of three different approaches: By endogenous dynamic feedback (Fliess et al. (1995)), by smooth mapping of Cartan vector fields, called Lie-Bäcklund equivalence (Fliess et al. (1999a)), or by feedforward (Hagenmeyer and Delaleau (2003b)). Although these approaches use different mathematical toolkits, they can all be used to establish that, similarly to the pure integrator in

(2.6), a differentially flat system can be transformed to a system in Brunovský form

$$\begin{aligned}\dot{\zeta}_{i,0} &= \zeta_{i,1}\\ \dot{\zeta}_{i,1} &= \zeta_{i,2}\\ &\vdots\\ \dot{\zeta}_{i,\kappa_i} &= \nu_i \quad i = 1, \dots, m,\end{aligned} \tag{2.13}$$

with so-called *flat state*

$$\begin{aligned}\zeta = (&z_1, \dot{z}_1, \ddot{z}_1, \dots, z_1^{(\kappa_1-1)},\\ &z_2, \dot{z}_2, \ddot{z}_2, \dots, z_2^{(\kappa_2-1)},\\ &\vdots\\ &z_m, \dot{z}_m, \ddot{z}_m, \dots, z_m^{(\kappa_m-1)}),\end{aligned} \tag{2.14}$$

with $\zeta_{i,j} = z_i^{(j)}$ and $\sum_{i=1}^{m} \kappa_i = n$, and with a new input ν_i (Hagenmeyer and Delaleau 2003b). The resulting m subsystems are differentially independent. We will mainly study differentially flat systems in their Brunovský form in the remainder of this work. To this end, we adopt the feedforward approach to obtain the linear Brunovský form (2.13).

The feedforward approach (Hagenmeyer and Delaleau (2003a,b)) relies on the map (2.8c) with respect to a nominal trajectory $z^*(t)$ with consistent initial condition to form the nominal feedforward

$$u^* = \Psi(z^*, \dot{z}^*, \dots, z^{*(\gamma)}). \tag{2.15}$$

The application of the nominal feedforward combined with a change of coordinates (2.8b) from x to ζ yields (2.13) with $z = z^*$ in (2.14) and $\nu = (z^*)^{(\kappa_i)}$. The specific dynamics of the original systems are thus hidden in the feedforward. For the proof, we refer to Hagenmeyer and Delaleau (2003a,b).

This relation allows us to study a differentially flat system $\dot{x} = f(x, u)$ in terms of its linear Brunovský form, if we have the maps (2.8), and if we only apply nominal trajectories $z^* \in C^{\gamma-1}(\mathcal{T}, \mathcal{Z})$ with consistent initial conditions (2.11). In practice, we additionally need a feedback controller to achieve robust trajectory following, as shown by Hagenmeyer and Delaleau (2003b,c, 2010).

Example 1 (continued) *The non-linear dynamics* (2.9) *of the manipulator can be rearranged to*

$$\ddot{z} = \frac{g}{L}\sin(z) - \frac{1}{L^2 m}\tau.$$

In this form, the dynamics can be linearized for trajectories z^*, with consistent initial conditions $z^*(0)$, by the feedforward

$$\tau = \frac{L^2 m}{1}\left(\frac{g}{L}\sin(z^*) + \nu\right) \tag{2.16}$$

to obtain $\ddot{z} = \nu$, which can also be written in Brunovský form in terms of the flat state ζ as

$$\begin{aligned}\dot{\zeta}_1 &= \zeta_2\\ \dot{\zeta}_2 &= \nu.\end{aligned}$$

2.2.3 The Behavioral Systems Theory Perspective

The behavioral approach to dynamical system modeling developed by Jan C. Willems offers a different perspective on dynamical systems in general, which can provide some useful insight. A relatively recent overview of the theory by Willems himself can be found in Willems (2007). In behavioral systems theory, the dynamics of a system are represented as a collection of feasible trajectories of that system. Contrary to classic systems and control theory, the behavioral approach does not explicitly differentiate between inputs, states and outputs of a system, but assumes all quantities to be system variables. This perspective of systems as sets of trajectories is not limited to either continuous or discrete variables, but it unifies the two domains, which is a helpful framework for our investigations. The definition of differential flatness in the behavioral framework is remarkably concise and intuitive.

In behavioral system theory, a dynamical system Σ is defined as $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$ Willems (2007). The set $\mathbb{T}$ defines a time axis, e.g. $\mathbb{T} = \mathbb{R}_{\geq 0}$ for continuous-time differential equations. The system variables are from the set $\mathbb{W}$, also called the signal space. In the classic control systems language, these are the states, the inputs, and the outputs. In the present work, we assume that these variables can either be continuous or discrete, i.e., from a finite set like $\{0, 1\}$. The evolution over time of the system is specified by the behavior $\mathcal{B}$, which is defined by all possible trajectories as a map $\mathbb{T} \to \mathbb{W}$. This map does not necessarily exist as an explicit function, but it is thought of as a restriction of all possible functions to those that the given system can follow.
Our continuous-time dynamical system (2.1) can be expressed in the behavioral framework as follows: We define

$$w = \begin{bmatrix} x \\ u \end{bmatrix}, \tag{2.17}$$

such that the ODE (2.1) can be expressed as an implicit ODE

$$f_{\mathcal{B}}(w) = 0. \tag{2.18}$$

The behavior $\mathcal{B}$ of the system is characterized as the set of all trajectories $w(t)$ that the system can produce, i.e., $w \in \mathcal{B}$. In the behavioral framework, a differentially flat system is defined by the existence of an image representation in a so-called latent variable, which is often denoted by l (Trentelman 2004, Willems 2007). Because for flat systems, the latent variable is the flat output of the system, we denote it by z. The image representation has the form

$$w = g(z, z^{(1)}, \dots, z^{(r)}). \tag{2.19}$$

When a system is flat, there also exists the inverse mapping of the image representation

$$z = h(w, w^{(1)}, \dots, w^{(q)}). \tag{2.20}$$

The mapping h is not a real inversion of g, because the dimensions of the domain and the co-domain differ. From a trajectory perspective, however, we can call it an inverse mapping, because it can map a sufficiently smooth trajectory w^* to z^*, and vice versa. Both mappings only hold for $w \in \mathcal{B}$ and z sufficiently smooth. The interesting point here is the asymmetry of freedom between w and z: The trajectories w are constrained by the system behavior $\mathcal{B}$, whereas the trajectories z are entirely free, except for a smoothness constraint. Moreover, all these "free" trajectories z can be mapped to a "constrained" trajectory w by Equation (2.19). In the absence of explicit constraints on the system states and inputs, we can thus say that differentially flat systems are controllable according to Definition 2 in the latent variable: For all pairs z^0, z^f, we can construct a smooth trajectory connecting them, and mapping it back to w.

The behavioral definition of a dynamical system is very general, and it is very useful to understand systems from different mathematical domains in a common framework. In the context of hybrid systems, this language allows us to formulate and explain concepts in discrete-event systems and in continuous-time systems in the same language. A frequent criticism of the behavioral approach is its limited use in practical applications. On the other hand, we also want to emphasize its conceptual usefulness. In particular, we will make use of its generality to define flatness for automata in Chapter 4.1.

2.2.4 Computing Flatness

The biggest obstacle to applying flatness theory is identifying whether a system is flat and finding its flat outputs. In the literature, there is no general test or algorithm to compute flatness for non-linear systems. However, there are many useful results that allow us to work with flat systems. From a practical point of view, there are several known flat systems in various textbooks and other publications. Notably, the textbooks Sira Ramírez and Agrawal (2004), Levine (2009) include several examples, such as the non-holonomic car, robotic manipulators, electric drives, and a DC-to-DC power converter. Other famous examples include drones (Faessler et al. 2018), certain chemical reactors (Faulwasser et al. 2014), electrical micro grids (Kastner et al. 2023), and private communication (Nicolau et al. 2020).

Besides these examples, flatness is also given for all controllable linear systems, and for all non-linear systems that are linearizable by static feedback transformation (Sira Ramírez and Agrawal (2004). These two classes are subclasses of flat systems, and both properties can be algorithmically verified. For the remaining flat systems –those which are linearizable by *dynamic* feedback– there is no general test. However, there are mathematical results, algorithms, and conditions for specific systems, which can be found in Levine (2009), Lévine (2011), and the references therein.

Whether a system is flat or not also depends on the number of inputs and their "position", i.e., their dynamical interaction with the states. In many applications, these two parameters are fixed by the respective physics. However, if we have the freedom to change these, there is an opportunity to make the system flat by construction from an engineering point of view. This approach is described, for example, in Nicolau et al. (2020). Another well-known application of this principle is the construction of robotic manipulators (Spong et al. 2020). Commonly, the actuators are positioned such that the system is fully actuated, i.e., all joints are controlled. In this case, full actuation leads to differentially flat dynamics (Sira Ramírez and Agrawal 2004).

Although there are no generic tests or algorithms to compute whether a system is flat, it is sufficient to find a flat output. Even if the known system classes and algorithms fail to provide an answer, it is still possible to find a flat output by using "physical intuition, careful inspection, educated guessing" (Sira Ramírez and Agrawal (2004)).

3 Hybrid Automata

There are a number of different classes of hybrid system models in the literature. Their common feature is that their trajectories cannot be described by differentiable functions. The non-differentiable points in the trajectories mark the moments in time when continuous state variables change as jumps or when a discrete variable changes value. This switching can encode a discontinuous change of the continuous state variable, a discrete state variable, a change in the continuous dynamics, or a combination of these. Although all hybrid system models include some form of discrete events, there are hybrid system models that do not include discrete state variables. A very similar field of research is the study of non-smooth dynamical systems. These systems are mostly modeled without discrete states. In general, any non-smooth dynamical system can also be modeled as a hybrid system, but the perspective and the modeling paradigms are rather different. A hybrid system class that unites the two paradigms is the hybrid differential inclusion

$$\dot{x} \in F(x), \quad x \in \mathcal{C} \tag{3.1}$$

$$x^{+} \in G(x), \quad x \in \mathcal{D} \tag{3.2}$$

described by Goebel et al. (2009). The first relation (3.1) describes the continuous evolution of the state as a differential inclusion. The second part, Equation (3.2), describes jumps in the state, where x^{+} is the value to which the state jumps. A jump happens whenever the state reaches the set $\mathcal{D}$. These differential inclusions model hybrid systems without explicitly considering discrete states, and their very broad definition encompasses all types of hybrid and non-smooth systems. Such system descriptions sacrifice structure to establish generality. They thus allow for hiding complex relations in an abstract, compact form. The drawback of this generality and compactness is the difficulty of deriving such models for a technical system, and of deriving control algorithms that can be implemented in practice.

In the present work, we choose a very structured hybrid system model type, the Hybrid Automaton (HA). A HA consists of both continuous states x and discrete states, i.e., locations l. The dynamics of the locations l is described by an automaton, which is defined in the next section. For each location, a continuous time system as described above defines the dynamics of the continuous state x. Their interaction is defined by the switching rules, which we describe below. Thanks to their general structure, HA can incorporate properties of many other hybrid system classes. They can be thus seen as a superclass of most hybrid system models, but a subclass of differential inclusions, see e.g. Goebel et al. (2009).

3.1 Elements of Hybrid Automata

HA consist of a combination of two system classes: They combine finite automata and differential equations. In the literature, there are different modeling frameworks for hybrid automata, which differ in the exact definitions of their constituent parts. In the following sections, we first define the automaton model we use. After that, we take a closer look at two important properties: The type of switching or transition, and the system inputs.

3.1.1 Automaton

An automaton consists of locations $\mathsf{l} \in \mathsf{L}$ and discrete inputs $\mathsf{v} \in \mathsf{V}$. Both sets L and V are finite. We assume, w.l.o.g., that the discrete inputs are modeled as binary vectors, i.e., $\mathsf{v} \in \{0, 1\}^{n_v}$. The dynamics of the automaton can be modeled by a transition function

$$\mathsf{l}' = \delta(\mathsf{l}, \mathsf{v}), \tag{3.3}$$

which specifies a successor location l' for a given location and input. The mapping $\delta : \mathsf{L} \times \mathsf{V} \to \mathsf{L}$ is a deterministic partial mapping. Thus, for each pair (l, v) from the corresponding sets, there is either one successor l', or none. To characterize the set of inputs that cause a specific transition, we introduce the set-valued map $g : \mathsf{L}\times\mathsf{L} \to P(\mathsf{V})$, where $g(\mathsf{l}, \mathsf{l}') = \{\mathsf{v} \in \mathsf{V} : \mathsf{l}' = \delta(\mathsf{l}, \mathsf{v})\}$. $P(\mathsf{V})$ denotes the power set of V, i.e., the set of all subsets of V, including the empty set, which is attained if no transition between l and l' exists. Every transition is therefore associated with a set of inputs that cause the transition. As a special case, a transition can be caused by a single input, such that the mapping becomes $\{\mathsf{v}\} = g(\mathsf{l}, \mathsf{l}')$.

Transitions can be either controlled or autonomous.[1] A controlled transition is caused by a subset of all possible inputs, i.e., $g(\mathsf{l}, \mathsf{l}') \subset \mathsf{V}$. We assume that for each location l, there is a subset of the power set $P(\mathsf{V})$, which does not cause a transition, i.e., for all $\mathsf{l} \in \mathsf{L}$, there is a $\mathsf{v} \in \mathsf{V}$, such that $\delta(\mathsf{l}, \mathsf{v})$ yields no successor. An autonomous transition can not be prevented (or caused) by the input, i.e., $g(\mathsf{l}, \mathsf{l}') = P(\mathsf{V})$.

A run of an automaton consists of a sequence of locations $(\mathsf{l}^j)_{j\in 0,\ldots,n}$ and corresponding inputs $(\mathsf{v}^j)_{j\in 0,\ldots,n}$ for some $n \in \mathbb{N}$. The structure of an automaton can be represented by a graph $\{\mathsf{L}, \mathsf{E}\}$, with vertices $\mathsf{l} \in \mathsf{L}$ and edges $\mathsf{e} = (\mathsf{l}, \mathsf{l}') \in \mathsf{E}$, where the edges are the transitions defined by $\delta(\cdot)$. An edge $\mathsf{e} = (\mathsf{l}, \mathsf{l}')$ has a head l' and a tail l. The graph representation does not include the inputs, but it encodes all possible sequences of locations, because all edges are deduced from the automaton dynamics (3.3).

3.1.2 Transitions

In the present thesis, all changes in location l are caused by one of two types of conditions: Either a particular state x is reached, which triggers the transition, or a discrete input signal v causes the transition. We use the terms *transition* and *switching* interchangeably to describe the change in location l, and the associated

[1] In the literature on discrete event systems, the definition of controlled and uncontrolled is different than ours: An event is controllable when it can be prevented from occurring, and it is uncontrollable if it can not be prevented (see e.g. Ramadge and Wonham (1989)).

change in the state x.[2] When transitions are driven by the continuous state, it is called autonomous switching or uncontrolled switching in the literature. In the following, we use the term *autonomous switching*. The conditions for such transitions are defined by sets, the so-called *guards* or *guard sets* $\mathcal{G}_{\mathsf{e}}^{x} \subset \mathcal{X}$. Thus, the transition is triggered when the state x enters a guard set, i.e., $x \in \mathcal{G}_{\mathsf{e}}^{x}$. This can model, for example, reaching a certain spatial position or an equilibrium state. Additionally, there is a reset function $x^{+} = L_{\mathsf{e}}(x^{-})$ that defines the change of the continuous state, and some discrete transition that defines the new location. The change of the location is modeled as an edge in a directed graph $\mathsf{e} = (\mathsf{l}, \mathsf{l}')$, which is an ordered pair of locations.

When a transition is caused by a discrete input signal v, it is called *forced switching* or *controlled switching*. There are two types of forced switching: The first type causes a specific switching action independently of the current system and location. This type of switching occurs for example in switched systems, in which a discrete signal selects the continuous dynamics. The second type of forced switching is restricted by the current location. From a given location, only a subset of forced transitions is allowed. In this setting, the discrete dynamics are more structured. Guard sets of forced switching are denoted by $\mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$ and are subsets of the power set of all possible discrete input combinations, i.e., $\mathcal{G}_{\mathsf{e}}^{\mathsf{v}} \subset P(\mathsf{V})$.

All transitions are thus described by a tuple $\{\mathsf{e}, \mathcal{G}_{\mathsf{e}}^{x}, L_{\mathsf{e}}\}$ or by a tuple $\{\mathsf{e}, \mathcal{G}_{\mathsf{e}}^{\mathsf{v}}, L_{\mathsf{e}}\}$. We assume that all transitions happen immediately, as soon as the condition is satisfied. Therefore, a transition $x(t')^{+} = L_{\mathsf{e}}(x(t')^{-})$ happens at a single time instant $t = t'$. As a consequence, the system state does not have a uniquely defined value at that point in time. We will resolve this ambiguity with the introduction of hybrid time on solutions and trajectories in the Section 3.3 below.

[2] Transitions, which are caused by passing time, can be modeled by the inclusion of a time variable as a continuous state. We do not include such timed transitions in our considerations because the inputs cannot control time.

3.1.3 Inputs

In the dynamical systems literature, one distinguishes systems with or without inputs. The same distinction can be made for hybrid systems, but for these, we have two possible types of inputs: discrete inputs $\mathsf{v} \in \{0, 1\}^{n_\mathsf{v}}$ and continuous inputs $u \in \mathbb{R}^m$. The continuous inputs directly influence the continuous trajectories, whereas the discrete inputs can cause transitions. In general, an input is discrete if the number of values it can have is countable, i.e., $\mathsf{v} \in \mathbb{N}$. However, any countable finite set can be mapped to a combination of binary variables. For the sake of simplicity, we therefore only consider binary discrete inputs.

Hybrid systems with no inputs are called autonomous. Therefore, they can only exhibit autonomous switching triggered by the continuous state. Hybrid systems with only continuous inputs do also include autonomous switching, but their trajectories can be controlled.

When only discrete inputs are present, the continuous dynamics of the system are autonomous, but switching can be autonomous and controlled. One important system class with only discrete inputs is switched systems of the form

$$\dot{x} = f_{\mathsf{v}(t)}(x). \tag{3.4}$$

The discrete input is a piecewise constant signal that selects the continuous dynamics. In switched systems, only controlled switching is allowed. In HA with continuous and discrete inputs, both types of switching can be modeled.

3.2 Hybrid Automaton Model

There are different definitions of hybrid automata in the literature. The following definition is based on Branicky et al. (1998) and our previous publications Zahn et al. (2022, 2024).

Definition 4 (Hybrid Automaton) *A hybrid automaton Σ_H is a dynamical system defined by the tuple $(\mathsf{L}, \mathsf{E}, \mathsf{V}, \Sigma, \mathcal{G}_\mathsf{e}, L_\mathsf{e}, \mathit{Init})$, consisting of the following:*

- $\mathsf{L} = \{\mathsf{l}_i\}$, $i \in 0, \ldots, n_\mathsf{l}$, *is the set of locations, which are the vertices of the automaton graph of the hybrid automaton.*
- $\mathsf{E} = \{\mathsf{e}_k\}$, $k \in 0, \ldots, n_\mathsf{e}$, *is the set of directed transitions* $\mathsf{e}_k = (\mathsf{l}_i, \mathsf{l}_j)$, $i \neq j$. *The sets* L, E *form a digraph.*
- $\mathsf{V} = \{0, 1\}^{n_\mathsf{v}}$ *are the* n_v *discrete binary inputs.*
- $\Sigma = \{\Sigma_\mathsf{l}\}_{\mathsf{l} \in \mathsf{L}}$ *is a collection of continuous dynamical systems. Each system* Σ_l *consists of continuous dynamics* $\dot{x}_\mathsf{l} = f_\mathsf{l}(x_\mathsf{l}, u_\mathsf{l})$, *a state space* $\mathcal{X}_\mathsf{l} \subset \mathbb{R}^{n_\mathsf{l}}$ *and inputs* $\mathcal{U}_\mathsf{l} \in \mathbb{R}^{m_\mathsf{l}}$. *The vector fields* f *are locally Lipschitz continuous.*
- $\mathcal{G}_\mathsf{e}^\mathsf{v}$ *or* $\mathcal{G}_\mathsf{e}^x$ *are guard sets for each edge* $\mathsf{e}\,(\mathsf{l}_i, \mathsf{l}_j) \in \mathsf{E}$. $\mathcal{G}_\mathsf{e}^x \subset \mathcal{X}_{\mathsf{l}_i}$ *are nonempty closed sets, which guard transitions caused by the value of the continuous state, often called autonomous or uncontrolled transitions. The guards* $\mathcal{G}_\mathsf{e}^\mathsf{v} \subset \mathsf{V}$ *define transitions caused by the discrete input, called forced or controlled transitions. The guard sets are disjoint, such that all possible* v *or* x *either belong to no guard set or to exactly one, i.e.,* $\mathcal{G}_{\mathsf{e}_i}^x \cap \mathcal{G}_{\mathsf{e}_j}^x = \emptyset$ *for* $i \neq j$, *and* $\mathcal{G}_{\mathsf{e}_i}^\mathsf{v} \cap \mathcal{G}_{\mathsf{e}_j}^\mathsf{v} = \emptyset$ *for* $i \neq j$, *such that the system is deterministic.*
- $L_\mathsf{e} : \mathcal{X}_{\mathsf{l}_i} \mapsto \mathcal{X}_{\mathsf{l}_j}$ *define reset maps for each edge* $\mathsf{e} = (\mathsf{l}_i, \mathsf{l}_j)$. *These maps specify the change of the continuous state when a transition takes place.*
- $\mathit{Init} \subset \Omega$ *is a set of possible initial conditions* (l^0, x^0) *in the hybrid state space* $\Omega = \cup_{\mathsf{l} \in \mathsf{L}} \{\mathsf{l}\} \times \mathcal{X}_\mathsf{l}$. □

The state of the HA is the hybrid state (l, x) from the aforementioned hybrid state space Ω. Whenever either the hybrid state or the discrete input matches a guard set, the transition takes place immediately.

We also define a set $\mathsf{Arrive}_{\mathsf{l}}^{0}$, which is computed by mapping the continuous states before switching x^- to the corresponding successor states x^+ in each location. This set is defined as

$$\begin{aligned}\mathsf{Arrive}_{\mathsf{l}}^{0} =&\{x \in \mathcal{X}_{\mathsf{l}} : x = L_{\mathsf{e}}(x') \text{ for all } e = (\mathsf{l}', \mathsf{l}) \\ &\text{and } x' \text{ is such that} \\ &x' \in \mathcal{G}_{\mathsf{e}}^{x} \text{ if } \mathcal{G}_{\mathsf{e}}^{x} \neq \emptyset \\ &\text{or } x' \in \mathcal{X}_{\mathsf{l}'} \text{ if } \mathcal{G}_{\mathsf{e}}^{\mathsf{v}} \neq \emptyset\}.\end{aligned}$$

To characterize all possible initial states in a given location l, we combine the corresponding arrival set $\mathsf{Arrive}_{\mathsf{l}}^{0}$ with the initial states

$$\mathsf{Arrive}_{\mathsf{l}} = \mathsf{Arrive}_{\mathsf{l}}^{0} \bigcup \mathsf{Init}_{\mathsf{l}}. \tag{3.5}$$

This set aggregates all possible starting points of continuous trajectories in a location l.

Furthermore, we define that the arrival sets are mutually exclusive with the guard sets in a given location, i.e., $\mathsf{Arrive}_{\mathsf{l}} \bigcap \mathcal{G}_{\mathsf{e}}^{x} = \emptyset$.[3]

The elements of the hybrid automaton model are summarized in Figure 3.1 exemplarily for a hybrid automaton with two locations.

The state spaces $\mathcal{X}_{\mathsf{l}}$ and the input spaces $\mathcal{U}_{\mathsf{l}}$ of the dynamical systems are defined by the allowed states and inputs, thereby including the constraints of the system. As an auxiliary set we define a flow set $\mathsf{Flow}_{\mathsf{l}}$, which specifies the states in which the system can evolve without causing a transition. This flow set is defined by

$$\mathsf{Flow}_{\mathsf{l}} = \mathcal{X}_{\mathsf{l}} / \bigcup_{\mathsf{l}' \in \mathsf{L}} \mathcal{G}_{(\mathsf{l},\mathsf{l}')}^{x}. \tag{3.6}$$

[3] We do not include a similar condition for the guards $\mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$. Thus, technically, the inputs can cause multiple transitions in a row. This choice is intentional, as we assume that the inputs can be chosen freely, and thus forced transitions are always intentional. To avoid unwanted transitions, in Section 3.1.1 we defined that there is always a combination of discrete inputs that does not cause a transition, thereby allowing to stay in a location.

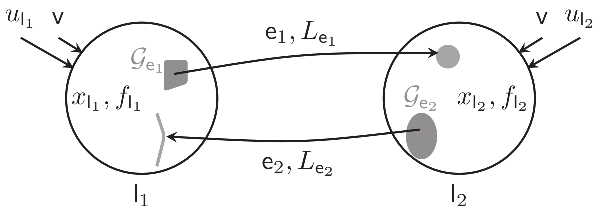

u_{l_1}
v
$\mathcal{G}_{\mathsf{e}_1}$
$x_{\mathsf{l}_1}, f_{\mathsf{l}_1}$
$\mathsf{e}_1, L_{\mathsf{e}_1}$
$\mathsf{e}_2, L_{\mathsf{e}_2}$
v
u_{l_2}
$\mathcal{G}_{\mathsf{e}_2}$
$x_{\mathsf{l}_2}, f_{\mathsf{l}_2}$
l_1
l_2

Abbildung 3.1: Depiction of basic elements of our hybrid automaton model.

We defined above that transitions can either be caused by the continuous state x, or by the discrete input v. In some cases, however, one might want to define guard sets $\mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}} \subset \mathcal{X} \times \mathsf{V}$, involving both the continuous state and the discrete input. This can be the case when a transition depends on the physical state of the system and a discrete input at the same time, e.g., reaching a specific desired state and receiving the external signal to move on. Although guard sets $\mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$ are not included in our definition, one can always find an equivalent hybrid system model, in which such a guard set is separated into two sets $\bar{\mathcal{G}}_{\mathsf{e}}^{x}$, $\bar{\mathcal{G}}_{\mathsf{e}}^{\mathsf{v}}$, and an additional location. The set $\bar{\mathcal{G}}_{\mathsf{e}}^{x}$ is the continuous part of $\mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$, such that for all $x \in \bar{\mathcal{G}}_{\mathsf{e}}^{x}$, there is a v, such that $(x, \mathsf{v}) \in \mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$. The set $\bar{\mathcal{G}}_{\mathsf{e}}^{\mathsf{v}}$ is the discrete part of $\mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$, such that for all $\mathsf{v} \in \bar{\mathcal{G}}_{\mathsf{e}}^{\mathsf{v}}$, there is a x, such that $(x, \mathsf{v}) \in \mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$. For a given guard set $\mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$, the separation can be achieved in one of the following ways, which are illustrated in Figure 3.2:

1. (blue part in upper half of Figure 3.2) Replace the corresponding edge $\mathsf{e} = (\mathsf{l}_1, \mathsf{l}_2)$ of $\mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$ by three new edges $\bar{\mathsf{e}}_1 = (\mathsf{l}_1, \bar{\mathsf{l}})$, $\bar{\mathsf{e}}_2 = (\bar{\mathsf{l}}, \mathsf{l}_1)$, $\bar{\mathsf{e}}_3 = (\bar{\mathsf{l}}, \mathsf{l}_2)$ and a location $\bar{\mathsf{l}}$. In the new location $\bar{\mathsf{l}}$, the flow set is only the nonempty guard set $\bar{\mathcal{G}}_{\mathsf{e}}^{x}$. The guard set $\mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$ in l_1 is replaced with a set $\bar{\mathcal{G}}_{\mathsf{e}}^{x}$, and the reverse switching is introduced as $\bar{\mathsf{e}}_2 = (\bar{\mathsf{l}}, \mathsf{l}_1)$. The reverse switching is triggered when the continuous state leaves the set $\bar{\mathcal{G}}_{\mathsf{e}}^{x}$. The guard set for $\bar{\mathsf{e}}_3 = (\bar{\mathsf{l}}, \mathsf{l}_2)$ in $\bar{\mathsf{l}}$ is $\bar{\mathcal{G}}_{\mathsf{e}}^{\mathsf{v}}$.

2. (red part in lower half of Figure 3.2) Replace the corresponding edge $\mathsf{e} = (\mathsf{l}_1, \mathsf{l}_2)$ of $\mathcal{G}_{\mathsf{e}}^{x,\mathsf{v}}$ by three new edges $\bar{\mathsf{e}}_1 = (\mathsf{l}_1, \bar{\mathsf{l}})$, $\bar{\mathsf{e}}_2 = (\bar{\mathsf{l}}, \mathsf{l}_1)$, $\bar{\mathsf{e}}_3 = (\bar{\mathsf{l}}, \mathsf{l}_2)$ and

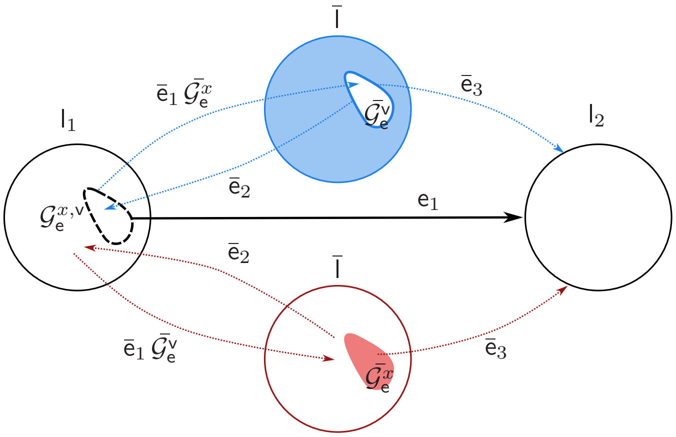

$\bar{\mathsf{l}}$
$\bar{\mathsf{e}}_1\ \bar{\mathcal{G}}^{x}_{\mathsf{e}}$
$\bar{\mathsf{e}}_3$
l_1
l_2
$\bar{\mathcal{G}}^{\mathsf{v}}_{\mathsf{e}}$
$\mathcal{G}^{x,\mathsf{v}}_{\mathsf{e}}$
$\bar{\mathsf{e}}_2$
e_1
$\bar{\mathsf{e}}_2$
$\bar{\mathsf{l}}$
$\bar{\mathsf{e}}_1\ \bar{\mathcal{G}}^{\mathsf{v}}_{\mathsf{e}}$
$\bar{\mathsf{e}}_3$
$\bar{\mathcal{G}}^{x}_{\mathsf{e}}$

Abbildung 3.2: Two options for the replacement of a mixed guard set.

a location $\bar{\mathsf{l}}$. In the new location $\bar{\mathsf{l}}$, the flow set is an exact copy of the original flow set of l_1. The guard set $\mathcal{G}^{x,\mathsf{v}}_{\mathsf{e}}$ in l_1 is replaced with a set $\bar{\mathcal{G}}^{\mathsf{v}}_{\mathsf{e}}$ for $\bar{\mathsf{e}}_1$, and the reverse switching is introduced as $\bar{\mathsf{e}}_2 = (\bar{\mathsf{l}}, \mathsf{l}_1)$. The reverse switching is triggered whenever $\mathsf{v} \notin \bar{\mathcal{G}}^{\mathsf{v}}_{\mathsf{e}}$. The guard set for $\bar{\mathsf{e}}_3 = (\bar{\mathsf{l}}, \mathsf{l}_2)$ in $\bar{\mathsf{l}}$ is $\bar{\mathcal{G}}^{x}_{\mathsf{e}}$.

This replacement partitions the transition action into two different steps: One switching step, which is caused solely by the discrete input v, and another switching step caused by the state x.

3.3 Solutions and Trajectories

A hybrid trajectory of the HA can be defined on a hybrid time domain. We use the definition of hybrid time domains from Goebel et al. (2009), Bernard and Sanfelice (2020):

Definition 5 (Finite hybrid time domain) *A union* $\mathcal{T} = \bigcup_{j\in(0,\dots,j^f)}[t^j, t^{j+1}]\times \{j\})$, *where* $0 = t^0 \leq t^1 \leq \dots \leq t^{j^f} = t^f$ *is some finite sequence of switching* *times with all* $t^j \in \mathbb{R}_{\geq 0}, j \in \mathbb{N}_{\geq 0}$*, is called a finite hybrid time domain* $\mathcal{T} \subset [0, t^f] \times (0, \ldots, j^f)$. $\square$

Transitions in the hybrid automaton happen at times t^j according to this definition, except for the times t^0 and t^f, which are initial and final times. As we only consider finite time domains in the present work, we use the terms *hybrid time* or *hybrid time domain* without the adjective *finite*. The hybrid time domain and the automaton model allow for two transitions in a single time instant t^j, in which case $t^j = t^{j+1}$. This can happen for successive transitions, in which the first transition is caused by a guard set $\mathcal{G}^x_\mathsf{e}$, and the subsequent transition is immediately triggered by the discrete input v. More than two simultaneous transitions are not possible, however, because we defined guard sets to be disjoint in Definition 4.

The hybrid state is uniquely defined at all hybrid times (t^j, j). The transitions and corresponding state resets can be described in hybrid time

$$\mathsf{e} = (\mathsf{l}(t^j, j-1), \mathsf{l}(t^j, j)) \tag{3.7}$$

$$x(t^j, j) = L_\mathsf{e}(x(t^j, j-1)). \tag{3.8}$$

With these relations, we can also properly define the shorthand notations x^+, x^-, l^+, l^- for a given transition e at time t^j:

$$\begin{aligned} x^+ &= x(t^j, j) \\ x^- &= x(t^j, j-1) \\ \mathsf{l}^+ &= \mathsf{l}(t^j, j) \\ \mathsf{l}^- &= \mathsf{l}(t^j, j-1). \end{aligned}$$

The trajectories of a hybrid automaton's variables are defined on a hybrid time domain. Such a function is called a hybrid trajectory; in the literature, it is also called a hybrid arc. A solution or a run of a hybrid automaton consists of the hybrid trajectories of $x, \mathsf{l}, u, \mathsf{v}$ with a corresponding hybrid time domain $\mathcal{T}$.

Definition 6 (Solution of a hybrid automaton) *The hybrid trajectories x^*, l^*, u^*, v^* over a hybrid time domain $\mathcal{T}^*$ are a solution of the hybrid automaton Σ_H, iff*

- $(\mathsf{l}^*(0,0), x^*(0,0)) \in \mathit{Init}$ *(initial condition)*
- *for all $(t^j, j) \in \mathcal{T}^*$ with $0 < j < f$:*
  - $(\mathsf{l}^*(t^j, j-1), \mathsf{l}^*(t^j, j)) = \mathsf{e} \in \mathsf{E}$ *(transitions in automaton graph)*
  - $x^*(t^j, j-1) \in \mathcal{G}^x_{\mathsf{e}} \quad \vee \quad \mathsf{v}^*(t^j, j-1) \in \mathcal{G}^{\mathsf{v}}_{\mathsf{e}}$ *(fullfillment of guards)*
  - $x^*(t^j, j) = L_{\mathsf{e}}(x^*(t^j, j-1))$ *(resets of continuous states)*
  - $x^*(t,j) \in \mathcal{X}_{\mathsf{l}^*(t,j)}, \quad t \in [t^j, t^{j+1}]$ *and $x^*_{\mathsf{l}^*(t,j)}, u^*_{\mathsf{l}^*(t,j)}$ are solutions to $f_{\mathsf{l}^*(t,j)}$ (solutions to continuous dynamics).* □

A solution to a hybrid automaton thus satisfies all specifications or data of the system. The four functions $x^*, \mathsf{l}^*, u^*, \mathsf{v}^*$ completely determine the solution. The corresponding hybrid time domain can be deduced from the function l^*, which encodes all transition times t^j.

We can define a solution map for solutions of a HA in the same way as for an ODE; mapping from an initial state (l^0, x^0), input functions (v^*, u^*) and hybrid time (t,j) to a hybrid state at time (t,j):

$$(\mathsf{l}^*(t,j), x^*(t,j)) = \phi_{\Sigma_H}((t,j), (\mathsf{l}^0, x^0), (\mathsf{v}^*, u^*)), \quad \forall (t,j) \in \mathcal{T}^*. \tag{3.9}$$

3.4 Reachability and Controllability

With the trajectory operator, we can also define reachability and controllability similarly to how we did for ODEs above, by replacing the state and time with their hybrid counterparts. For controllability and for reachability of hybrid states, this is straightforward, and we do not include the definitions to avoid redundancy. For reachability of sets, however, we either have to differentiate between sets of locations and sets of continuous states, or we have to rewrite the definition in terms of sets of hybrid states. We choose the former option, because it allows for separate evaluation and for more precision. We first define reachability of locations in terms of initial and final locations, without choosing a particular continuous state. A location l^f is reachable from location l^0, if there are corresponding continuous states x and inputs, such that a hybrid trajectory between them exists.

Definition 7 (Reachable Location) *Given a HA Σ_H, then a location $\mathsf{l}^f \in \mathsf{L}$ is reachable from $\mathsf{l}^0 \in \mathsf{L}$, if there is a state $x^0 \in \mathcal{X}_{\mathsf{l}^0}$, inputs v^*, u^* and a hybrid time domain $\mathcal{T}^*$, such that*

$$(\mathsf{l}(t^f, f), x(t^f, f)) = \phi_{\Sigma_H}((t^f, f), (\mathsf{l}^0, x^0), (\mathsf{v}^*, u^*))$$

with $\mathsf{l}(t^f, f) = \mathsf{l}^f$ and $x(t^f, f) \in \mathcal{X}_{\mathsf{l}^f}$. □

For reachability of sets of continuous states, we specify subsets of the state sets of locations. We thus have two state sets $x^0 \subset \mathcal{X}_{\mathsf{l}^0}$, and $x^f \subset \mathcal{X}_{\mathsf{l}^f}$ from corresponding initial and final locations.

Definition 8 (Reachable Set of continuous HA states) *Given a HA Σ_H, and locations $\mathsf{l}^f \in \mathsf{L}$, $\mathsf{l}^0 \in \mathsf{L}$, then a set $\mathcal{X}^f \subset \mathcal{X}_{\mathsf{l}^f}$ is reachable from $\mathcal{X}^0 \subset \mathcal{X}_{\mathsf{l}^0}$, if for all $x^0 \in \mathcal{X}^0$, there is a state $x^f \in \mathcal{X}^f$ and there are inputs v^*, u^* and a hybrid time domain $\mathcal{T}^*$, such that*

$$(\mathsf{l}^f, x^f) = (\mathsf{l}(t^f, f), x(t^f, f)) = \phi_{\Sigma_H}((t^f, f), (\mathsf{l}^0, x^0), (\mathsf{v}^*, u^*)).$$

□

These definitions allow us to evaluate reachability of the locations and the continuous states separately. Controllability can be defined by generalizing Definition 8 as reachability of every state from every state. If all continuous states are reachable, this automatically implies that all locations can be reached. Therefore, the generalization of reachability of locations in Definition 7 is a necessary condition for controllability, and it can be evaluated separately. Proving controllability and reachability for HA is generally very hard. Without introducing limiting assumptions, proving reachability of HA is even undecidable (Lunze and Lamnabhi-Lagarrigue 2009, Chapter 3).

The properties of controllability and reachability for other hybrid system classes are both studied in the hybrid system literature, e.g. Ezzine and Haddad (1989), Bemporad et al. (2000), Caines and Lemch (2002). Although these works include theoretical results for controllability, most of the literature focuses on reachability, in particular on numerical reachability computation (Althoff et al. 2010, 2021). Proving controllability of hybrid systems is generally difficult, because it is a global property, stated in terms of “from all, to all”. It is therefore inherently difficult to compute numerically. Theoretical results are also hard to obtain. For HA, a definition of controllability is given by van Schuppen (1998), which relies on the characterization of reachable sets, without specifying how these reachable sets can be computed. The works by Lemch et al. (2000) and Liñán et al. (2020) both state mathematical controllability conditions for HA, but they do not provide algorithms or approaches to check these conditions. Therefore, to the best knowledge of the author, proving controllability of HA remains an open problem.

The literature on reachable sets for hybrid systems is more developed, much of it going under the name of *hybrid system verification* (Alur 2011, Doyen et al. 2018). Its primary focus is the numerical computation of reachable sets, and there are a number of approaches and algorithms to compute them. For an overview on this topic, see Althoff et al. (2021). Although these techniques have proven to be useful, they rely on approximations and suffer from the curse of dimensionality, making them hard to scale. Their general approach is to fix an initial set of a particular shape and to compute stepwise propagation in discrete time through the HA.

The flatness-based approach to HA that we present in the following chapter offers a simplification: Instead of evaluating reachability over the whole HA, it allows us to partition the verification by location. Additionally, the differential flatness of the continuous dynamics enables us to formulate computable conditions and algorithms to evaluate reachability, which we present in the subsequent Chapter 5.

4 Flat Hybrid Automata

The core of the present thesis is the novel definition of flatness for HA. To extend flatness from continuous time systems to HA, we first derive a definition of flatness for automata in Section 4.1. This definition has been published in Zahn et al. (2024). Based on these results, we move on to define flatness for HA in Section 4.2. This definition is based on the previous publications Zahn et al. (2022, 2024), but we present a refined version with slightly different notation. In the last part of this chapter, we explain the connection and differences between flatness and controllability in HA.

4.1 Automaton Flatness

The following material is based on Zahn et al. (2024), where we introduce a definition of flatness for automata. The classic flatness definitions from Equations (2.8) cannot be applied directly to automata, because there is no equivalent concept of a time derivative, and their inputs and states are discrete valued, not continuous. The behavioral definition introduced above is more general. Applying the behavioral framework to modeling and control of discrete and hybrid systems has been reported, for example, in Willems (1991) and in Moor and Raisch (1999), Moor et al. (2001). We start by modeling the automata in behavioral systems language. To this end, we regard automata as being represented by their underlying directed automaton graph $\{\mathsf{L}, \mathsf{E}\}$. The graph is the common structure of most types of automata, e.g., discrete-event systems (Ramadge and Wonham 1989). We use the symbols defined above to model the automaton with locations $\mathsf{l} \in \mathsf{L}$ and edges or events $\mathsf{e} \in \mathsf{E}$, which form the simple directed graph of the automaton, as defined in Section 3.1.1. We assume that the transition from one discrete state to another, represented by an edge, is controllable. A transition is controllable if it is caused by a specific input, as defined in Chapter 3. In the following, we use the terms edge and event interchangeably.

Trajectories of such an automaton can be represented in two equivalent ways: Either as a sequence of locations $(\mathsf{l}^j)_{j\in 0,\dots,n}$, or as a sequence of events $(\mathsf{e}^k)_{k\in 0,\dots,m}$ for some $n, m \in \mathbb{N}$. In the following, we use the short form notation $(\mathsf{l}^j)_j$. All trajectories that are consistent with the dynamics of the automaton, or, equivalently, which are sequences consistent with the automaton graph, form the behavior of the automaton, in the sense of Willems theory as introduced in Section 2.2.3.[1] Given that we only consider automata with simple directed graphs, the two representations of trajectories in either l or in e are equivalent. For the definition of flatness for automata, we only consider sequences of the locations.

The image representation defining flatness has to map between trajectories of the system to trajectories of a trivial system. Thus, for automata we can find a mapping

$$(\mathsf{l}^j)_j = g((\mathsf{d}^k)_k) \tag{4.1}$$

$$(\mathsf{d}^k)_k = h((\mathsf{l}^j)_j) \tag{4.2}$$

in terms of sequences $(\mathsf{l}^j)_j$ that are feasible. The different indices j, k indicate that the sequences can be of different lengths. The mapping (4.1) is the image representation of the behavior, and (2.20) is its inverse mapping. For a system to be flat with respect to these equations, the behavior in the variable d has to be free, i.e., all sequences $(\mathsf{d}^k)_k$ have to be feasible. From this condition, we can easily deduce which type of representation can be a candidate for an image representation of an automaton $\{\mathsf{L}, \mathsf{E}\}$, namely an automaton with a complete graph $\{\mathsf{D}, \hat{\mathsf{E}}\}$ and with the same set of locations. In a complete graph, any pair of

[1] The behavior of an automaton is often characterized in terms of the sequences of events, the so-called *language* of an automaton (Ramadge and Wonham 1989). We choose the characterization by sequences of locations, because it is closer to the concept of states in differential flatness theory.

discrete states $\mathsf{d}^i, \mathsf{d}^j$ is connected by one edge $\hat{\mathsf{e}}$. Therefore, all sequences $(\mathsf{d}^k)_k$ are feasible by definition. In accordance with the behavioral definition of flatness, we can thus say that an automaton $\{\mathsf{L}, \mathsf{E}\}$ is flat if it has an image representation as the behavior of an automaton with complete graph $\{\mathsf{D}, \hat{\mathsf{E}}\}$ and with the same set of locations, see Figure 4.1. That means that $\mathsf{D} = \mathsf{L}$. This is precisely the case for automata with a strongly connected automaton graph. In this case, the original automaton graph is a spanning subgraph of a corresponding complete graph (Bang-Jensen and Gutin 2007).

Theorem 2 *An automaton $\{\mathsf{L}, \mathsf{E}\}$ with a strongly connected automaton graph and controllable events has an image representation $(\mathsf{l}^j)_j = g((\mathsf{d}^k)_k)$ and an inverse mapping $(\mathsf{d}^k)_k = h((\mathsf{l}^j)_j)$, where $(\mathsf{d}^k)_k$ are feasible sequences of an automaton with complete graph $\{\mathsf{D}, \hat{\mathsf{E}}\}$.*

Proof 1 *We define the sets of all feasible sequences $\mathsf{L}^*, \mathsf{D}^*$ for both the flat automaton and the complete automaton, both of which have the same number of discrete states. As mentioned above, for the complete automaton, any sequence $(\mathsf{d}^k)_k$ with $\mathsf{d}^k \in \mathsf{D}^*$ is feasible. First, we prove that there exists a mapping $g : \mathsf{L}^* \to \mathsf{D}^*$: Mapping each state $\mathsf{l} \in \mathsf{L}$ to a state $\mathsf{d} \in \mathsf{D}$ in a one-to-one manner by a bijection $\varphi(\cdot)$, we can see that the graph $\{\mathsf{L}, \mathsf{E}\}$ is a spanning subgraph of $\{\mathsf{D}, \hat{\mathsf{E}}\}$, see Bang-Jensen and Gutin (2007). Therefore, every sequence $(\mathsf{l}^j)_j \in \mathsf{L}^*$ is also a sequence in D^* after a state transformation, i.e., $\varphi((\mathsf{l}^j)_j) \in \mathsf{D}^*$.*
Next, we prove the existence of the inverse mapping $h : \mathsf{D}^* \to \mathsf{L}^*$: In this case, not every sequence $(\mathsf{d}^k)_k \in \mathsf{D}^*$ is also a sequence in L^*. However, due to the strong connectedness of $\{\mathsf{L}, \mathsf{E}\}$, we know that for any two states $\mathsf{l}_i, \mathsf{l}_j \in \mathsf{L}$, there is a sequence in L^* with first and last element l^i and l^j, respectively. Thus, we can define a map $\psi : \mathsf{D} \times \mathsf{D} \to \mathsf{L}^*$ which maps every pair $\mathsf{d}^i, \mathsf{d}^j$ to a sequence $\{\varphi^{-1}(\mathsf{d}^i), (\mathsf{l}^i)_i, \varphi^{-1}(\mathsf{d}^j)\}$.

Based on Theorem 2, we define flatness of automata.

Definition 9 (Flat automaton) *An automaton $\{\mathsf{L}, \mathsf{E}\}$ with strongly connected automaton graph and controllable events is a flat automaton.*

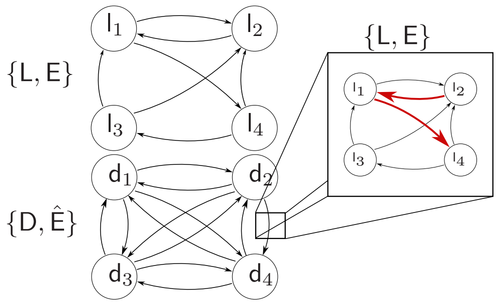


Abbildung 4.1: Example of equivalent paths in a strongly connected graph $\{L, E\}$ and a complete graph $\{D, \hat{E}\}$.

Remarks on the Similarities and Differences of Automaton Flatness to Differential Flatness

There are interesting parallels and differences between differential flatness and automaton flatness because of their difference in time and information structure. In differential flatness, time derivatives play an important role. The time derivatives of an analytical function contain information about the whole function at a single time point. In contrast, the sequences used in the definition of automaton flatness cannot be reduced to a single time instant. In the differential geometric framework (Fliess et al. 1999a), flatness and equivalence of dynamical systems are defined by resorting to infinite derivatives of the system input. Together with an initial state, the infinite derivatives effectively define the entire analytical input function and, as a consequence, the whole system trajectory. Thus, all trajectory information is concentrated in a single time point. In automata, a sequence is always defined over multiple time steps, but it contains the analogous information as the derivatives of a differentially flat continuous system: When the current state and a sequence of inputs for an automaton are known, the trajectory of the automaton is entirely determined.

As a consequence of this difference in the time structure, a one-step sequence in the complete graph may be mapped to a multi-step sequence in the strongly connected graph (see Figure 4.1). However, two key aspects of flatness are preserved: 1. All trajectories in the complete graph can be realized in the original strongly connected graph, and 2. the trajectories in the complete graph can be inverted to calculate the corresponding events.

We also point out that, in a real-world application, flatness of continuous systems displays a similar challenge: As the mapping from the flat output to the states and inputs involves the derivatives of z, we can never precisely calculate the inputs and the states from the flat output alone, because the actual time derivatives cannot be measured directly.

Furthermore, the mapping (4.2) is not unique, which is analogous to the flat output of continuous systems that are also not unique, see Section 2. In general, a sequence in the complete automaton that includes transitions which are not part of the original automaton can be mapped to different original sequences. This is inevitable, as the domain and the codomain of the mapping do not have the same cardinality (by definition, there are more edges in the complete graph, and therefore more feasible sequences). Therefore, a bijection does not exist. From a trajectory computation perspective, this is not a problem, as long as the mapping is known and fixed.

The use of sequences instead of time derivatives is also the basis of a flatness definition for discrete-time linear systems, i.e., systems with linear dynamics over discrete time, see e.g. Diwold et al. (2022). For these systems, flatness definitions either rely on forward time shifts or backward time shifts (Diwold et al. 2022). The key difference to automaton flatness is the presence of continuous-valued states and inputs in discrete-time linear systems.

4.2 Flatness of Hybrid Automata

Having defined flatness of automata, we can now proceed to define flatness for hybrid automata. To this end, we partition the elements of the hybrid automaton into three groups:

- The automaton $\{\mathsf{L}, \mathsf{E}, \mathsf{V}\}$
- The continuous dynamics Σ_l
- The switching dynamics $\mathcal{G}^x_\mathsf{e}, \mathcal{G}^\mathsf{v}_\mathsf{e}, L_\mathsf{e}$.

For the automaton and the continuous dynamics, we have flatness definitions that we can apply. But simply applying these to the hybrid automaton is not sufficient. In the following, we go through the three groups and define flatness in the hybrid automaton context. Finally, we synthesize a definition of flatness for the whole hybrid automaton. The results in this Section are adapted from the publications Zahn et al. (2022, 2024).

4.2.1 Automaton

The automaton of the hybrid automaton model consists of the locations $\mathsf{l} \in \mathsf{L}$, the edges $\mathsf{e} \in \mathsf{E}$ and the discrete inputs $\mathsf{v} \in \mathsf{V}$. The events that cause switching are defined by the guard sets $\mathcal{G}^x_\mathsf{e}, \mathcal{G}^\mathsf{v}_\mathsf{e}$. The automaton is flat according to Definition 9 if the automaton graph $\{\mathsf{L}, \mathsf{E}\}$ is strongly connected, and the events are controllable. The strong connectedness of the automaton graph is a structural property that can be verified from $\{\mathsf{L}, \mathsf{E}\}$ alone. The controllability of the events, however, can only be decided for guard sets $\mathcal{G}^\mathsf{v}_\mathsf{e}$, which are formulated in terms of the discrete input v. We assume that these events are controllable. For the guard sets $\mathcal{G}^x_\mathsf{e}$, we have to consider the continuous dynamics and the flow sets, in order to decide controllability.

4.2.2 Continuous Dynamics

The continuous dynamics Σ_l are defined at each location $\mathsf{l} \in \mathsf{L}$ by $\dot{x}_\mathsf{l} = f_\mathsf{l}(x_\mathsf{l}, u_\mathsf{l})$, $\mathcal{X}_\mathsf{l} \subset \mathbb{R}^{n_\mathsf{l}}$ and $\mathcal{U}_\mathsf{l} \subset \mathbb{R}^{m_\mathsf{l}}$. The state x_l can only evolve in the corresponding flow sets Flow_l defined in Section 3.2. These flow sets are shaped by constraints on the continuous system, represented by $\mathcal{X}_\mathsf{l}$, and by the guard sets. Thus, we define the continuous subsystems to be flat if the dynamics are flat in the whole flow set

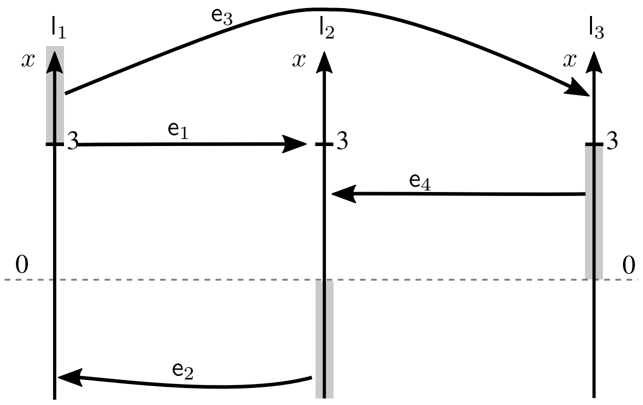


Abbildung 4.2: Illustration of Example 2. The grey areas are the guard sets.

Flow, according to Definition 3. This ensures the existence of the maps $F(\cdot)$, $\Phi(\cdot)$, $\Psi(\cdot)$, Equations (2.8). Furthermore, the flow sets have to be path-connected.

Definition 10 (Path-connectedness Manetti (2023)) *A topological space $\mathcal{X}$ is path-connected if, given any two points $x, y \in X$, there is a continuous mapping $\alpha : [0, 1] \to \mathcal{X}$, such that $\alpha(0) = x$ and $\alpha(1) = y$. Such an α is called a path from x to y.*

The path-connectedness is crucial in the presence of guard sets in the state space $\mathcal{X}$, because it ensures that the state set is not separated by a guard set, making one part of the space unreachable. This is illustrated in the following example.

Example 2 *Consider a hybrid system with three locations*

$\mathsf{l}_1 : \ \dot{x} = u$

$\mathsf{l}_2 : \ \dot{x} = u^3$

$\mathsf{l}_3 : \ \dot{x} = -u$

and state transitions with corresponding guards $\mathcal{G}^x_{\mathsf{e}}$

$$\begin{aligned}
&\mathsf{e}_1 : \ (\mathsf{l}_1, \mathsf{l}_2) \ \ x = 3\\
&\mathsf{e}_2 : \ (\mathsf{l}_2, \mathsf{l}_1) \ \ x < 0\\
&\mathsf{e}_3 : \ (\mathsf{l}_1, \mathsf{l}_3) \ \ x > 3\\
&\mathsf{e}_4 : \ (\mathsf{l}_3, \mathsf{l}_2) \ \ 0 < x < 3\,.
\end{aligned}$$

The value of the state x does not change when state switching occurs, i.e. $x^+ = L(x^-) = x^-$, the initial state is $x_{\mathsf{l}}^0 = 2$, $\mathsf{l}^0 = \mathsf{l}_2$. The automaton graph of the example is strongly connected and all continuous subsystems are controllable. However, not all states are reachable: The discrete state l_3 can only be reached from an initial state $x_{\mathsf{l}1} > 3$ and the states $x_{\mathsf{l}3} < 0$ could only be reached if the automaton was always initialized in this very set. Although the HA is not controllable, it is possible to obtain controllability by designing appropriate initial sets and final sets.

We define a *flat flow set* as a combination of the aforementioned properties.

Definition 11 (Flat flow set) *A flow set Flow with dynamics $\dot{x} = f(x,u)$ is a flat flow set, iff the set is path-connected, and there are flatness mappings according to Definition 3 for $\dot{x} = f(x,u)$ that are well defined in the whole set Flow.*

4.2.3 Switching Dynamics

The missing link in establishing flatness of the automaton is the controllability of the events with guard sets $\mathcal{G}_{\mathsf{e}}^x$. These events are controllable, iff reachability of the guard sets can be guaranteed. The reachability of a guard set has to be studied at its corresponding location l, where for $\mathcal{G}_{\mathsf{e}}^x$, l is the tail of e. A guard set is considered reachable if, for every admissible initial state in the location, there is a trajectory leading to that set. The possible initial states have been defined in Section 3.2 by the arrival set $\mathsf{Arrive}_{\mathsf{l}}$. If we can guarantee that all guard sets are reachable from the arrival sets, then the events are controllable.

4.3 Flat Hybrid Automata

Based on the Definitions 9, 11 and the consideration in Sections 4.2 above, we define flatness of hybrid automata in the following way:

Definition 12 (Zahn et al. (2024)) *A hybrid automaton is flat iff*

- *the automaton graph* $\{\mathsf{L}, \mathsf{E}\}$ *is strongly connected,*
- *all flow sets* Flow *are flat flow sets,*
- *for all edges* e_k *with autonomous transitions departing from a discrete state* l*, the respective guard sets* $\mathcal{G}^x_{\mathsf{e}}$ *can be reached from* $\mathsf{Arrive}_{\mathsf{l}}$.

For a hybrid automaton which is flat, we introduce the name Flat Hybrid Automaton (FHA). The three conditions in Definition 12 ensure flatness of the automaton and flatness of the continuous dynamics. This does not induce the same properties as differential flatness, but it leads to similar properties, some of which will be described in the following sections. We also note that the third property –reachability of the guard sets– is non-trivial to prove. We discuss this issue in Chapter 5.

The continuous dynamics and the guard sets and reset maps of FHA can be represented in terms of the flat state ζ instead of the state x. In the following, we assume that the flow sets Flow, the guard sets $\mathcal{G}^x_{\mathsf{e}}$ and the arrival set Arrive_L can be expressed in terms of equalities or inequalities

$$c(x) \leq 0 \tag{4.3a}$$

$$c_{\text{eq}}(x) = 0. \tag{4.3b}$$

To express these in terms of the flat state ζ, we can simply use Equation (2.8b) to replace x:

$$c(\Phi(\zeta)) \leq 0 \tag{4.4a}$$

$$c_{\text{eq}}(\Phi(\zeta)) = 0. \tag{4.4b}$$

The reset functions can be adapted in the same way, leading to

$$\Phi(\zeta^{+}) = L_{\mathsf{e}}(\Phi(\zeta^{-})). \tag{4.5}$$

We denote the transformed sets with a superscript ζ, i.e., $\mathsf{Flow}^{\zeta}, \mathcal{G}^{\zeta}, \mathsf{Arrive}^{\zeta}_{L}$. Also, the continuous dynamics $\dot{x} = f(x,u)$ in the FHA can be expressed in terms of the flat state, as described in Section 2. We express them using in feedforward-linearized form with new input ν,

$$\dot{\zeta} = g(\zeta, \nu). \tag{4.6}$$

The initial conditions in terms of the continuous state x also can be expressed in terms of the flat state ζ, as shown in Equation 2.11.

It is also possible to represent the automaton of the FHA by a complete automaton and the corresponding map as described in Section 4.1, but this is not helpful for numerical computation of trajectories. Therefore, we preserve the original automaton $\{\mathsf{L}, \mathsf{E}, \mathsf{V}\}$.

4.4 Flat Hybrid Automaton Design

We discuss the implications of the defined properties of FHA in the following. In continuous dynamical systems, differential flatness can be deduced from the mathematical model, but it can also be a guiding principle for design: One can make many systems flat by placing appropriate actuators, as discussed in Section 2. We adopt the same perspective for our FHA model. We discuss the defined properties to show how they can be assessed from the model, and we discuss their implications for system design.

In Definition 12, we defined three conditions for hybrid automaton flatness. In any practical application, we have to check the three conditions to see whether a HA is flat. Alternatively, we can use available degrees of freedom to adapt our model. In general, we can refine constraints and modify switching conditions and automaton graph of the application. In the following, we describe how to evaluate whether the conditions for flatness are met for the first two conditions. The third condition of reachability is discussed separately in Chapter 5.

4.4.1 Connectedness of Automaton Graph

The first condition of strong connectedness of the automaton graph can be assessed based on the automaton graph's adjacency matrix (Berman and Plemmons 1979). The information whether two locations are connected by an edge can be described by the so called adjacency matrix A of dimension $n_{\mathsf{l}} \times n_{\mathsf{l}}$, where n_{l} denotes the number of locations in the graph. The matrix consists of entries $a_{i,j}$ with i denoting the row index and j denoting the column index. An entry $a_{i,j}$ is 1 if there exists an edge $\mathsf{e} = (\mathsf{l}_i, \mathsf{l}_j)$ from l_i to l_j and it is 0 otherwise. The sum of the elements in the i-th row thus is the number of edges with tail l_i, the sum of the j-th column the number of edges with head l_j. From the matrix A we can compute whether the corresponding graph is strongly connected: If the inequality

$$(I + A)^{n_{\mathsf{l}}-1} > 0 \tag{4.7}$$

holds, then the graph is strongly connected (Berman and Plemmons 1979). In this equation, I denotes the identity matrix. The inequality > 0 means that every entry of the matrix $(I + A)^{n_{\mathsf{l}}-1}$ is greater than zero, i.e., the matrix is positive. This can be explained by the fact that for any natural number $m > 0$, the entries in the matrix $(I + A)^m$ denote the number of valid paths of length m between l_i and l_j in the graph $\{\mathsf{L}, \mathsf{E}\}$. This test is simple and can be implemented with standard linear algebra methods. However, it necessitates $n_\mathsf{l} - 1$ matrix multiplications, which have a complexity of $\mathcal{O}(n^3)$ in a non-optimized implementation.

Alternatively, strong connectedness of $\{\mathsf{L}, \mathsf{E}\}$ can be computed on the graph itself. A classic solution to the problem is Tarjan's algorithm (Tarjan 1972). The algorithm is based on a depth-first search that produces a partitioning of a directed graph into its strongly connected components in linear time. The complexity of Tarjan's algorithm is $\mathcal{O}(n_\mathsf{l} + n_\mathsf{e})$, i.e., linear in the number of locations and edges. If it partitions the graph into more than one component, it is not strongly connected, otherwise it is.

If the computations reveal that the automaton graph is not strongly connected, this indicates a problem with our model from an engineering point of view. The locations generally model different modes of operation, and the connectedness indicates that we can enter and leave every mode. If the graph is not strongly connected, there are either modes without access or exit, or there are disconnected groups of modes. In both cases, we have a problem with our model for automation and control design. We can then identify the missing transitions and decide whether to get rid of isolated locations, or whether we can add transitions. If neither option is possible, the application at hand can simply not be modeled as an FHA.

4.4.2 Flatness of Flow Set

The second condition states that the continuous dynamics in all locations $\mathsf{l} \in \mathsf{L}$ have to be differentially flat in the entire flow set, and that the flow set has to be path-connected. The differential flatness property should be known from the application at hand. When the continuous dynamics are differentially flat, one only has to make sure that no singularities are present in the maps in Definition 3. If singularities are present, they can be treated with one of the existing methods in the literature, as described in Section 2.

Path-connectedness of the flow set simply demands that the state space from our model does not include disconnected regions. In this way, we exclude all states that are inherently unreachable, and we make sure that the system constraints are sufficiently weak. Again, this property should be mostly known from the application. We can evaluate path-connectedness in the original state space in terms of x, because it is preserved under the smooth mapping $\Phi(\cdot)$.[2]

4.5 Flat Hybrid Automaton Reachability and Controllability

Differential flatness of continuous dynamical systems is closely related to controllability, as we discussed in Chapter 2. Therefore, we could assume that FHA are also controllable in some sense. For the automaton part of FHA, this is the case: By definition, all locations are reachable from all locations according to Definition 7. We can thus say that the automaton part of the FHA is controllable. For the entire hybrid state (l, ζ), however, this is certainly not the case in general. The reason is that in the presence of constraints, not all continuous states can be reached, as we will see in Chapter 5.

In our definition of HA, we define initial states *Init* and we have to guarantee that we can reach all locations from these initial states. The guard sets and the flow set can be seen as constraints on the continuous state. These constraints potentially make it difficult to know which states are reachable, just like for differentially flat systems in general (see e.g. Greco et al. (2022)). As a consequence, we should always assume that parts of the flow set are not reachable. From this we conclude that FHA are not controllable in the sense of Definition 2. However, for a practical application of the concept to technical systems, we may want to know that some sets of predefined states can be reached, which we define as our control goals. As we have introduced in Zahn et al. (2022), this can be formalized by defining a set

[2] In the hypothetical case that $\Phi(\cdot)$ is not continuous, as opposed to the Definition 3 of flatness we use, we would have to verify path-connectedness after the transformation.

of final states $\mathsf{Final}_l \subset \mathcal{X}_l$ to locations, which contain control targets. These can represent target positions, for example.

To use a model with final sets as an automation model, we want to guarantee the reachability of these sets. Due to the structure of FHA, this can be achieved by simply guaranteeing their reachability from the respective arrival set in their location. This is sufficient, because all arrival sets can be reached from the initial sets *Init* by construction.

Definition 13 (Reachable FHA) *A flat hybrid automaton with final sets Final_l is reachable, if each Final_l is reachable from the corresponding arrival set Arrive_l.*

This relation allows for the construction of automation models with decoupled computations for each location. In the following chapter, we show how to compute the reachability of sets for constrained differentially flat systems.

4.6 Summary and Discussion

In this Chapter, we introduce flatness for HA and the corresponding system class of FHA. Our definition is based on a decomposition of the HA into three parts: The automaton, the continuous dynamics, and the transitions. For each of the parts, we derive a property, which together yield flatness of the HA. This definition allows for the assessment of the flatness of HA for each of the parts separately. Strong connectedness of graphs for the automaton part and differential flatness of the continuous dynamics are well-known problems, for which solutions and algorithms exist in the literature. Reachability of the guards, however, is more specific to HA. Therefore, we study this problem in detail in Chapter 5.

The overarching idea of our definition of FHA is to carry over a few key properties of differential flatness to HA. As we discuss, it is not possible to preserve all properties because of the structural limitations of the combination of discrete and continuous dynamics. However, some properties persist: The locations of FHA are all reachable by design, allowing to realize all possible sequences of locations.

The corresponding continuous trajectories and the inversion to compute the inputs are guaranteed to exist. We leverage these properties in the examples in Chapters 6 and 7. A major advantage of differentially flat systems is their representation in terms of the flat output, which can follow any sufficiently smooth trajectory. The trajectories of FHA do not possess the same degree of freedom: The continuous trajectories are always constrained by the path through the automaton, and possibly constrained by the guards. This introduces a kind of hierarchy to the trajectories, in which the sequence of locations constrains the continuous trajectories. However, from an engineering point of view, even differentially flat systems are never free to follow any sufficiently smooth trajectory, because of technical and physical constraints. In a way, we can thus also interpret FHA as differentially flat systems with non-linear constraints.

5 Reachability of Guard Sets

The critical challenge in assessing flatness of HA according to Definition 12 lies in making sure that all guard sets in each location are reachable, see Section 4.4. The guard sets $\mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$, which are formulated in terms of the discrete input v, are always reachable, as defined above. For the other guard sets $\mathcal{G}_{\mathsf{e}}^{x}$, flatness as defined above demands the existence of trajectories from all points in the arrival set $\mathsf{Arrive}_{\mathsf{l}}$ to the guard set $\mathcal{G}_{\mathsf{e}}^{x}$. The material in this section is based on the results presented in Zahn et al. (2024), except for Section 5.3, which contains new material only.

We start by formalizing the problem of reachability of guard sets. To decide whether a guard set $\mathcal{G}_{\mathsf{e}}^{x}$ is reachable from an arrival set $\mathsf{Arrive}_{\mathsf{l}}$, we have to solve the following problem:

Problem 1 *For a differentially flat system $\dot{x} = f(x, u)$, $x \in \mathsf{Flow}$, $u \in \mathbb{R}^m$, with flat output z, given the set of initial states $\mathsf{Arrive}_{\mathsf{l}}$, the set of final states $\mathcal{G}_{\mathsf{e}}^{x}$ and a flow set Flow, is at least one point in the set $\mathcal{G}_{\mathsf{e}}^{x}$ reachable from $\mathsf{Arrive}_{\mathsf{l}}$ via trajectories (x^*, u^*), such that $(x^*(t), u^*(t)) \in \mathsf{Flow}$, $\forall t \in [0, t_f)$?*

The number of such problems one needs to solve to prove flatness for a given HA is equal to the number of edges $\mathsf{e} \in \mathsf{E}$ with guard set $\mathcal{G}_{\mathsf{e}}^{x}$. Each problem can be solved independently of the other problems.

For differentially flat systems, we can transform this problem to a simpler form by resorting to its linearized Brunovský form. As described in Chapter 2, a linearized differentially flat system is equivalent to a system in Brunovský form

$$\begin{aligned} \dot{\zeta}_{i,0} &= \zeta_{i,1} \\ &\vdots \\ \dot{\zeta}_{i,\kappa_i} &= \nu_i \quad i \in [1, \dots, m], \end{aligned} \tag{5.1}$$

consisting of m integrator chains. From this representation, we can see that any smooth path $\zeta^*_{i,0}$ is a valid trajectory of (5.1). In the following, we use the notation ζ_0 for the flat output, $\zeta_{i,\geq 1}$ for its derivatives up to κ_i, and ζ denoting the whole flat state vector

$$\zeta = [\zeta_{1,0}, \zeta_{1,1}, \dots, \zeta_{1,\kappa_1}, \zeta_{2,0}, \dots, \zeta_{m,0}, \dots, \zeta_{m,\kappa_m}]^\top. \tag{5.2}$$

We reformulate Problem 1 in terms of the flat state:

Problem 2 *For a system with flat state ζ, given the set of initial states $\mathsf{Arrive}^{\zeta}_{\mathsf{l}}$, a guard set $\mathcal{G}^{\zeta}_{\mathsf{e}}$ and a flow set Flow^{ζ}, for each $\zeta^*(0) \in \mathsf{Arrive}^{\zeta}_{\mathsf{l}}$, is there a function $\zeta^*(t), t \in [0, t_f]$, such that $\zeta^*(t_f) \in \mathcal{G}^{\zeta}_{\mathsf{e}}$, and $\zeta^*(t) \in \mathsf{Flow}^{\zeta}$, $\forall t \in [0, t_f)$?*

This formulation simplifies the problem: We do not need to consider the possibly non-linear system dynamics, but we can simply study a generic integrator chain. However, depending on the structure of the arrival set and on the constraints, this problem can still be hard to solve. Although any smooth path can be a trajectory of a flat system, identifying feasible trajectories under general constraints remains non-trivial.

Both problem formulations can be addressed with numerical methods. A common approach to compute reachable sets is numerical set propagation techniques as described, for example, in Althoff et al. (2021) and references therein. With these techniques, one can compute explicit representations of reachable sets for a given initial state or initial set. The reachable sets produced by these methods are either under- or over-approximations. Although these methods are quite mature, they remain computationally expensive. Moreover, in our case, we do not need to compute the reachable sets explicitly, but we only want to decide whether a set is reachable. For the problem at hand, we therefore develop a simple approach to decide reachability. We base our approach on the structural properties of differentially flat systems. Building on these, we propose lightweight numerical methods to compute reachability.

We study Problem 2 in flat coordinates. Thus, all constraints and all sets are expressed in terms of the flat state. Besides the number and lengths of the integrator chains, Problem 2 contains three elements: The flow set Flow^{ζ}, the arrival set $\mathsf{Arrive}_{\mathsf{l}}^{\zeta}$, and the guard set $\mathcal{G}_{\mathsf{e}}^{\zeta}$. A generic solution applicable to all configurations of these elements is either excessively restrictive in its conditions or computationally expensive. Therefore, we proceed by partitioning the problems into the three main parts: We first investigate the nature of the flow set and provide simple sufficient conditions for reachability of steady states. This is achieved by restricting the initial and final states to be steady states. We then investigate the arrival set and guard set. For these, we only study reachability from and of steady states, respectively. The idea is to derive simple rules or computations by accepting the trade-off to be more conservative.

We introduce additional notation in order to differentiate between the flat output and its derivatives in a concise way: The flow set Flow^{ζ} defines the domain of the flat state ζ. We want to consider its properties concerning the flat output z and its derivatives $\zeta_{i,\geq 1}$ separately. To this end, we split the flow set into two parts: We consider the set $\mathcal{Z}$ as defined in Section 2, such that the flat output z or, equivalently, ζ_0, is from the constrained set $\mathcal{Z}$. We further introduce the set Z', holding all derivatives of z, i.e., all $\zeta_{i,\geq 1}$. We thus have $\mathsf{Flow}^{\zeta} = \mathcal{Z} \times Z'$. These sets can be computed from the system constraints, as described in Section 4.3, Equations (4.4).

We further introduce open neighborhoods: The neighborhood of a point ζ is defined for some $\varepsilon > 0$ as

$$\mathcal{B}_{\zeta}^{\varepsilon} = \{\hat{\zeta} \in Z : \|\hat{\zeta} - \zeta\| < \varepsilon\}, \tag{5.3}$$

where $\|\cdot\|$ denotes the Euclidian norm. We also define a steady state neighborhood for a flat output z as

$$\mathcal{B}_{z,0}^{\varepsilon} = \{[z\top \hat{\zeta}\top]\top : \hat{\zeta} \in Z', \|\hat{\zeta}\| < \varepsilon\}. \tag{5.4}$$

This neighborhood expresses a ball around a steady state in the space of derivatives Z'.

5.1 Steady State Transitions

In a first step, we study Problem 2 under the assumption that the arrival set and the guard set only contain steady states, i.e., $\zeta_{i\geq 1}(t) = 0$ for $t = 0, t^f$. For this setting, Faulwasser et al. (2014) give a proof for constrained reachability, which relies on the notion and the properties of the set of steady states. We give an alternative proof that is tailored to our specific problem and is less technical. Under the described conditions, our problem reduces to the question, whether two steady states ζ_0^0, ζ_0^1 can be connected by a smooth path $p(\tau)$, which stays inside the flow set. As we described above, any smooth path is a valid trajectory for (5.1) if the derivatives are not constrained. A smooth path between ζ_0^0, ζ_0^1 always exists, if both lie in an open path-connected subset of Euclidean space, see Nerode and Greenberg (2022).

We further need to ensure that we can transform the path into a trajectory that respects the constraints on the derivatives $\zeta_{\geq 1}$. To transform the path to a trajectory of arbitrary duration t^f, we have a degree of freedom by scaling the interval $[0, 1]$ to physical time t. The magnitude of the derivatives can be modified by this scaling. Considering a linear scaling, i.e., $t = \tau t^f$, we can make the derivatives of $p(\tau)$ arbitrarily small by increasing t^f. Thus, for any smooth path in $p(\tau)$, we can limit the derivatives to arbitrarily small values without changing the shape of the path. Although the scaling can make the derivatives arbitrarily small, it cannot change their sign. Thus, we define another condition: For all $z \in \mathcal{Z}$, the flow set has to contain a neighborhood $\mathcal{B}^{\varepsilon}_{z,0}$ of $\zeta_{i \geq 1} = 0$. Under these conditions, we can guarantee reachability, which is summarized in the following theorem:

Theorem 3 *Given a differentially flat system with flat state ζ, and flow set Flow^{ζ}. For any two steady states $\zeta_0^0, \zeta_0^f \in \mathsf{Flow}^{\zeta}$, there is a feasible trajectory $\zeta^* \in \mathsf{Flow}^{\zeta}$ for $t \in [0, t^f]$ with $\zeta^*(0) = \zeta_0^0$, $\zeta^*(t^f) = \zeta_0^f$ if the set $\mathcal{Z}$ is an open path-connected set, and if Flow^{ζ} contains an open neighborhood of $\zeta_{i \geq 1} = 0$ for all $z \in \mathcal{Z}$, i.e., $\mathcal{B}^{\varepsilon}_{z,0} \subset Z'$.*

For a given open path-connected set Flow^{ζ}, we choose two points $\zeta_0^0, \zeta_0^f \in \mathsf{Flow}^{\zeta}$. We know from the properties of the set that a smooth path $p(\tau)$ with $p(0) = \zeta_0^0$, $p(1) = \zeta_0^f$ exists. We thus can construct a path between the two steady states with $p(0) = \zeta^0, p(1) = \zeta^f$. We also know that the derivatives of p, i.e., $\frac{\partial^d}{\partial \tau^d} p(\tau)$ exist. We replace τ with a linear scaling $\tau = \frac{t}{t_f}$. The new derivative of p is

$$\begin{aligned} \frac{\partial p}{\partial t} &= \frac{\partial p}{\partial \tau}\frac{\partial \tau}{\partial t} = \frac{\partial p}{\partial \tau}\frac{1}{t_f} \\ \frac{\partial^2 p}{\partial t^2} &= \frac{\partial \dot{p}}{\partial \tau}\frac{\partial \tau}{\partial t} = \frac{\partial^2 p}{\partial \tau^2}\frac{1}{{t_f}^2} \\ &\vdots \end{aligned}$$

Thus, any $t^f > 1$ can make the derivatives arbitrarily small, such that they are in the given neighborhood $\mathcal{B}^{\varepsilon}_{z,0} \subset Z'$. According to this result, we only have to make sure that for all $z \in \mathcal{Z}$, its steady state $[z, 0, 0, \ldots]$ is contained in the flow set, if the flow set is open. Otherwise, we can only guarantee the existence of a trajectory for states in the interior of the flow set, which is open by definition. We illustrate the result with an example.

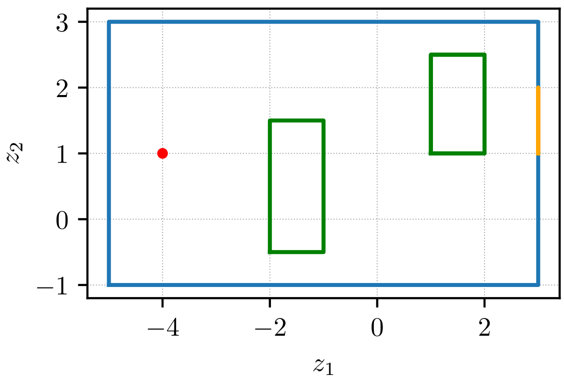


Abbildung 5.1: Flow set and obstacles for mobile robot example.

Example 3 *We have a mobile robot moving on a flat surface, with the task to reach a target set. We model the robot as a double integrator*

$$\dot{x} = v$$
$$\dot{v} = a$$

with position $x = [x_1, x_2]^\top$, velocity $v = [v_1, v_2]^\top$ and acceleration $a = [a_1, a_2]^\top$. The dynamics are flat with flat output $z = x$ and input a. The flat dynamics are $\ddot{z}_1 = a_1, \ddot{z}_2 = a_2$. We restrict the flow set as depicted in Figure 5.1 by an outer rectangular constraint and by two rectangular objects. The flow set can be described by a linear constraint $Ax - b \leq 0$, or in the flat output by $Az - b \leq 0$. The arrival set is a single starting position z^0 with $\dot{z}^0, \ddot{z}^0, \dots = 0$, the guard set $\mathcal{G}_\mathrm{e}^x$ to finish the task is a line, which can directly be expressed in terms of z, again with $\dot{z}, \ddot{z}, \dots = 0$. We also constrain the derivatives of the flat output by constraining the velocity v and the acceleration a with $-1 \leq v \leq 1$ and $-1 \leq a \leq 1$.

The set-up of the example is depicted in Figure 5.1. We can easily verify that the guard set is reachable by checking the conditions described in Theorem 3: The union of the flow set and the guard set is path connected in z, and the constraints on the derivatives include a neighborhood around 0. In this example, it becomes

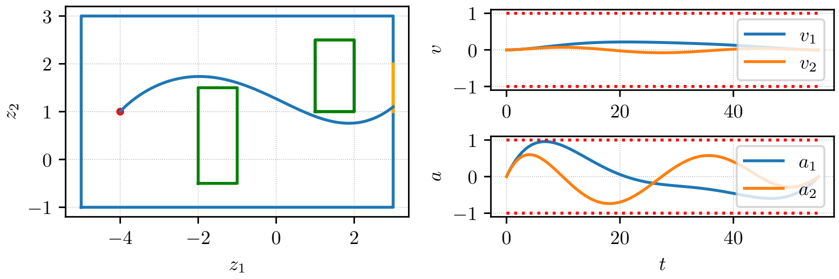


Abbildung 5.2: Left: Topology and path for mobile robot example. Right: Velocities and accelerations for the trajectory, with constraints as dotted red lines.

clear that with a restriction to allow only either positive or negative acceleration or velocity, we would not have reachability.

In Figure 5.2 we see an exemplary trajectory, which was computed as a Bézier curve path on $\tau \in [0,1]$. Scaling the end time to $t_f = 55$ leads to a trajectory that respects the constraints on v and a. From the results above, we can further see that for any starting position z^0 with $\dot{z}^0, \ddot{z}^0, \dots = 0$ inside the flow set, the guard set is reachable. We also see that we can make the constraints on v and a arbitrarily small, as long as they allow for positive and negative values. However, as a consequence, the durations of the trajectories then become longer.

5.2 Non-Steady Initial and Final States

In the previous section, we restricted the initial state to be a steady state. In a HA that models a dynamical system, this assumption may be too restrictive. We also want to consider cases in which the initial and final states are not steady states, i.e., with derivatives $\zeta_{i\geq 1} \neq 0$. However, when we allow the initial and the final states to have non-zero derivatives, we have to make sure that there are feasible trajectories that stay inside the flow set. For initial or final values near the boundary of the flow set, there may be no feasible trajectory that stays in the flow set, because of constraints in the higher-order derivatives. We illustrate this problem with a small example:

Example 4 *We consider a dynamical system in Brunovský form:*

$$\begin{aligned}\dot{\zeta}_0 &= \zeta_1\\ \dot{\zeta}_1 &= \zeta_2\\ \dot{\zeta}_2 &= \nu\end{aligned}$$

with the box constraints

$$\begin{aligned}-1 &\leq \zeta_0 \leq 1\\ -1 &\leq \zeta_1 \leq 1\\ -1 &\leq \zeta_2 \leq 1.\end{aligned}$$

For an initial value $\zeta_0(0), \zeta_1(0) = 0.9$, there is no feasible trajectory respecting the constraints: Choosing an input $\nu(t) \equiv 0$, and $\zeta_2(0) = -1$, such that we move away from the constraints in ζ_0, ζ_1 as fast as possible, the solution can be computed by

$$\zeta_1(t) = 0.9 + \int_0^t -1 d\tau = 0.9 - t \tag{5.5a}$$

$$\zeta_0(t) = 0.9 + \int_0^t 0.9 - \tau d\tau = 0.9 + 0.9t - \frac{1}{2}t^2. \tag{5.5b}$$

We see that in this case $\zeta_0(t)$ is not smaller than 1 for all times t. Thus, we cannot respect the constraints. We can also compute, for which initial values $\zeta_0(0), \zeta_1(0)$, it is possible to stay inside the constrained flow set: We look at

$$\begin{aligned}\zeta_2(0) = -1: \quad \zeta_0(t) =& \zeta_0(0) + \zeta_1(0)t - \frac{1}{2}t^2 \leq 1\\ \zeta_2(0) = 1: \quad \zeta_0(t) =& \zeta_0(0) - \zeta_1(0)t + \frac{1}{2}t^2 \geq -1\end{aligned}$$

Knowing that its maximum is reached at $t = \zeta_1(0)$, we compute the set of initial values $\zeta_0(0), \zeta_1(0)$, for which feasible trajectories exist. The result is depicted in Figure 5.3.

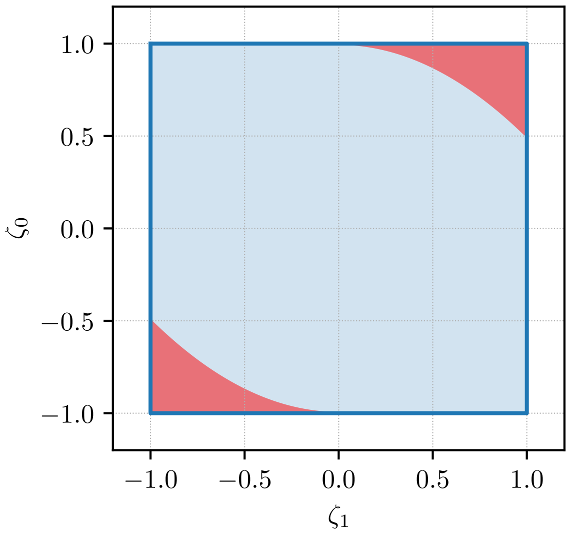


Abbildung 5.3: Valid initial states for Example 4 in blue. The red shaded area shows initial states for which no trajectory exists that does not violate the constraints.

This example demonstrates the necessity to make sure that any initial state that is not a steady state has a sufficient distance from the boundary of the flow set. The necessary distance depends on the magnitude and the direction of the derivatives. The same applies to final states. For these, we have to show that they are reachable from inside the flow set.

We can characterize the necessary distance algebraically by the general solution of Brunovský forms of order γ, given by the convolution

$$z^{(k)}(t) = z_0^{(k)} + \sum_{i=1}^{\gamma-k} \frac{1}{i!} z_0^{(i+k)} t^i + \int_0^t \frac{1}{(\gamma-k)!} (t-\tau)^{\gamma-k} \nu(\tau) \mathrm{d}\tau. \tag{5.6}$$

For initial and final states or sets and constraints, one can directly calculate whether feasible trajectories exist by looking at the extrema of each derivative and comparing them to the constraints. However, this computation becomes quite complicated when we allow for more complex constraints. Additionally, we have to specify an input function $\nu(t)$ to compute the solutions. The ideas we present in the following are based on the fact that an analytic solution exists, but our computations follow a different approach.

In the following, we use the terms *feasible initial state* and *feasible initial set* to characterize states. A feasible initial state is a state, for which a trajectory to a steady state inside the flow set exists. For a feasible initial set, this holds for all states in the set.

5.3 Numerical Reachability Computation

As we explained above, we want to complete the reachability computation by showing the reachability of steady states and the reachability of guards from steady states. The necessary computations are slightly different for initial and final states. For the set of initial states that are not steady states, we have to compute whether a steady state is reachable from the whole set, i.e., for all $\zeta \in \mathsf{Arrive}_{\mathsf{I}}^{\zeta}$. Our approach aims at two outcomes: First, we want to characterize the subset of the flow set from which a steady state is reachable. Second, we want to be able to evaluate for a given initial state ζ^0, whether a steady state is reachable.

For guard sets, which define the final states, the task at hand is different: Because we only have to make sure that some point in the set can be reached, we want to compute whether any state $\zeta^f \in \mathcal{G}^\zeta$ is reachable from any steady state $\zeta \in \mathsf{Flow}^\zeta$. We solve both problems by formulating them as optimization problems.

For the following derivations, we assume that all sets—arrival set, guard set, and flow set—are either convex or can be approximated as convex sets. We further assume that they can be represented as polyhedra with a halfspace representation

$$H\zeta - h \leq 0 \tag{5.7}$$

including the special case of the equality $H\zeta - h = 0$. We comment on the limitations of these assumptions below in Section 5.4.

The general idea is to formulate the reachability computation as a linear program of the form

$$
\begin{aligned}
\min \quad & \alpha && \text{(5.8a)}\\
\text{s.t.} \quad & \underline{\alpha} \leq \alpha \leq 0 && \text{(5.8b)}\\
& \text{Linear constraint on } \zeta^*(0) && \text{(5.8c)}\\
& \text{Linear constraint } \zeta^*(t) \in \mathsf{Flow}^\zeta && \text{(5.8d)}\\
& \text{Linear constraint on } \zeta^*(t^f). && \text{(5.8e)}
\end{aligned}
$$

We choose some $\underline{\alpha} < 0$ for the slack variable α, so that we can write the linear constraints as $H\zeta - h \leq \alpha$. This allows for two types of computation: First, we can check the feasibility of the problem, which is sufficient for reachability. Second, we can compute a trajectory respecting the constraints, as we will see in Section 5.3.3.

In the following, we present two different methods to compute reachability for non-steady initial and final states.

5.3.1 Bézier Curves

Our first approach is based on a Bézier curve representation of the trajectories. The following definitions are from Marcucci (2024). A Bézier curve $\gamma(\tau)$ over an interval $\tau \in [a, b]$ is defined as a linear combination of Bernstein polynomials. The Bernstein polynomials of order N are

$$
\beta_n(\tau) := \binom{N}{n} \left(\frac{\tau - a}{b - a}\right)^n \left(\frac{b - \tau}{b - a}\right)^{N-n}, \quad \forall n = 0, \dots, N. \tag{5.9}
$$

The corresponding Bézier curve is defined by

$$
\gamma(\tau) := \sum_{n=0}^{N} \beta_n(\tau) \gamma_n \tag{5.10}
$$

with control points γ_n. The control points γ_n can be real-valued vectors for curves in higher dimensions. A fundamental property of Bézier curves is their convex hull property: The whole curve is contained in the convex hull of its control points, i.e.,

$$\gamma(\tau) \in \text{conv}(\{\gamma_0, \dots, \gamma_N\}), \quad \forall \tau \in [a, b]. \tag{5.11}$$

Another useful property is that the initial and final values of the curve are defined by the first and last control points:

$$\gamma(a) = \gamma_0, \quad \gamma(b) = \gamma_N. \tag{5.12}$$

The derivative of a Bézier curve is again a Bézier curve of reduced order:

$$\frac{\partial \gamma}{\partial \tau} = \sum_{n=0}^{N-1} \beta(\tau)_n \gamma_n^{(1)} \tag{5.13}$$

$$\gamma_n^{(1)} = \frac{N}{b-a}(\gamma_{n+1} - \gamma_n). \tag{5.14}$$

A Bézier curve and its derivatives are thus completely defined by the control points and the degree. The control points of the derivatives are simply linear combinations of the original control points. In the following, we only consider intervals $a = 0, b = t^f$.

We formulate the constraints of a trajectory in terms of the control points of a given one-dimensional Bèzier curve of fixed order and interval. To this end, we introduce matrices Δ^0, Δ^f to calculate the initial and final states of the curve

$$\Delta^0 \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_N \end{bmatrix} = \begin{bmatrix} \gamma(0) \\ \gamma^{(1)}(0) \\ \gamma^{(2)}(0) \\ \vdots \\ \gamma^{(N)}(0) \end{bmatrix}, \quad \Delta^f \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_N \end{bmatrix} = \begin{bmatrix} \gamma(t^f) \\ \gamma^{(1)}(t^f) \\ \gamma^{(2)}(t^f) \\ \vdots \\ \gamma^{(N)}(t^f) \end{bmatrix}. \tag{5.15}$$

We also define matrices $\Delta^{(i)}$ to calculate the control points of the derivatives:

$$\Delta^{(i)} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_N \end{bmatrix} = \begin{bmatrix} \gamma_0^{(i)} \\ \gamma_1^{(i)} \\ \vdots \\ \gamma_{N-i}^{(i)} \end{bmatrix}. \tag{5.16}$$

With these matrices, we can formulate trajectory constraints to compute reachability. With the matrices Δ^0, Δ^f we can constrain the initial and the final states, e.g.,

$$\Delta^0 \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_N \end{bmatrix} = \zeta^*(0). \tag{5.17}$$

The matrices $\Delta^{(i)}$ allow us to constrain the trajectory and its derivatives. Assuming that the flow set is given as a convex set in halfspace representation, we obtain linear constraints of the form

$$H\Delta^{(i)} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_N \end{bmatrix} - h \leq 0. \tag{5.18}$$

These constraints encode that the control points $\gamma^{(j)}$ are within the constraints. Note that, by (5.11), $\frac{d^i}{dt^i}\zeta^*(t)$ stays within the convex hull of these control points. With the constraint equations, we can now formulate the reachability computation for initial and final states.

The set of initial states, from which a steady state can be reached, can be characterized by combining Equation (5.18) with a constraint

$$\Delta^f_{\geq 0} \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_N \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{5.19}$$

which restricts the final state to be a steady state. The matrix $\Delta^f_{\geq 0}$ is the matrix Δ^0 without the first row, such that it can be used to compute all derivatives $\zeta_{\geq 0}$ of the final state. As both expressions are linear inequalities, we can characterize the set of feasible initial states in terms of the way points as

$$\hat{H}\bar{\gamma} - \hat{h} \leq 0, \tag{5.20}$$

with $\hat{H}$ combining matrices $\Delta^0, H\Delta^{(i)}$ and the vector of way points $\bar{\gamma}$. To determine if a steady state can be reached from a given initial state $\zeta^*(0)$, we select an end time t^f.

This leads to the linear program

$$\begin{aligned} \min_{\bar{\gamma},\alpha} \quad & \alpha && (5.21a)\\ \text{s.t.} \quad & \underline{\alpha} \leq \alpha \leq 0 && (5.21b)\\ & \Delta^0\bar{\gamma} = \zeta^*(0) && (5.21c)\\ & \hat{H}\bar{\gamma} - \hat{h} \leq \alpha && (5.21d)\\ & \Delta^f_{\geq 0}\bar{\gamma} = [0, 0, \ldots, 0]\top. && (5.21e) \end{aligned}$$

The initial state $\zeta^*(0)$ is feasible if the optimization problem (5.21) has a solution.

For the reachability computation of final states, we can adapt problem (5.21). Instead of restricting the final state to be a steady state, we restrict the initial state.

The final state is restricted to the guard set. Again, we assume that the guard set can be represented as a linear inequality. The corresponding problem reads

$$\begin{aligned}
\min_{\bar{\gamma},\alpha} \quad & \alpha && \text{(5.22a)}\\
\text{s.t.} \quad & \underline{\alpha} \leq \alpha \leq 0 && \text{(5.22b)}\\
& \Delta^0 \bar{\gamma} = [\cdot, 0, 0, \dots]^\top && \text{(5.22c)}\\
& \hat{H}\bar{\gamma} - \hat{h} \leq \alpha && \text{(5.22d)}\\
& \Delta^f \bar{\gamma} \in \mathcal{G}. && \text{(5.22e)}
\end{aligned}$$

If problem (5.22) has a solution, the guard set is reachable from a steady state. Again, it is sufficient to show that the feasible set is non-empty.

Using Bézier curves for the computation introduces a certain degree of conservatism, because the trajectories are constrained to be polynomials of a certain degree. Moreover, the input ν is therefore a polynomial of lower degree. However, the approach is straightforward to implement and to compute. In the following section, we propose another approach based on a discrete-time representation of the dynamics.

5.3.2 Exact Discretization

For a constant input ν, we can calculate the system trajectory for a given Brunovský form and initial states with Equation (5.6). Assuming piecewise constant input signals, we can discretize the exact trajectory by solving the integral. This yields a piecewise polynomial. For the constant input ν^*, we can calculate the state at time t^f

$$\zeta(t^f) = \underbrace{\begin{bmatrix} 1 & t^f & \frac{{t^f}^2}{2} & \dots & \frac{{t^f}^{\kappa-1}}{(\kappa-1)!} \\ & 1 & t^f & \dots & \frac{{t^f}^{\kappa-2}}{(\kappa-2)!} \\ & & \ddots & & \vdots \\ & & & & 1 \end{bmatrix}}_{A_{t^f}} \zeta^0 + \underbrace{\begin{bmatrix} \frac{{t^f}^{\kappa}}{\kappa!} \\ \frac{{t^f}^{\kappa-1}}{(\kappa-1)!} \\ \vdots \\ t^f \end{bmatrix}}_{B_{t^f}} \nu^*. \tag{5.23}$$

This solution can be generalized to a sequence of constant inputs $\{\nu^i\}_{i=0}^k$ with fixed time intervals d

$$\zeta[k+1] = {A_d}^{k+1}\zeta[0] + \sum_{i=0}^{k} A_d^{k-1-i} B_d \nu[i], \tag{5.24}$$

which can also be written as

$$\zeta[k+1] = \hat{A}_{d,k} \begin{bmatrix} \zeta[0] \\ \nu[0] \\ \nu[1] \\ \vdots \\ \nu[k] \end{bmatrix}. \tag{5.25}$$

The matrix A_d^k can be calculated without matrix multiplication as

$$A_d^k = \begin{bmatrix} 1 & d*k & \frac{(d*k)^2}{2} & \dots & \frac{(d*k)^{\kappa-1}}{(\kappa-1)!} \\ & 1 & d*k & \dots & \frac{(d*k)^{\kappa-2}}{(\kappa-2)!} \\ & & \ddots & & \vdots \\ & & & & 1 \end{bmatrix}. \tag{5.26}$$

With the discretized system dynamics, we can compute reachable states. For the computation, we have to choose a step size d and a maximum step number k_{max}, such that $t^f = d * k_{\text{max}}$. A trajectory is then defined by the initial state $\zeta[0]$ and the sequence of inputs $\{\nu^i\}_{i=0}^{k_{\text{max}}-1}$. Each state $\zeta[k]$ is a linear combination of the initial state and the inputs up to index $k-1$. We can thus constrain each state separately by a linear constraint

$$H\zeta[k] - h \leq 0, \tag{5.27}$$

where $\zeta[k]$ is given by Equation (5.24). To characterize the set of feasible initial states, we can derive an inequality

$$\hat{H}\hat{A}_d \begin{bmatrix} \zeta[0] \\ \nu[0] \\ \nu[1] \\ \vdots \\ \nu[k_{\text{max}}-1] \end{bmatrix} - \hat{h} \leq 0, \tag{5.28}$$

in which $\hat{H}, \hat{h}$ are matrices characterizing the constraints, and $\hat{A}_d$ is a concatenation of all matrices $\hat{A}_{d,k}$.

In order to verify a given initial state, we can again solve a linear program

$$\begin{aligned} \min_{\bar{\nu},\alpha} \quad & \alpha && (5.29a) \\ \text{s.t.} \quad & \underline{\alpha} \leq \alpha \leq 0 && (5.29b) \\ & \hat{H}\hat{A}_d \begin{bmatrix} \zeta[0] \\ \bar{\nu} \end{bmatrix} - \hat{h} \leq \alpha && (5.29c) \\ & \hat{A}_{d,k_{\text{max}}-1} \begin{bmatrix} \zeta[0] \\ \bar{\nu} \end{bmatrix} = [\cdot, 0, 0, \dots]\top, && (5.29d) \end{aligned}$$

or simply check that the interior of the feasible set is non-empty, i.e., that the problem has a solution. The dot $\cdot$ in Equation (5.29d) denotes that the first entry remains unconstrained, because it is already constrained to lie in the flow set by Equation (5.29c).

For the final set, we can also formulate a corresponding linear program as

$$\begin{aligned}
\min_{\bar{\nu},\alpha} \quad & \alpha && (5.30a)\\
\text{s.t.} \quad & \underline{\alpha} \leq \alpha \leq 0 && (5.30b)\\
& \zeta[0] = [\cdot, 0, 0, \dots]\top && (5.30c)\\
& \hat{H}\hat{A}_d \begin{bmatrix} \zeta[0] \\ \bar{\nu} \end{bmatrix} - \hat{h} \leq \alpha && (5.30d)\\
& \hat{A}_{d,k_{\max}-1} \begin{bmatrix} \zeta[0] \\ \bar{\nu} \end{bmatrix} \in \mathcal{G}. && (5.30e)
\end{aligned}$$

5.3.3 Example

We apply the two reachability computation methods to Example 4. For the third order Brunovský system with box constraints, we evaluate which areas are reachable from which initial position. The steady state reachability is given, because the box constraint $-1 \leq \zeta_i \leq 1$ clearly includes a neighborhood of 0.

The corresponding sets of initial and final states can be formulated as described above. We use the resulting inequalities to compute the subset of feasible initial states. To this end, we discretize the plane ζ_0–ζ_1, as depicted in Figure 5.4. Each point represents an initial state. For each initial state, we solve the optimization problems for Bézier curves (5.21) and for the exact discretization (5.29) with fixed end time $t = 10$. For each method, we select three distinct parameters: we compute Bézier solutions for three different orders N, and we compute discretized solutions for three different time intervals d. The parameters are chosen such that the computations are of similar time for the row in Figure 5.5. We solve the

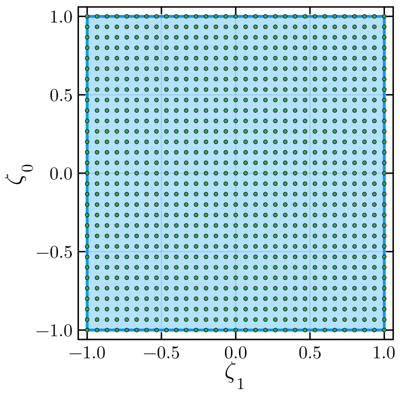


Abbildung 5.4: Grid points for reachability computation of area in box constraints.

optimization problems with the open-source solver IPOPT (Wächter and Biegler 2006). The results and the corresponding parameters for the computations are depicted in Figure 5.5. In the plots, the analytical solution for the feasible initial set from Example 4 is given by the blue shaded area. Each dot marks a feasible initial state. We can see that for lower order N and larger time steps d, the approximated feasible set is smaller than the original set. Almost all points lie in the analytical feasible region. For the Bézier method, this is guaranteed by the convex hull property. For the discretized solution with a larger time step, there is a point outside the true feasible region. We analyze the corresponding solution below.

As the order N increases, or the time step d decreases, we can see that the approximated feasible set approaches the analytical set. For the larger red dots, we also provide the solution for the trajectories in Figure 5.6. We also show corresponding trajectories to the initial states marked as red dots in Figure 5.5 to illustrate the effect of the different methods and parameters. In Figure 5.6, we see trajectories for the different methods and parameters. The plots in Figures 5.6a, 5.6c show trajectories as Bézier curves with the corresponding way points. For the low number of way points in Figure 5.6a, we can see that the curves stay

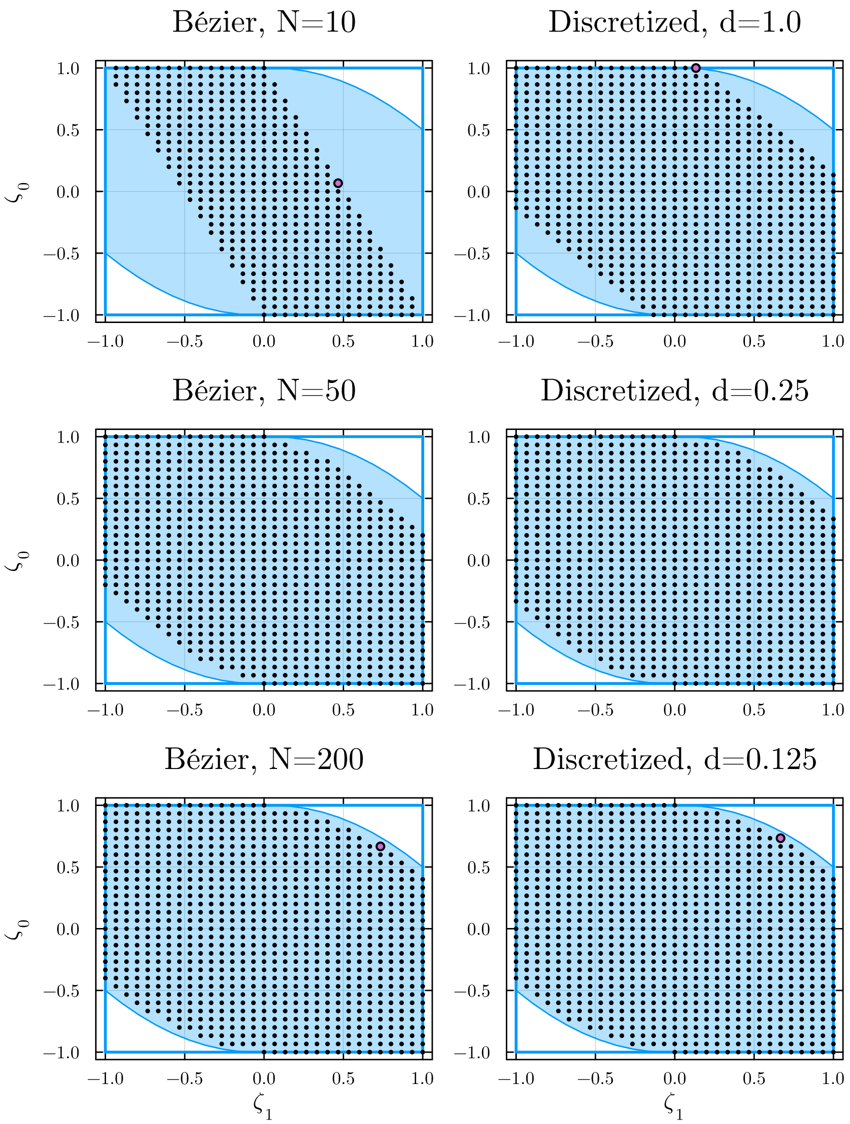


Abbildung 5.5: Feasible initial points for different parameters. The solutions for the initial values marked as large red dots are depicted in Figure 5.6 below.

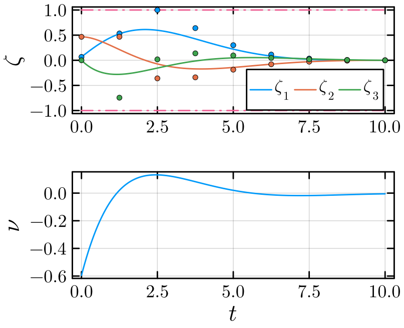
far away from the constraints, although a way point lies on the constraints. This explains the quite conservative feasible region in Figure 5.5 for $N = 10$. In Figure 5.6c, the way points and the trajectories are almost identical, which illustrates the well-known fact that the curve and the way points converge for high orders.

In the Figures 5.6b and 5.6d, we see trajectories of the discretized method. For the larger time step $d = 1$, we choose an initial condition which is infeasible, but was labeled feasible by the computation (see top right in Figure 5.5 for the initial condition). This can be seen by the dashed lines, which are the continuous-time trajectories corresponding to the solution for the piecewise constant input function ν plotted below. This is due to the fact that the discretized solution only considers the time points $t = kd$, but it ignores the behavior in between. We can see that the continuous solution violates the constraints. In Figure 5.6d, we see a trajectory for the smallest time step, which is very close to the continuous solution with the corresponding inputs. Comparing the Bézier solution 5.6c and the discretized solution 5.6d, we observe that the discretized solution is more aggressive in the input, and consequently in ζ_3. The Bézier solution cannot be as aggressive, because the input is a continuous polynomial. One can thus expect the Bézier approach to be generally more conservative. Despite these differences, the trajectories for large N and small d are quite similar.

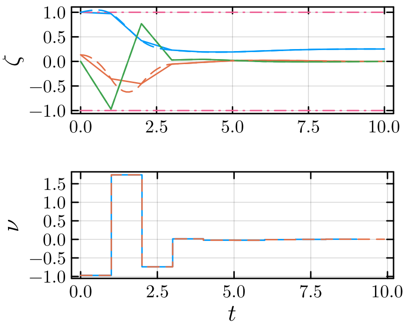
5.3.4 Discussion of the Results

The two methods we presented are derived for convex sets in the half-plane representation. In this polyhedral form, the constraints can be projected to feasible initial conditions, eliminating all other variables. However, such projections are numerically expensive and were not investigated in the present work.

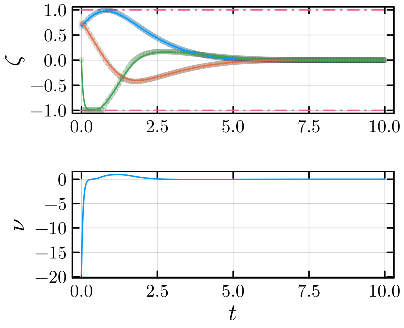
Although we restricted the application of the methods to constraints in the form of polyhedra, they can easily be used for arbitrary convex sets. The discretized approach can also be applied to non-convex constraints, because convexity is



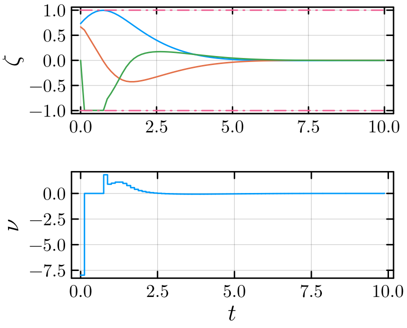
(a) Bézier trajectory with $N = 10$ way points.



(b) Trajectory of discretized dynamics with $d = 1$.



(c) Bézier trajectory with $N = 200$ way points.



(d) Trajectory of discretized dynamics with $d = 0.125$.

Abbildung 5.6: Trajectory plots: (a) and (c) show the effects of increasing the order N for Bézier curves. (b) shows a discretized solution for large time steps and the continuous-time solution for the corresponding piecewise constant input (dashed lines), which violates the constraints. (d) shows a solution for small step sizes.

not explicitly used there. However, the optimization problem then becomes non-convex and non-linear. Such problems can also be solved with IPOPT (Wächter and Biegler 2006).

If the end time t_f is not fixed but a decision variable, the Bézier approach becomes a bilinear program, but it remains convex. The discretized approach can not directly incorporate a free end time, but we can let the time step d become a (constrained) free variable. The optimization problem then becomes non-linear, because d enters the matrix A_d^k in polynomial form (see Equation (5.26)).

Although these approaches remain approximations of the exact feasible sets, we argue that they can be even more useful than any analytical solution: In order to implement trajectory computations, we generally want to use numerical methods anyway. If we have proven reachability by a method with a particular set of parameters, we can use this same method to compute trajectories with guaranteed feasibility. In this case, the analytical set would not guarantee feasibility.

5.4 Summary and Discussion

In this Chapter, we investigate how to evaluate reachability of guard sets. To this end, we explore two different approaches: We present theoretical results in the form of necessary and sufficient conditions, and we demonstrate numerical computations. Depending on the system at hand and its application, we may use one of these or a combination of both. Their great advantage over verification methods for HA is their modularity. We do not need to evaluate the whole HA, but we can do our reachability assessments for each location separately. Additionally, we leverage the specific properties of the differentially flat continuous dynamics to further simplify the computations. However, this does not make the problem trivial because of the potential complexity of constraints. For non-convex constraints or non-linear constraints, we may have to resort to approximations of the sets. An advantage of our approach is the direct carry-over to applications: Once we have proven reachability for a certain approach, e.g., Bézier curves, we can use these in a control environment. Because they under-approximate the actual reachable states, the result is much more practical than a theoretical proof of reachability.

For systems with low order or mainly controlled transitions, reachability may be straightforward to check. Once we have modeled a system as FHA and verified reachability, we can explore the possibilities for automation and control. In the next chapter, we demonstrate the developed theory by applying it to examples from different engineering domains.

6 FHA Modeling and Control

In this chapter, we show how to model a system as FHA, and we demonstrate the versatility of the model. We present step-by-step demonstrations on how to derive the FHA model and what design choices are associated with the derivation. Furthermore, we describe how to use the model for inversion and feedforward, and what the potential challenges are. We investigate four examples from different engineering domains: A switched Direct Current (DC) circuit and an Alternating Current (AC) grid model from the electrical domain, a simple tank example, and a robotic manipulator from the mechanical domain. Some of the material in this section is unpublished. We only explicitly mark the published parts.

6.1 Modeling of Systems as FHA

The FHA model allows us to describe the hybrid dynamics of different types of systems. Because their continuous dynamics are always differentially flat, they mainly differ in the nature of the hybrid elements, namely, the locations and the transitions. We distinguish two major categories of hybrid phenomena:

1. **Physical discontinuities.** These include switches, impacts, jumps in states, and discontinuities in the continuous dynamics. Transitions caused by phenomena of this type change the dynamics, the state, or both.
2. **Operation modes.** The locations of the FHA can also model different operation modes, which may share the same or similar continuous dynamics, but differ in constraints and targets. This allows to model processes and complex tasks.

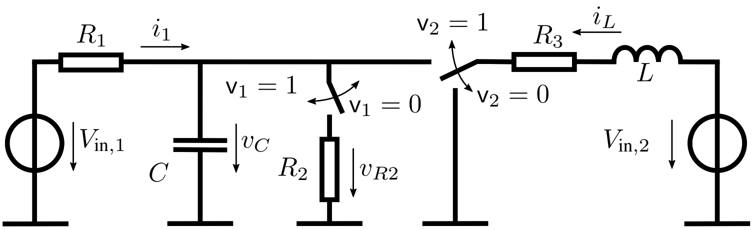


Abbildung 6.1: DC network with loads R_1, R_2, R_3, capacitor C, inductance L, power sources $V_{\text{in},1}, V_{\text{in},2}$ and two switches $\mathsf{v}_1, \mathsf{v}_2$.

These two categories are not mutually exclusive, and many systems include both, as will become clear from the examples.

In the sections below, we present four examples from three different technical domains. Besides being from different technical domains, the examples also exhibit different types of transitions. We explain the modeling of each example, and we discuss various aspects of control.

6.1.1 Electrical DC Switched Circuit

The first example we present is an electrical DC network. The example is inspired by Gensior et al. (2006), and it has been published in a similar form in Zahn et al. (2022). The electrical DC network in Fig. 6.1 has two variable voltage sources $V_{\text{in},1}$ and $V_{\text{in},2}$, three purely resistive loads R_1, R_2 and R_3, and two switches $\mathsf{v}_1 \in \{0,1\}$ and $\mathsf{v}_2 \in \{0,1\}$. We assume ideal switches that open and close infinitely fast without causing disturbances in voltage or current. The continuous inputs $V_{\text{in},1}$ and $V_{\text{in},2}$ control the voltage of the capacitor C and the current of load R_3, i_L. The states of the system are i_L and v_C.

The following equations describe the dynamics of the system, combining the continuous states and the discrete inputs:

$$\begin{aligned} L\frac{\mathrm{d}i_L}{\mathrm{d}t} &= V_{\text{in},2} - R_3 i_L - \mathsf{v}_2 v_C \\ C\frac{\mathrm{d}v_C}{\mathrm{d}t} &= \left(V_{\text{in},1} - \left(1 + \mathsf{v}_1\frac{R_1}{R_2}\right) v_C\right) R_1^{-1} + \mathsf{v}_2 i_L \end{aligned} \tag{6.1}$$

We model the switches as automated switches that open for low voltages or currents. For high voltages or currents, they are closed. Formally stated, their state depends on limit values:

if $v_C < v_0$ then $\mathsf{v}_1 = 0$, else $\mathsf{v}_1 = 1$
if $i_L < i_0$ then $\mathsf{v}_2 = 0$, else $\mathsf{v}_2 = 1$

The positions of the switches only depend on the two states v_C and i_L. Therefore, v_1 and v_2 are not modeled as inputs.

FHA Model

The hybrid nature of the system is induced by the switches. Operating a switch leads to two changes: First, there is a discontinuity in the state or in the continuous input. This will become clear from the equations below. Furthermore, the dynamics of the system are different for different switch positions. When the position is $\mathsf{v}_1, \mathsf{v}_2 = 0$, terms in the dynamics (6.1) are canceled. We show in the following how to model the system as a hybrid automaton, and we prove its flatness. For this example, we do not constrain the set of initial states. Thus, all values i_l^0, v_C^0 that are inside the flow sets are admissible initial conditions.

Permuting v_1 and v_2 in system (6.1) by their values $0, 1$ yields four different continuous dynamical systems. We model each continuous dynamics as a separate location l_i. All the continuous dynamics are flat with flat outputs $z_1 = v_C$ and $z_2 = i_L$, which are also the only flat states $\zeta_1 = v_C$, $\zeta_2 = i_L$. The flat dynamics in terms of the flat states and the continuous inputs $u_1 = V_{\text{in},1}$ and $u_2 = V_{\text{in},2}$ are the following:

$$\begin{aligned}
\mathsf{l}_1 : \quad & \mathsf{v}_1 = 0, \mathsf{v}_2 = 0 \\
& \dot{\zeta}_1 = (u_1 - \zeta_1)(R_1 C)^{-1} && \text{(6.2a)} \\
& \dot{\zeta}_2 = (u_2 - R_3\zeta_2)L^{-1} && \text{(6.2b)} \\
\mathsf{l}_2 : \quad & \mathsf{v}_1 = 0, \mathsf{v}_2 = 1 \\
& \dot{\zeta}_1 = (u_1 - \zeta_1)(R_1 C)^{-1} + \zeta_2 C^{-1} && \text{(6.2c)} \\
& \dot{\zeta}_2 = (u_2 - R_3\zeta_2 - \zeta_1)L^{-1} && \text{(6.2d)} \\
\mathsf{l}_3 : \quad & \mathsf{v}_1 = 1, \mathsf{v}_2 = 0 \\
& \dot{\zeta}_1 = \left(u_1 - \left(1 + \frac{R_1}{R_2}\right)\zeta_1\right)(R_1 C)^{-1} && \text{(6.2e)} \\
& \dot{\zeta}_2 = (u_2 - R_3\zeta_2)L^{-1} && \text{(6.2f)} \\
\mathsf{l}_4 : \quad & \mathsf{v}_1 = 1, \mathsf{v}_2 = 1 \\
& \dot{\zeta}_1 = \left(u_1 - \left(1 + \frac{R_1}{R_2}\right)\zeta_1\right)(R_1 C)^{-1} + \zeta_2 C^{-1} && \text{(6.2g)} \\
& \dot{\zeta}_2 = (u_2 - R_3\zeta_2 - \zeta_1)L^{-1} && \text{(6.2h)}
\end{aligned}$$

In the following, we use i_L and v_C instead of the flat output or state, for the sake of readability. From all discrete states, it is possible to switch to every other discrete state. Therefore, the automaton graph in Fig. 6.2 of the hybrid system is complete.

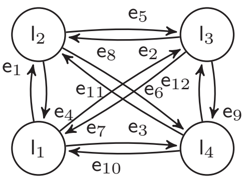


Abbildung 6.2: Automaton graph of the DC network model.

The inputs for each location can be computed by:

$$\mathsf{l}_1: \quad u_1 = CR_1\dot{v}_C + v_C \tag{6.3a}$$

$$u_2 = L\dot{i}_L + R_3 i_L \tag{6.3b}$$

$$\mathsf{l}_2: \quad u_1 = \left(C\dot{v}_C - i_L\right) R_1 + v_C \tag{6.3c}$$

$$u_2 = L\dot{i}_L + R_3 i_L + v_C \tag{6.3d}$$

$$\mathsf{l}_3: \quad u_1 = CR_1\dot{v}_C + \left(1 + \frac{R_1}{R_2}\right) v_C \tag{6.3e}$$

$$u_2 = L\dot{i}_L + R_3 i_L \tag{6.3f}$$

$$\mathsf{l}_4: \quad u_1 = \left(C\dot{v}_C - i_L\right) R_1 + \left(1 + \frac{R_1}{R_2}\right) v_C \tag{6.3g}$$

$$u_2 = L\dot{i}_L + R_3 i_L + v_C \tag{6.3h}$$

The transitions are defined by the values of the flat outputs, as defined above. Consequently, the guard sets can be formulated in terms of i_L and v_C. The transitions corresponding to the locations defined above and to the numbering in Figure 6.2 are listed in Table 6.1. The corresponding reset maps can be computed from the Equations (6.2). When the switches open or close, the flat outputs i_L and v_C do not change, but according to the equations, either their derivatives, or the inputs have to jump. We show this for the transition e_1 from l_1 to l_2 with jump set $\mathcal{G}_{\mathsf{e}_1}$. The reset map is simply

$$i_L^- = i_L^+ \tag{6.4a}$$

$$v_C^- = v_C^+. \tag{6.4b}$$

Tabelle 6.1: Guards for the electrical DC switched circuit.

$\mathcal{G}_{\mathrm{e1}}$:	$\mathsf{l}_1 \rightarrow \mathsf{l}_2$	$i_L = i_0$
$\mathcal{G}_{\mathrm{e2}}$:	$\mathsf{l}_1 \rightarrow \mathsf{l}_3$	$v_C = v_0$
$\mathcal{G}_{\mathrm{e3}}$:	$\mathsf{l}_1 \rightarrow \mathsf{l}_4$	$i_L = i_0 \cap v_C = v_0$
$\mathcal{G}_{\mathrm{e4}}$:	$\mathsf{l}_2 \rightarrow \mathsf{l}_1$	$i_L < i_o$
$\mathcal{G}_{\mathrm{e5}}$:	$\mathsf{l}_2 \rightarrow \mathsf{l}_3$	$i_L < 0 \cap v_C = v_0$
$\mathcal{G}_{\mathrm{e6}}$:	$\mathsf{l}_2 \rightarrow \mathsf{l}_4$	$v_C = v_0$
$\mathcal{G}_{\mathrm{e7}}$:	$\mathsf{l}_3 \rightarrow \mathsf{l}_1$	$v_C < v_0$
$\mathcal{G}_{\mathrm{e8}}$:	$\mathsf{l}_3 \rightarrow \mathsf{l}_2$	$v_C < v_0 \cap i_L = i_0$
$\mathcal{G}_{\mathrm{e9}}$:	$\mathsf{l}_3 \rightarrow \mathsf{l}_4$	$i_L = i_0$
$\mathcal{G}_{\mathrm{e10}}$:	$\mathsf{l}_4 \rightarrow \mathsf{l}_1$	$v_C < v_0 \cap i_L < i_0$
$\mathcal{G}_{\mathrm{e11}}$:	$\mathsf{l}_4 \rightarrow \mathsf{l}_2$	$v_C < v_0$
$\mathcal{G}_{\mathrm{e12}}$:	$\mathsf{l}_4 \rightarrow \mathsf{l}_3$	$i_L < i_0$

The derivatives from the Equations (6.2c),(6.2d) in l_2 after the transition are defined by

$$\begin{aligned}\dot{v}_C^+ &= \left(u_1^+ - v_C^+\right)(R_1 C)^{-1} + i_L^+ C^{-1}\\ \dot{i}_L^+ &= (u_2^+ - R_3 i_L^+ - v_C^+) L^{-1}.\end{aligned}$$

The change in the derivative or in the input is revealed by comparing the equations above to the equation before the transition:

$$\begin{aligned}\dot{v}_C^- &= \left(u_1^- - v_C^-\right)(R_1 C)^{-1}\\ \dot{i}_L^- &= (u_2^- - R_3 i_L^-) L^{-1}.\end{aligned}$$

Because of the identities (6.4), we can see that for inputs $u_i^- = u_i^+$ the derivatives jump by the terms $i_L C^{-1}$ and $v_C L^{-1}$. If we adapt the input accordingly, this jump can be smoothed to zero, as we will demonstrate below.

Flatness of the Model

The model derived above is an FHA according to Definition 12. The automaton graph is complete, which induces strong connectedness. We have also shown that the continuous dynamics in each location are differentially flat. Regarding the reachability of the jump sets, we do not need any numerical computations: All dynamics are of order 1, i.e., they only include first derivatives, which are unconstrained. With the additional assumption of unconstrained inputs u_i, we see that the derivative can be changed to arbitrary values. The flow sets of the model are simply quadrants separated by the limit values v_0, i_0, as depicted in Figure 6.3. We can thus conclude that the flow sets are path-connected. Together, these properties yield reachability of all v_C, i_L from everywhere, which includes the jump sets.

Computing smooth trajectories and corresponding inputs

In the present example, the automaton graph is complete, and the switching only depends on the value of the flat output. Thus, instead of computing paths through a graph and corresponding discrete inputs, we can simply choose trajectories for the flat output and compute the corresponding inputs. We see that the concept of FHA reduces the trajectory generation for the hybrid system to a continuous problem, while preserving the hybrid nature of the system.

We construct a periodic trajectory for the two flat outputs

$$\begin{aligned} v_C^*(t) &= v^* \sin(t) \\ i_L^*(t) &= i^* \sin(t - \frac{1}{2}\pi) + i^*, \quad t \in [0, 2\pi] \end{aligned}$$

with $i^* = 2.5$ A, $v^* = 4$ V, which yield an ellipsis in the phase space starting and ending in $v_C^0, i_L^0, v_C^f, i_L^f = 0$, as depicted in Fig. 6.3. For the computations we use the following parameters: $R_1 = 5\ \Omega$, $R_2 = 2\ \Omega$, $R_3 = 3\ \Omega$, $C = 0.8$ F,

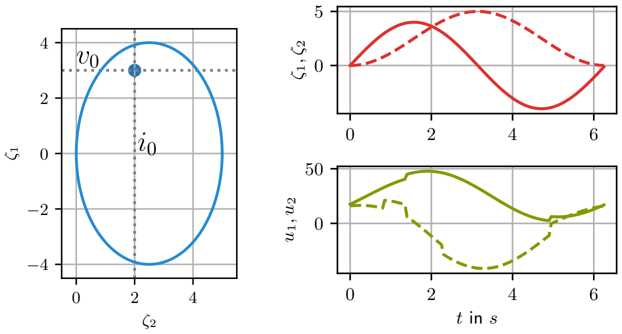


Abbildung 6.3: Left: Trajectory of the flat outputs in the state space. The dotted lines indicate the switching sets. Right: Trajectories of the flat outputs and the continuous inputs over time (Solid lines: v_C, u_1; dashed lines: i_L, u_2).

$L = 7$ L, $v_0 = 6$ V, $i_0 = 2.5$ A.[1] The inputs are calculated from the trajectory and its derivatives with Equations (6.3). From the values of $v_C^*(t), i_L^*(t)$, we deduce the discrete state and choose the corresponding equations. The trajectories and the corresponding inputs are depicted in Fig. 6.3. We see that we are able to control the hybrid system to follow continuous trajectories of the flat outputs in a straightforward way.

Robustness Trajectory Design

Around the point $v_C = v_0, i_L = i_0$ in the state space (blue dot in Fig. 6.3), the jump sets are very close to one another. In scenarios with limited inputs u, this property can be problematic. If, for example, a discrete state switching from I_1 to I_2 occurs (induced by $i_L = i_0$), and simultaneously v_C is just below v_0 with very high derivative $\dot{v}_C$, the adjacent switching set $v_C = v_0$ may be unavoidable. With unlimited inputs and the direct influence of the input on $\dot{v}_C, \dot{i}_L$ in the present example, however, this is not a problem. Remark that in real-world applications,

[1] These parameters are arbitrarily chosen for the simulation and do not represent any real electric circuit.

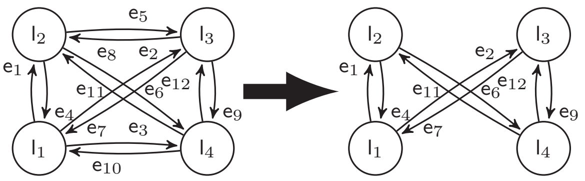


Abbildung 6.4: Automaton graph reduction for the DC network model.

even unconstrained inputs may be insufficient to solve this problem, as we showed in Chapter 5. One way to address such issues is to impose a certain distance to the point $v_C = v_0, i_L = i_0$ when planning trajectories.

Considering disturbances and modeling errors, there is another potential difficulty: Some transitions are triggered when both v_C and i_L reach a certain value simultaneously. Controlling both flat outputs to a precise value at exactly the same time may be difficult or even impossible in real-world conditions. One solution to the problem, which is also discussed in detail in Kleinert et al. (2020), is to eliminate the corresponding transitions from the model used for trajectory generation. This reduces the graph on the left in Figure 6.4 to the graph on the right. This way, the FHA model enforces the controller to compute more robust trajectories. In the present example, the cost of this approach is an increase in the complexity of trajectory computations for the model. Since the graph is no longer complete, all trajectory computations have to take into account the precise path through the automaton graph. We show how such a problem can be solved with optimization algorithms below in Section 7.

6.1.2 Tank Example

We present a one-tank system, which is depicted in Figure 6.5. The example is inspired by Fliess et al. (2005), and it is based on a two-tank example from Kleinert et al. (2020), but it has not been published in the form we present in the following. We model a tank as a storage for liquids with one inflow and multiple outflows, and we simulate a task to control the liquid level in the tank, denoted by l. We model the tank with a continuous, controlled input flow u, a switchable

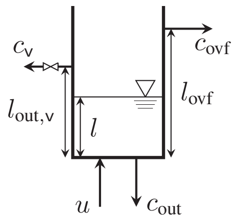


Abbildung 6.5: One-tank system setup.

outflow at level $l_{\text{out},\mathsf{v}} = 0.3$ m, an overflow at level $l_{\text{ovf}} = 0.6$ m, and a continuous outflow at the bottom. We model the dynamics as

$$\dot{l} = u - c_{\text{out}}\sqrt{l} - \mathsf{v}c_{\mathsf{v}}H(l - l_{\text{out},\mathsf{v}})\sqrt{|l - l_{\text{out},\mathsf{v}}|} - H(l - l_{\text{ovf}})c_{\text{ovf}}\sqrt{|l - l_{\text{ovf}}|} \quad (6.5)$$

with the discrete input v modeling open or closed outflow, where $\mathsf{v} = 1$ means that the outflow is open, and with constants $c_{\text{out}}, c_{\mathsf{v}}, c_{\text{ovf}} > 0$. The unit step function $H(\cdot)$ in Equation (6.5) is defined as

$$H(l) = \begin{cases} 0 & \text{if } l \leq 0, \\ 1 & \text{if } l > 0. \end{cases} \quad (6.6)$$

The model of the tank is very simplified, but it contains the non-linear relation between the liquid level and the outflows. We assume that the inflow u is normalized for the cross-sectional area, such that its unit is m/min. The parameters c_i of the outflows represent the shape and size of the outflows.

Additionally, we introduce constraints on the liquid level $l \in [0, l_{\text{max}}]$, which represent the maximum and minimum allowed levels. We also assume the input to be non-negative and limited, i.e., $u \in [0, u_{\text{max}}]$.

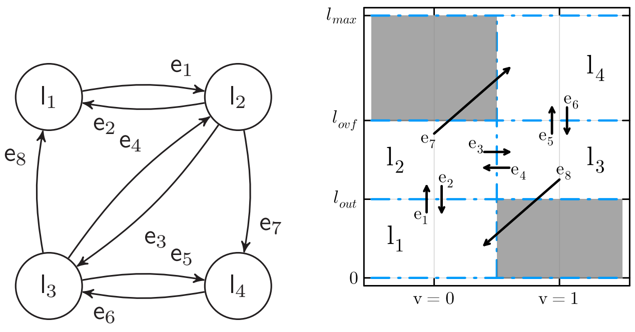


Abbildung 6.6: Automaton graph and plot of the locations depending on the flat output and the discrete input of the tank example.

FHA Model

To model the system as FHA, we first design the locations. Permuting the three zones of the liquid levels ($[0, l_{\mathrm{out,v}}]$, $[l_{\mathrm{out,v}}, l_{\mathrm{ovf}}]$, $[l_{\mathrm{ovf}}, l_{\mathrm{max}}]$) leads to six possible locations, as depicted in Figure 6.6. However, there are two locations which we do not include in our model: The location for $l < l_{\mathrm{out,v}}$ has the same dynamics for both values of district input v. We define the outflow to be closed for $l < l_{\mathrm{out,v}}$. For liquid level $l > l_{\mathrm{ovf}}$, the overflow is active. We define that in this operation mode, the switched outflow is open and can not be closed. These decisions lead to the locations and the graph shown in Figure 6.6. As we will see, locations l_1 and l_2 have the same continuous dynamics. Their difference lies in the guard sets: The switched outflow can only be opened in l_2.

The transitions are also shown in Figure 6.6. All transitions are caused by changes in the liquid level, except for transitions $\mathrm{e}_3, \mathrm{e}_4$, which are caused by the discrete input. Therefore, we have both autonomous transitions and controlled transitions. The guard sets are defined below.

The continuous dynamics in each location are the following:

$$\mathsf{l}_1 : \dot{l} = u - c_{\text{out}}\sqrt{l} \tag{6.7a}$$

$$\mathsf{l}_2 : \dot{l} = u - c_{\text{out}}\sqrt{l} \tag{6.7b}$$

$$\mathsf{l}_3 : \dot{l} = u - c_{\text{out}}\sqrt{l} - c_{\mathsf{v}}\sqrt{l - l_{\text{out,v}}} \tag{6.7c}$$

$$\mathsf{l}_4 : \dot{l} = u - c_{\text{out}}\sqrt{l} - c_{\mathsf{v}}\sqrt{l - l_{\text{out,v}}} - c_{\text{ovf}}\sqrt{l - l_{\text{ovf}}} \tag{6.7d}$$

with flow parameters c. These dynamics are all differentially flat with flat output $z = l$. The explicit input equations are simply

$$\mathsf{l}_1 : u = \dot{l} + c_{\text{out}}\sqrt{l} \tag{6.8a}$$

$$\mathsf{l}_2 : u = \dot{l} + c_{\text{out}}\sqrt{l} \tag{6.8b}$$

$$\mathsf{l}_3 : u = \dot{l} + c_{\text{out}}\sqrt{l} + c_{\mathsf{v}}\sqrt{l - l_{\text{out,v}}} \tag{6.8c}$$

$$\mathsf{l}_4 : u = \dot{l} + c_{\text{out}}\sqrt{l} + c_{\mathsf{v}}\sqrt{l - l_{\text{out,v}}} + c_{\text{ovf}}\sqrt{l - l_{\text{ovf}}}. \tag{6.8d}$$

The flow sets are :

$$\mathsf{Flow}_{\mathsf{l}_1} = \{0 \leq l \leq l_{\text{out,v}}\} \tag{6.9a}$$

$$\mathsf{Flow}_{\mathsf{l}_2} = \{l_{\text{out,v}} \leq l \leq l_{\text{ovf}}\} \tag{6.9b}$$

$$\mathsf{Flow}_{\mathsf{l}_3} = \{l_{\text{out,v}} \leq l \leq l_{\text{ovf}}\} \tag{6.9c}$$

$$\mathsf{Flow}_{\mathsf{l}_4} = \{l_{\text{ovf}} \leq l \leq l_{\max}\} \tag{6.9d}$$

Again, we write l instead of z or ζ, for the sake of readability.

The automaton graph in Figure 6.6 shows the possible transitions. We define the corresponding guard sets:

$$
\begin{aligned}
\mathcal{G}^{x}_{\mathsf{e}_1} &= \{l : l > l_{\text{out},\mathsf{v}}\} \\
\mathcal{G}^{x}_{\mathsf{e}_2} &= \{l : l < l_{\text{out},\mathsf{v}}\} \\
\mathcal{G}^{\mathsf{v}}_{\mathsf{e}_3} &= \{\mathsf{v} : \mathsf{v} = 1\} \\
\mathcal{G}^{\mathsf{v}}_{\mathsf{e}_4} &= \{\mathsf{v} : \mathsf{v} = 0\} \\
\mathcal{G}^{x}_{\mathsf{e}_5} &= \{l : l > l_{\text{ovf}}\} \\
\mathcal{G}^{x}_{\mathsf{e}_6} &= \{l : l < l_{\text{ovf}}\} \\
\mathcal{G}^{x}_{\mathsf{e}_7} &= \{l : l > l_{\text{ovf}}\} \\
\mathcal{G}^{x}_{\mathsf{e}_8} &= \{l : l < l_{\text{out},\mathsf{v}}\}
\end{aligned}
$$

The set of initial conditions are all steady states inside the flow sets.

Flatness of the Model

The HA model of the tank example is an FHA. First, we can see that the automaton graph is strongly connected, because the edges $\mathsf{e}_1, \mathsf{e}_7, \mathsf{e}_6, \mathsf{e}_8$ form a loop, implying that each location can be reached from every other location (see Figure 6.6). Second, we have shown above that the dynamics in each location are flat. Although there are theoretically non-flat sets if the arguments of the square roots are negative, these sets are not part of the flow set.

The reachability of the switching sets is the third property that we need to show. In the present example, the flow sets are obviously connected and even convex, but we have the complication that the input is non-negative and constrained. However, the derivative of the flat output, $\dot{l}$, also depends on l itself. The continuous outflow at the bottom of the tank guarantees that negative derivatives $\dot{l}$ are always possible for $u = 0$. In order to achieve positive $\dot{l}$, we must have sufficiently large u_{max}.

We proved in Section 5, that for reachability it is sufficient to have a 0-neighborhood included in the flow set. From the dynamic equations (6.7), we can directly see

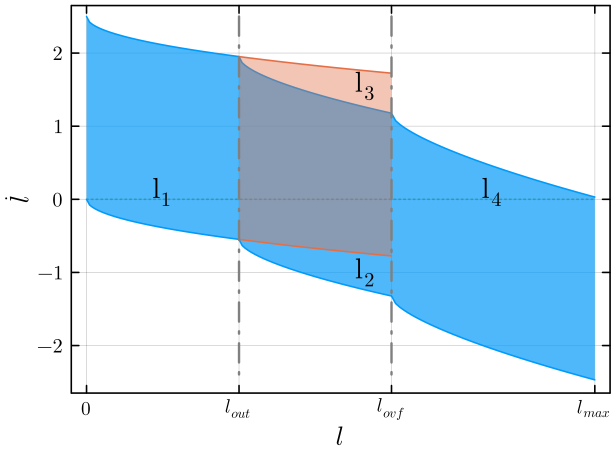


Abbildung 6.7: Depiction of possible derivatives $\dot{l}$ for all locations and levels l

when that is the case: If we substitute u_{max} for u in the equations, we can calculate the minimum allowable u_{max}. For l_1, l_2 this amounts to

$$u_{\text{max}} > c_{\text{out}}\sqrt{l},$$

and for l_3 and l_4 to

$$\begin{aligned} u_{\text{max}} &> c_{\text{out}}\sqrt{l} + c_{\text{v}}\sqrt{l - l_{\text{out,v}}} \\ u_{\text{max}} &> c_{\text{out}}\sqrt{l} + c_{\text{v}}\sqrt{l - l_{\text{out,v}}} - c_{\text{ovf}}\sqrt{l - l_{\text{ovf}}} \end{aligned}$$

respectively. Choosing the maximum allowable level l for each flow set and using parameters $c_i = 1$, we depict the flow sets in the l-$\dot{l}$ plain in Figure 6.7 for $u_{\text{max}} = c_{\text{out}}\sqrt{l_{\text{max}}} + c_{\text{v}}\sqrt{l_{\text{max}} - l_{\text{out,v}}} - c_{\text{ovf}}\sqrt{l_{\text{max}} - l_{\text{ovf}}}$.
We can thus conclude that the HA described above is an FHA, if $u_{\text{max}} \geq c_{\text{out}}\sqrt{l_{\text{max}}} + c_{\text{v}}\sqrt{l_{\text{max}} - l_{\text{out,v}}} - c_{\text{ovf}}\sqrt{l_{\text{max}} - l_{\text{ovf}}}$

Discussion

In the tank FHA model, we design the locations and the transitions according to two different reasons: The discrete input for one of the outflows and the dependence of their activation are physical properties of the system, which are reflected in the model. By putting these into the automaton part of the model, we are able to eliminate the non-smooth terms from the differential equations, which lead to much simpler continuous dynamics. We also make modeling choices that are only partly motivated by the physics. The exclusion of two of the permutations illustrates the design aspects of FHA modeling: We can model desired operation modes and exclude undesired modes, possibly also reflecting underlying controllers or logic. This way, we end up with an automation model to compute trajectories, as we will demonstrate below in Section 6.2.

The design choices and the physical phenomena yield the automaton graph with the locations and transitions. For systems with a larger number of locations and transitions, this can lead to more complex graphs. In Kleinert et al. (2020), we discuss how to eliminate transitions without losing the flatness property of the FHA. The publication also includes a three-tank example, representing a more complex system.

6.1.3 1 Degree-of-Freedom Robotic Thrower

Our third example is from the mechanical domain. We model a one-degree-of-freedom planar manipulator with the task to pick up objects at specific positions and to throw them into one of two buckets. The idea for this example is based on the pick-and-place examples published in Zahn et al. (2024). We model the dynamics of the manipulator as

$$L^2 m\ddot{\theta} = mgL\sin(\theta) + \tau \tag{6.10}$$

with input torque τ, manipulator length L, mass m, and angle θ, assuming that the mass of the manipulator is concentrated at its end. The setup is depicted in Figure

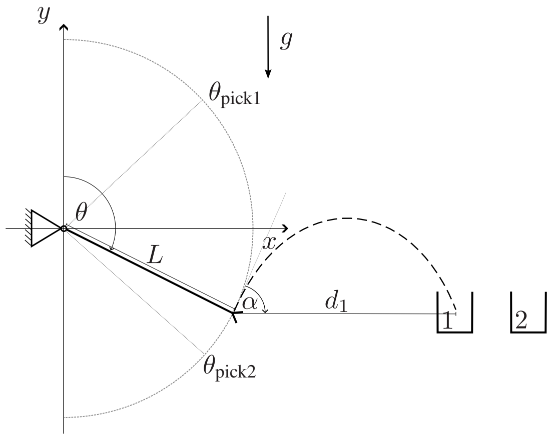


Abbildung 6.8: Schema of the throwing robot. Positive directions of θ and g are indicated by arrows.

6.8, which also shows how the angle is defined. These non-linear dynamics are differentially flat with flat output $z = \theta$. In the following, we state all equations in terms of the variable θ for the sake of clarity. As the flat output is equal to the state in this case, we only have to find an equation to express the input:

$$\tau = mL\left(L\ddot{\theta} - g\sin(\theta)\right). \tag{6.11}$$

In the field of robotics, this representation is well known as inverse dynamics. This inversion is possible because of the flatness property of fully actuated mechanical systems like robotic manipulators (Levine 2009, Section 6.2.2).

The task of the manipulator is to move to one of the pick locations θ_{pick}, pick an object, and throw it into one of the two buckets. The objects from the different pick locations have different masses m_{o1}, m_{o2}, respectively. We do not model the end effector doing the pick and the throw action. To derive an FHA model for this task, we first partition it into different phases, which are depicted in the automaton graph in Figure 6.9. The basic mode l_1 is free movement between the two pick locations, without carrying an object. To pick an object, the manipulator must move to a designated pick location and stop there. The angles of the pick locations with zero derivatives are therefore the guards of the autonomous transitions to l_2 and l_3. In the pick modes l_2 and l_3, discrete inputs trigger the picking and choose the target bucket, modeling controlled transitions. When $\mathsf{v}_1 = 1$ and $\mathsf{v}_2 = 0$, the target is bucket 1, and vice versa. If both inputs are 1 simultaneously, nothing happens. When the manipulator starts moving without having picked an object, it simply transitions back to free movement (l_1). In the throwing modes, the manipulator has to accelerate to reach a specific combination of angle θ and angular velocity $\dot{\theta}$. This triggers a release of the object. To hit the buckets, the object has to travel a specific distance d_1 or d_2. We assume free motion of the object after the release without air resistance. The motion of the object can be described by the equations of free motion

$$x(t) = v_0 t \cos(\alpha) \tag{6.12a}$$

$$y(t) = v_0 t \sin(\alpha) - \frac{1}{2} g t^2 \tag{6.12b}$$

with initial absolute velocity v_0 and angle α. The angles θ and α are geometrically related by $\pi - \theta = \alpha$, and $v_0 = \dot{\theta} L$. With these relations, we can write the relation of the distance, the angle, and the angular velocity:

$$L^2 \dot{\theta} \sin(2(\pi - \theta)) + g d_i = 0 \tag{6.13}$$

This equation assumes that the object lands at the same height as it was released at, and it takes the distance from the end effector to the landing spot. The actual distance from the end effector and the buckets again depends on the angle θ. To keep the model simple, we neglect this mismatch. However, we restrict the angles at which the objects are thrown, such that the mismatch remains small, $\theta_{\text{throw}} \in [\frac{3}{5}\pi, \frac{2}{3}\pi]$. The guard set to transition from throwing to free movement is therefore modeled by Equation (6.13) with the corresponding distances, and the additional condition $\dot{\theta} \leq 0$. We constrain the angle θ in l_1 and l_4-l_7 to be

$$\theta \in [\theta_{\text{pick1}}, \theta_{\text{pick2}}]. \tag{6.14}$$

Furthermore, we constrain the input torque by $\tau \in [\tau_{\text{min}}, \tau_{\text{max}}]$. This constraint can directly be mapped to $\theta, \ddot{\theta}$ by Equation (6.11). The values of the parameters

Tabelle 6.2: Parameters of the throwing robot.

L	1 m
m	1 kg
m_{o1}, m_{o2}	1.1 kg, 1.3 kg
g	$9.81\ \frac{\text{m}}{\text{s}^2}$
$\tau_{\text{min}}, \tau_{\text{max}}$	-50 Nm, 50 Nm
$\theta_{\text{min}}, \theta_{\text{max}}$	$\frac{1}{8}\pi$ rad, $\frac{7}{8}\pi$ rad
$\theta_{\text{pick1}}, \theta_{\text{pick2}}$	$\frac{1}{8}\pi$ rad, $\frac{7}{8}\pi$ rad
d_1, d_2	1.5 m, 2 m

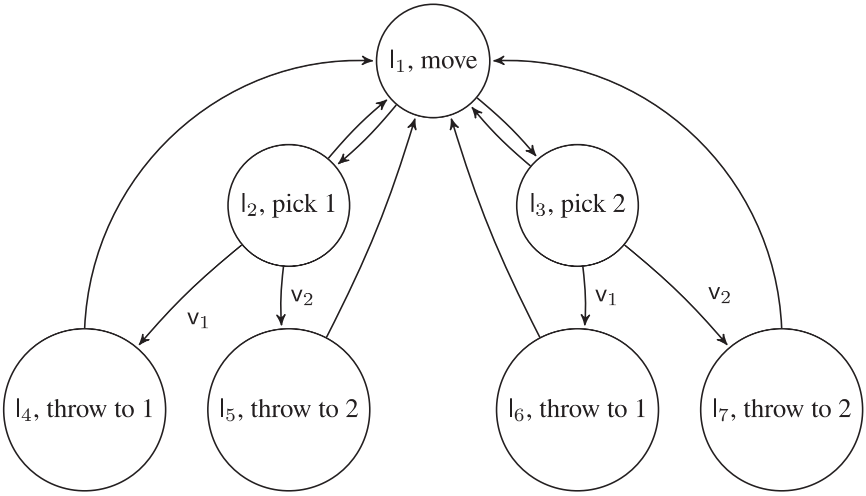


Abbildung 6.9: Automaton graph of the throwing robot.

are given in Table 6.2. The manipulator always starts operation from a steady state in location l_1.

Flatness of 1-DoF Thrower HA model

All parts of the hybrid automaton are defined in accordance with our definition in Section 3.2. We proceed to check whether conditions from Definition 12 defining FHA in Section 4.3 are met. The strong connectedness of the automaton graph can easily be deduced from the structure of the graph in Figure 6.9. Starting in l_1, we can form four loops $\{\mathsf{l}_1, \mathsf{l}_2, \mathsf{l}_4, \mathsf{l}_1\}$, $\{\mathsf{l}_1, \mathsf{l}_2, \mathsf{l}_5, \mathsf{l}_1\}$, $\{\mathsf{l}_1, \mathsf{l}_3, \mathsf{l}_6, \mathsf{l}_1\}$, $\{\mathsf{l}_1, \mathsf{l}_3, \mathsf{l}_7, \mathsf{l}_1\}$, which together visit all nodes.

The second condition is that the flow sets of each discrete state have to be flat. The dynamics have already been shown to be differentially flat above. Furthermore, for the locations l_1 and l_4-l_7, we have to show that the flow sets are path connected. In these locations, we have two constraints concerning the states: The angle is constrained by (6.14), and $\ddot{\theta}$ is implicitly constrained by the input constraint, as depicted in Figure 6.10. In the $\theta - \dot{\theta}$ plane, we additionally have the guards for throwing and picking. We can see from the definitions that the guards only cut out very small areas of the flow sets. This is also visible in the depictions of the flow set in Figure 6.11.
The third property is the reachability of the guards. We first note that the flow sets in l_1 and l_4-l_7 contain a 0-neighborhood, which can be seen in Figure 6.10. For the locations $\mathsf{l}_2, \mathsf{l}_3$, the transitions to the throwing modes are controlled and thus are reachable. The transitions to l_1 are trivially reachable by any movement.

We now look at the arrival sets and the transitions which are not defined by steady states and evaluate reachability. In location l_1, the arrival sets are given by the transitions from the throwing angle and velocity, described by Equation (6.13), and depicted in Figure 6.11. This arrival set models the state of the manipulator right after an object has been thrown. We evaluate numerically whether trajectories starting in the arrival set can stay in the flow set, as described in Chapter 5. This corresponds to decelerating after the throw while staying inside the constrained state space. To compute whether the arrival set is a set of feasible initial conditions, we use the numerical method based on Bézier curve introduced in Section 5.3.1 with the parameters in Table 6.2 and $N = 100$ control points. As depicted in Figure

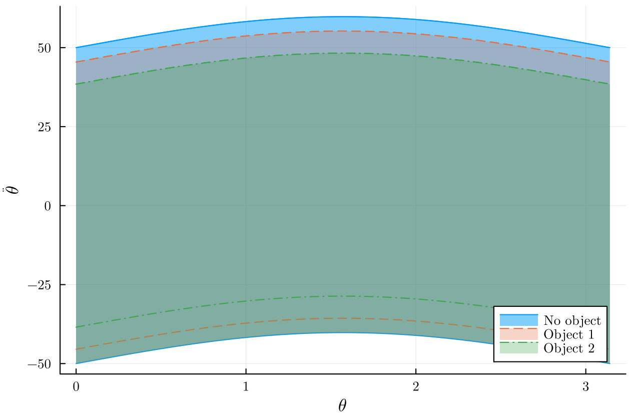


Abbildung 6.10: Input constraints of robotic thrower mapped to the $\theta - \ddot{\theta}$ plane for free movement, and movement with one of the two objects.

6.11 by several sample points, all initial states in the arrival sets are feasible. The guards are steady states at the pick locations and are therefore reachable.

In the locations l_4-l_7, the arrival sets are steady states at the pick positions. We can easily see that feasible trajectories inside the flow set exist from these states. The guards, however, are characterized by the throwing angle and velocity, described by the non-linear Equation (6.13). We evaluate the reachability of these guards by computing a trajectory to each, starting from an arbitrary steady state, using the Bézier approach and parameters as mentioned above. The results are again depicted in Figure 6.11, proving the reachability of the guards. We can thus conclude that the guards in l_4-l_7 are reachable.

We conclude that our model of the throwing robot is an FHA. Note that the reachability properties depend on the parameters. Therefore, the flatness of the model can only be proven for a given set of parameters. The model we developed for the throwing robot integrates its physics in different operation modes and the structure of its task. This allows for a straightforward computation of trajectories for a given sequence of tasks, as we will show in Section 7.2.1 below.

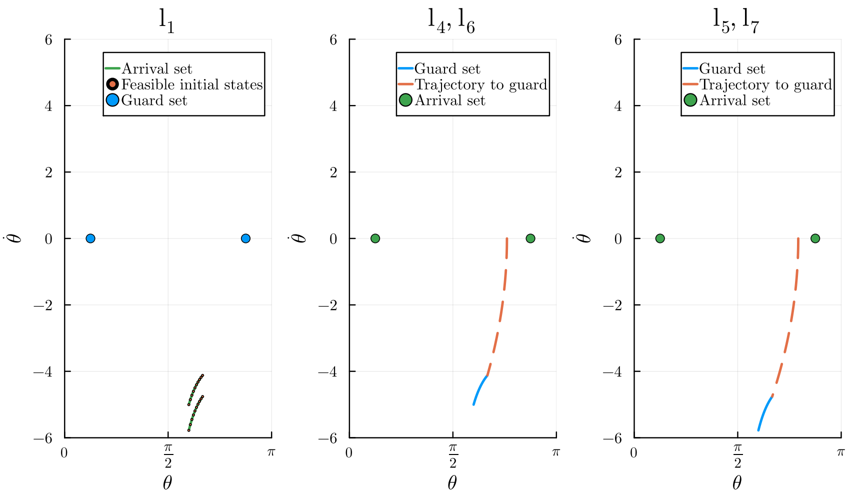


Abbildung 6.11: Reachability computation based on Bézier curves for robotic thrower for different locations.

6.2 FHA Inversion, Feedforward and Control

Probably the most useful property of differentially flat systems is their invertibility: Given a sufficiently smooth trajectory of their flat output z^*, we can use the map Ψ from Equation (2.8c) to compute the corresponding inputs u^*. The input signal u^* can then be applied to the system to produce the desired trajectory. Because of unmodeled effects and noise, however, the system will most certainly not produce the desired trajectory without an additional feedback controller. This combination is well studied for differentially flat systems, see e.g. Hagenmeyer and Delaleau (2003b).

For FHA, we have no mapping to compute the inputs (v^*, u^*) from a hybrid trajectory (l^*, z^*). The mixture of discrete-valued and continuous-valued variables makes it much harder to derive a closed-form inversion. That does not mean, however, that the inversion does not exist. Instead of deriving such a mapping, we will show its existence from the properties of FHA. We also demonstrate its application with the tank example described above, and with an electrical line switching example.

We assume that we have an FHA, and a hybrid trajectory for that FHA consisting of a sequence of locations $\{\mathsf{l}_j^*\}$, a sequence of flat output trajectories $\{z_j^*\}$, and a hybrid time $\mathcal{T}^*$, with $j = 0, \dots, j^f$. The trajectory is a solution to the FHA (see Definition 6), implying that all z_j^* are sufficiently smooth. We can compute the corresponding inputs with the following steps:
For each l_j^*

- Compute u_j^* from z_j^* via the corresponding Ψ_{l_j}.
- Compute v^* such that $\mathsf{v}^*(t^j, j) \in \mathsf{V}_{\mathsf{l}_j} \setminus \cup_{\mathsf{e}} \mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$, where $\mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$ are all controlled jump sets in l_j. The discrete input is thus chosen such that no switching is caused.
- From $\mathsf{l}_j, \mathsf{l}_{j+1}$ take the corresponding edge e_{j+1}. If the guard set is from a controlled transition, i.e., $\mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$, set $\mathsf{v}^*(t^{j+1}, j) \in \mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$, else set $\mathsf{v}^*(t^{j+1}, j) = \mathsf{v}^*(t^j, j)$.

We briefly discuss the complexity of these computations. For the first step, we assume that the trajectories and the maps are available as explicit function objects. The computation then amounts to constructing a nested function. The second step is potentially the most expensive, as we have to compute the set of all controlled switching sets and find an element that is outside of this set. The sets $\mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$ are mutually exclusive sets of binary vectors. This problem can be formulated as a constraint satisfaction problem, which is combinatorial and therefore of theoretically high complexity (Brailsford et al. 1999). For a low number of discrete inputs, the problem can be solved efficiently, for example, by backtracking. However, it can be expected to become intractable for systems with a very high number of discrete inputs. The last step in the inversion simply involves look-ups of values and is therefore of low complexity. Because of the simple connectedness of the graph, each pair $\mathsf{l}_j, \mathsf{l}_{j+1}$ can be mapped to a single edge. Obtaining the corresponding discrete input can be underdetermined if a guard set $\mathcal{G}_{\mathsf{e}}^{\mathsf{v}}$ contains multiple combinations. Thus, the inversion for FHA always exists by construction, but it cannot generally be a bijection. The solution to the inversion problem can be non-unique. In the following, we demonstrate inversion, feedforward, and control for the tank example from Section 6.1.2.

6.2.1 Application to Tank Example

Taking the example of the tank from Section 6.1.2, we first compute a trajectory. To this end, we choose three operation points: A starting point at the initial hybrid state $(\mathsf{l}_1, l = 0.1\ \mathrm{m})$. We then fill the tank to reach $(\mathsf{l}_4, l = 0.8\ \mathrm{m})$ and keep the level for some time. In the end, we empty the tank to go back to $(\mathsf{l}_1, l = 0.2\ \mathrm{m})$. For the computations we use the parameters $c_{\mathrm{out}} = c_{\mathsf{v}} = c_{\mathrm{ovf}} = 1$.

To compute a trajectory, we first decide on a sequence of discrete states that satisfy the three operation points. We choose the sequence $\{\mathsf{l}_1, \mathsf{l}_2, \mathsf{l}_4, \mathsf{l}_3, \mathsf{l}_1\}$. For the continuous trajectories, we use simple cubic polynomials of the form

$$l^*(t) = at^3 + bt^2 + ct + d. \tag{6.15}$$

We choose the switching times (t^j, j) as the sequence $\{(0,0), (1,1), (2,2), (5,3), (6,4), (7,5)\}$ where the unit of time is minutes. The trajectory in the time interval $[2,5]$ is computed as a collocation of three polynomials to achieve the operation at a constant level. The resulting trajectory for the liquid level is depicted in the left plot in Figure 6.12. Combining the path, the hybrid time domain, the initial and end states, and the flat output, we compute the inputs to realize the trajectory. We compute the discrete inputs using the switching sets: To transition from l_2 to l_4, v has to change from 0 to 1 at $t^2 = 2$ min. At the following transition from l_4 to l_3, v changes back to 0. For the continuous inputs, we use the dynamic equations (6.7a)-(6.7d) and compute the values directly from the trajectories l^*, as discussed above. The result is depicted in Figure 6.12.

In a real-world application of the computed inputs, we would face at least two problems: Model mismatch and disturbances. To simulate these effects, we use the dynamic model (6.7) and add a disturbance term such that $u = u^* + d$, where

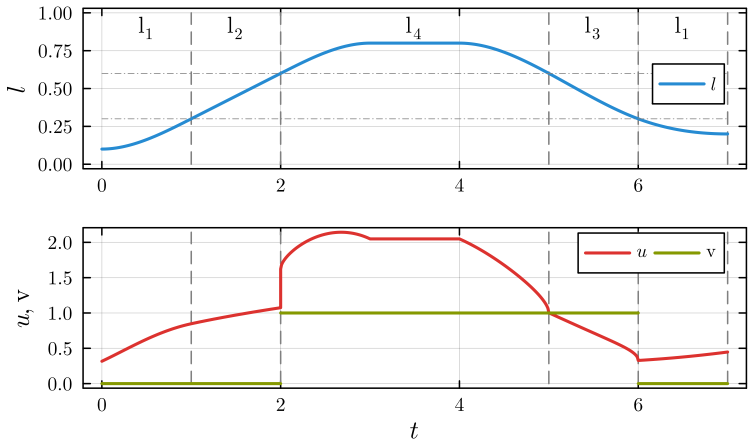


Abbildung 6.12: Liquid level and inputs for the one-tank example. The green line shows the discrete input v, the red line shows the continuous input u. The dashed gray lines mark the times of the state switching.

d is a random variable with normal distribution, e.g. $d \sim \mathcal{N}(0, 0.2)$. Additionally, we vary the parameters c_i of the simulation model to be $c_{\text{out}} = 1.2$, $c_{\text{v}} = 0.98$, $c_{\text{ovf}} = 1.08$. When we apply the feedforward input to the simulation model, we obtain the system response depicted in Figure 6.13. We can clearly see that the desired trajectory is not achieved. Around $t = 2$ min, we observe that the switched input and the jump in the continuous input do not occur at the right system state, because the actual trajectory is much lower than the computed trajectory.

To make the system follow the desired trajectory, we further introduce a PI-controller with the terms

$$u_c(t) = k_P(l^*(t) - l(t)) + l_I \int_0^t (l^*(\tau) - l(\tau))\,\mathrm{d}\tau, \tag{6.16}$$

with parameters $k_P = 10, k_I = 10$. This gives rise to the classic combination of feedforward control and PI feedback control (Hagenmeyer and Delaleau 2003b). Further considerations of stability and robustness are beyond the scope of this work. With the PI controller (6.16), the new input becomes $u = u^* + d + u_c$.

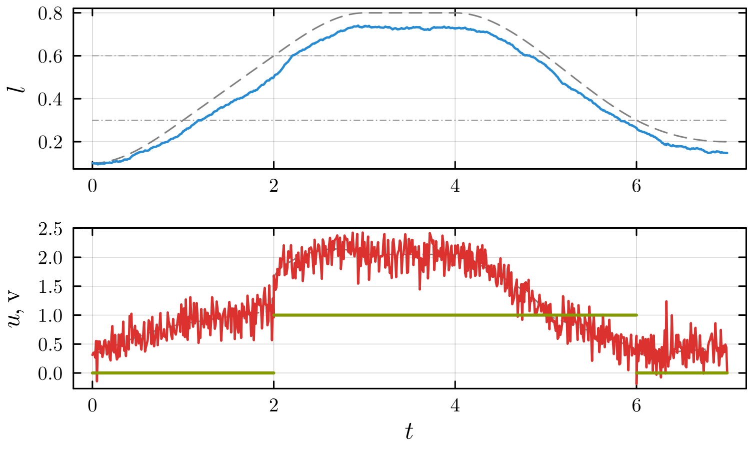


Abbildung 6.13: Simulation of feedforward of the computed inputs with model mismatch and disturbance.

We run our simulation again, with the results in Figure 6.14. The results show a smooth trajectory, which closely tracks the desired trajectory. We can also see the adapted input, which does not track the precomputed input from the inversion but compensates for the disturbance and the model mismatch. In the example, we only applied a controller to the continuous input. If the timing of the discrete input is of critical importance, we would also need to include v in the controller. One way to do this is optimization-based predictive controllers, as described in Section 7 below. Additionally, an observer may be necessary to measure the actual system state. In the present work, we do not include hybrid system observers. The interested reader may refer to Bernard and Sanfelice (2022).

6.2.2 Application to Transmission Line Switching

In the following, we present a 3-bus Alternating Current (AC) microgrid example, to which we apply trajectory planning and feedforward for controlled transmission line switching. This example is published in Zahn and Hagenmeyer (2022). Electrical grids are complex dynamical systems, which naturally include switching

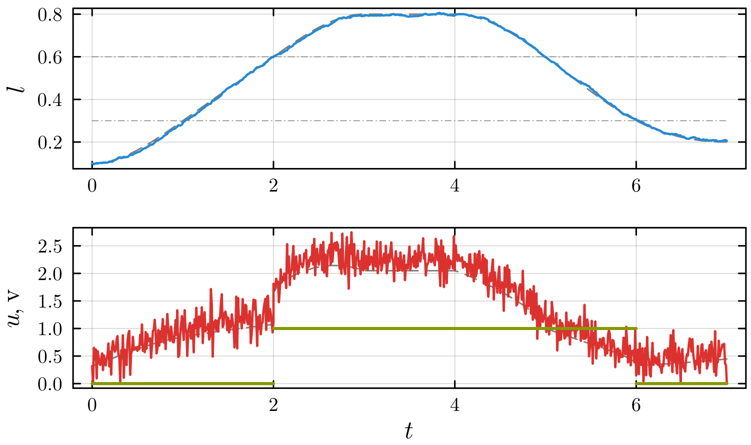


Abbildung 6.14: Simulation of feedforward with PI-controller for tank example. The simulation includes model mismatch and disturbance.

elements. They are designed to transport electrical power from the generators to the consumers. The flow of power in a grid is mainly determined by the power injections and consumptions, and by the grid topology. One important means to control the power flows besides the generation is to switch lines. Lines in electrical grids are generally opened and closed for maintenance or to change the topology of the grid. Changing the topology of a grid can lead to greater transmission efficiency and stability (Taheri and Molzahn 2025). However, opening or closing a line in regular operation can lead to transients and wear the switches, as demonstrated by Huang et al. (2014).

Electrical grids are commonly modeled as graphs. The vertices are the connection points of generators and loads, which are called buses. The buses are connected by lines, represented by edges, and form the grid topology. In normal operation, the loads change over time, and the generators adapt their output to match the loads. Depending on the locations and magnitudes of the loads, different grid topologies can be advantageous to minimize transmission losses and generation cost. The research literature on transmission line switching is largely focused on computing the optimal steady-state topology (Taheri and Molzahn 2025), evaluating or improving stability before and after switching (Bruno et al. 2012). The topic is becoming ever more relevant with electrical grids being operated closer to their limits, and with an increasing number of controllable power electronic elements.

In Zahn and Hagenmeyer (2022), we present the idea to consider line switching in a hybrid system framework to minimize the transients and the risk of instability. To this end, we propose to control the system to a state in which the line that we want to switch does not carry any load.

Swing Equation Dynamic Grid Model

A common approach to model the dynamics of AC power networks with generators and loads is to approximate the system behavior by second-order swing equations, as described in Dörfler and Bullo (2012), Bergen and Hill (1981), Kundur (2012). Assuming lossless transmission, constant voltage amplitude and second-order generator dynamics, we can model the dynamics of each bus as

$$M_i \ddot{\delta}_i + D_i \dot{\delta}_i = P_i - \sum_{j=1, j \neq i}^{n} b_{ij} \sin(\delta_i - \delta_j) \quad i \in \{1, \dots, n\} \tag{6.17}$$

for n buses with generator inertia M_i and damping D_i, line susceptances b_{ij}, bus net power injection P_i and generator angles δ_i. When two buses i and j are connected by a line, the susceptance b_{ij} is a positive constant; otherwise, it is zero. The relative angles between the buses determine the power transmitted through the corresponding line between them, i.e., $P_{ij} = b_{ij} \sin(\delta_i - \delta_j)$. The angles are the state variables of the dynamic system, and the power injections P_i are the inputs.

Assuming a controllable power source at each bus, the power system dynamics described by 6.17 are differentially flat. Taking angles δ_i as flat outputs and mechanical power injection P_i as inputs, system (6.17) is differentially flat for

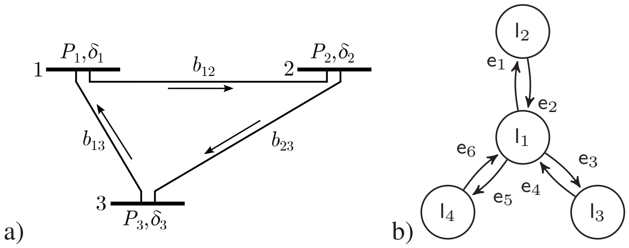


Abbildung 6.15: a) 3 bus AC microgrid with power injections P_i, angles δ_i and line susceptances b_{ij}. The arrows denote the direction of positive power transmitted by the lines. b) Automaton graph of the FHA model of the power network.

$\delta_i - \delta_j \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. We can easily rearrange (6.17) to solve for the inputs, the relation of the flat outputs and the states is trivial.

In the following, we assume our grid to be a microgrid in islanded mode, i.e., without connection to a larger grid. Furthermore, we model all buses as controllable power sources. In the microgrid literature, controllable power sources are routinely used, which is justified by assuming energy storage as a fast-reacting power source, see, e.g., Schiffer et al. (2016). In this setup, all buses consist of a combination of generation, load, and inverter-connected storage. Thus, the power P_i at bus i is the net power injection that is provided to the grid at that bus. Accordingly, all buses of the example network are modeled as generator buses with controllable net power injection.

FHA Model of a 3 Bus AC Network

To demonstrate our ideas, we derive an FHA model for a 3-bus network depicted in Figure 6.15a. With Equation (6.17) we obtain the following set of differential equations:

$$M_1\ddot{\delta}_1 + D_1\dot{\delta}_1 = -\mathsf{v}_{12}b_{12}\sin(\delta_1-\delta_2) - \mathsf{v}_{13}b_{13}\sin(\delta_1-\delta_3) + P_1 \tag{6.18a}$$
$$M_2\ddot{\delta}_2 + D_2\dot{\delta}_2 = -\mathsf{v}_{12}b_{12}\sin(\delta_2-\delta_1) - \mathsf{v}_{23}b_{23}\sin(\delta_2-\delta_3) + P_2 \tag{6.18b}$$
$$M_3\ddot{\delta}_3 + D_3\dot{\delta}_3 = -\mathsf{v}_{13}b_{13}\sin(\delta_3-\delta_1) - \mathsf{v}_{23}b_{23}\sin(\delta_3-\delta_2) + P_3. \tag{6.18c}$$

The hybrid nature of this system is modeled by the discrete inputs v_{ij}, which are used for opening or closing lines. As established above, these dynamics are differentially flat. The locations of the HA are defined by the values of the discrete inputs v_{ij}. In the present work, we only consider opening one single switch at a time for the sake of simplicity. Thus, modeling the system as FHA, we identify four locations as shown in Figure 6.15. Location l_1 models the system with all switches closed, i.e. $\mathsf{v}_{12}, \mathsf{v}_{13}, \mathsf{v}_{23} = 1$. The remaining states each model the system with one switch of a respective line open, i.e., for l_2, $\mathsf{v}_{12} = 0$; for l_3, $\mathsf{v}_{13} = 0$; for l_4, $\mathsf{v}_{23} = 0$.

Due to the fact that only one switch can be opened at a time, locations $\mathsf{l}_2, \mathsf{l}_3$ and l_4 are all only connected to l_1 and not to each other, yielding an automaton graph with 6 edges. The guards are defined in terms of the values of the discrete inputs, yielding controllable transitions only. The automaton graph is strongly connected. Reachability of the guards is given because all transitions are controllable. The reset functions of the transitions define the change in the second derivatives induced by opening or closing a line:

$$M_i \ddot{\delta}_i^+ = M_i \ddot{\delta}_i^- + D_i \dot{\delta}_i^- - \sum_{j=1, j\neq i}^{n} b_{ij} \sin\left(\delta_i^- - \delta_j^-\right)(\mathsf{v}_{ij}^- - \mathsf{v}_{ij}^+) \tag{6.19}$$

The inputs P_i are unchanged by the transition and cancel out. The angles δ_i and their first derivative are not directly affected by the transition.

Feedforward Switching Control of a 3 Bus AC Network

We take the 3-bus model derived above for smooth feedforward line switching. The grid is assumed to be in a steady state before switching, and we want to return it to a steady state after switching. The process of opening a line of the network can be described in terms of the FHA model as follows: Starting in l_1 with all lines closed, the system dynamics are described by Equations (6.18). The angles can be controlled to any value in $[-\pi, \pi]$ by the inputs P_1, P_2, P_3. At all moments, a line can be opened by the discrete inputs v_{ij}. This causes the discrete state to change.

The new dynamics of the grid are described by simply setting the corresponding parameter b_{ij} to 0 in Equations (6.18).

Opening a line in an arbitrary system state causes a discontinuity in the second derivatives of the angles according to Equation (6.19). The discontinuity are describe by the term $b_{ij} \sin{(\delta_i^- - \delta_j^-)}(\mathsf{v}_{ij}^- - \mathsf{v}_{ij}^+)$ of the switched line. From this relation, we can see that no discontinuity occurs if the angle difference of the adjacent buses is zero. A zero angle difference over the line implies that no power is transmitted. For our FHA model, we thus want to compute trajectories and feedforward inputs to control the line switching to happen at a point where the angle difference over the corresponding line is zero.

The controllability of the FHA model allows for the computation and implementation of the corresponding trajectories and inputs in a straightforward manner. In the following, we describe the computation of a trajectory that connects an initial and a final state and includes the opening line b_{12}. With a non-flat automaton model, one would need to search for trajectories using integration algorithms, without knowing whether any feasible solution exists. The FHA approach allows for the direct computation of trajectories and the corresponding inputs, because controllability is guaranteed by model design. Moreover, including constraints like line current limits in the model would directly be visible in the automaton graph and the domain of the continuous variables, effectively eliminating certain system transitions. Thus, all actions that are included in the FHA model are feasible.

We consider the case that two steady states with fixed power and angles of the system are known, an initial state at time $t^0 = 0$ sec and a final state at $t^f = 20$ sec. We choose $P_1 = 0.3$ p.u., $P_2 = 0.2$ p.u., $P_3 = -0.5$ p.u. as initial and final powers and compute the remaining states, assuming that the system is in a steady state. The units $p.u.$ refer to the per-unit system commonly used in power system studies. In the present example, bus 3 with negative power is a net receiver of power from the grid in steady state. With line parameters $b_{12} = 0.8$, $b_{13} = 1$, $b_{23} = 1.3$, the corresponding angles are $\delta_1(t^0) = 0.045$, $\delta_2(t^0) = 0.287$, $\delta_3(t^0) = 0.263$ and $\delta_1(t^f) = -0.147$, $\delta_2(t^f) = 0.453$, $\delta_3(t^f) = 0.376$. For the sake of simplicity, we assume $M_i = 1$ and $D_i = 5$, inspired by values from Dörfler and Bullo (2012).

We want to compute a trajectory for the system that allows us to open line b_{12} at $t^s = 10$. In order to compute smooth trajectories for the angles, we use two fourth-order polynomials of the form

$$p(t) = at^4 + bt^3 + ct^2 + dt + e \tag{6.20}$$

for each state δ_i. We use one polynomial p_1 to compute the trajectory from the initial state to the switching set and another polynomial p_2 to connect the final state with the switching set for each δ_i. The initial and final conditions mentioned above are completed by imposing zero derivatives at t^0 and t^f to obtain steady states. We further join the polynomials at t^s by demanding equal derivatives

$$\dot{p}_1(t^s) = \dot{p}_2(t^s) \quad \ddot{p}_1(t^s) = \ddot{p}_2(t^s), \tag{6.21}$$

and setting

$$\delta_1(t^s) = \delta_2(t^s) = 0 \quad \delta_3(t^s) = \frac{\delta_3(t^0) + \delta_3(t^f)}{2}. \tag{6.22}$$

Computing the trajectories as polynomials amounts to solving a linear system of equations. In the present case, the dimension of the block diagonal problem matrix is 30 $\times$ 30, i.e., 10 equations per bus for this example. These conditions, together with the initial and the final state, yield the trajectories for the angles in Figure 6.16. In the figure, we also see the corresponding inputs P_1, P_2, P_3 that we can compute directly from the angles. Furthermore, in Figure 6.16 we see the power transmitted by each line. The sign of the power is according to the directions given in Figure 6.15. The switch of the line between bus 1 and bus 2 is opened at a point where it does not carry any load, leading to smooth trajectories for the whole system.

In Figure 6.17, we see the system response for opening the same line at the initial steady state. There is a big difference in the trajectories between controlled line switching and simply opening a line in the steady state. Although the system comes back to a steady state eventually, we can see quite large angle deviations, which lie outside the permitted ranges given above. However, it is important to

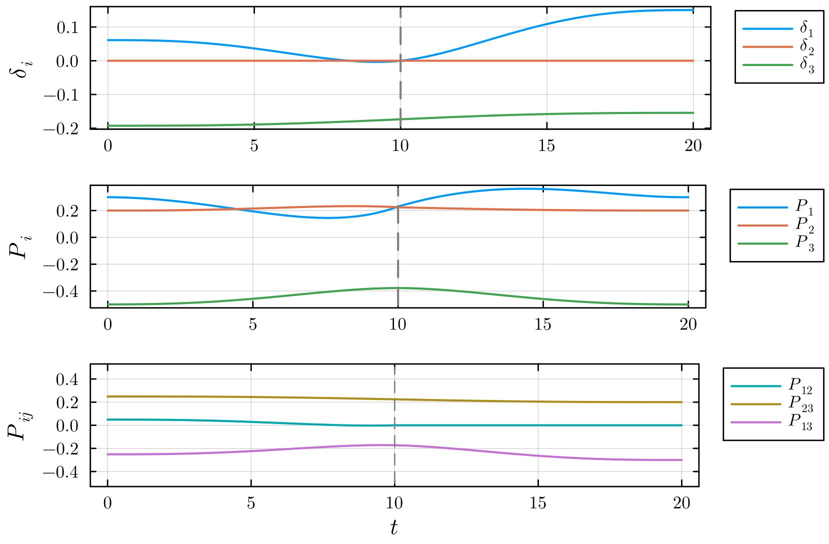


Abbildung 6.16: Trajectories of the angles, input signals and line loads for the line switching feedforward example.

emphasize that this system response is due to the small size of the grid and is not representative of line switching in real grids. For a study on the system response for larger systems, we refer to Dörfler and Bullo (2012).

6.3 Summary and Discussion

In this chapter, we apply our approach to four different examples. The hybrid nature of their dynamics is rooted either in physics or in the specification of the operation modes or tasks, and the transition between these. In many cases, both are present. For each example, we show step-by-step how to model them as HA, how to prove that the resulting HA are flat, and we discuss the modeling choices that lead to the flatness property. The electrical DC circuit example in Section 6.1.1 has physical switches. We derive the corresponding FHA model and demonstrate how to compute smooth trajectories, as well as the effects of design

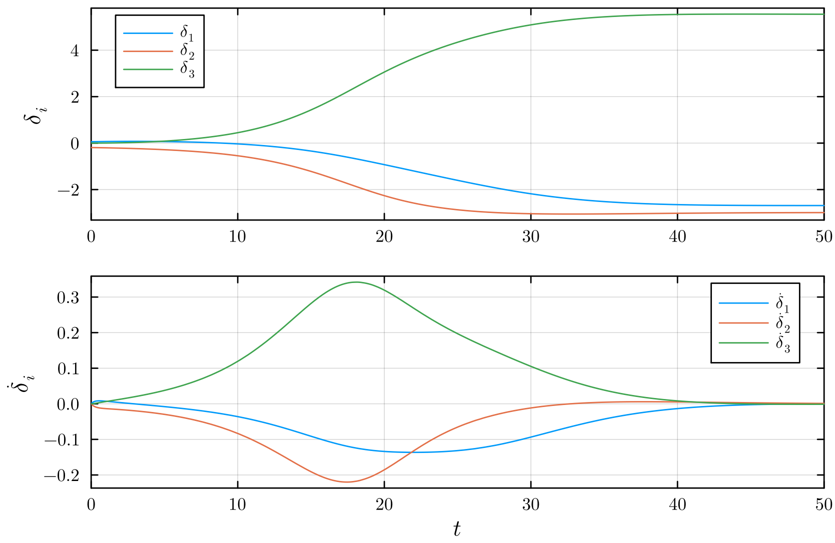


Abbildung 6.17: Transient response to opening line 12 at $t = 0$.

choices on robustness and control. The tank example combines physical changes and the specification of operation modes. Both are encoded in the FHA model. Furthermore, we discuss the effect of input constraints on the flatness property. In the robot example, the hybrid nature mainly originates from the task specification. We demonstrate how to use the reachability computations from Chapter 5 to establish the flatness property of our model, even for a non-linear guard set that includes the derivatives of the flat output.

In the second part of the chapter, we discuss feedforward for FHA. We show the basic algorithm for inversion, and we demonstrate how to apply it for feedforward to two examples. We also identify possible challenges regarding the computation of the different steps. For the tank example, we simulate feedforward control. In order to address potential issues from disturbances and parameter mismatch, we also demonstrate the use of a classic PI controller in that example. In the second example for AC grid switching, we apply our approach to a use case in transmission grid control. Here, the FHA framework allows for switching transmission lines without causing transients in the system.

As we show in our examples, modeling systems as FHA can involve not only physics, but also design decisions. Therefore, the model can include information about processes and tasks, which can be advantageous for automation. The constraints, automaton graph, and transition rules can be designed to limit possible trajectories to specific desired characteristics. However, modeling systems as FHA may require considerable time and effort by the engineer. The specific choices also influence whether the resulting model is flat or not. We can thus view flatness for HA as a design principle, which emphasizes intentional model design for reachability and inversion. Whether the modeling process can be automated to some degree remains an open question for future work.

The great benefits of modeling a system as FHA are their reachability property, the relative freedom of the trajectories, and the possibility of inversion. As we demonstrate, inversion can be very simple for a given hybrid trajectory. For systems with a very high number of discrete inputs, the inversion may become more computationally challenging, but in practice, this may be handled by look-up tables and rules. However, the inversion relies on a given feasible trajectory. In the present chapter, we rely on simple polynomial approaches, which are very limited. In the following chapter, we investigate how to compute trajectories with numerical optimization methods and how complex the corresponding problems are.

7 Optimization-Based Trajectory Generation and Control for FHA

Trajectories play a vital part in control and automation applications: Controlling processes, guiding vehicles and mobile robots on the ground, in the air, and in the water, and automating the operation of technical systems generally involves the computation of feasible and safe trajectories. In the context of flat systems, these trajectories are a necessary ingredient for the inversion described in the previous section. Additionally, for model predictive control schemes, optimized trajectories form the core of the computations. Even if we are not necessarily interested in the globally optimal trajectory, optimization offers a way to compute feasible trajectories that are efficient with respect to some cost criteria. However, their generation can be computationally challenging for non-linear systems, and for hybrid systems in particular. The main challenge lies in the presence of non-linear constraints from the system dynamics, and in the combination of continuous and discrete variables. As we described above, trajectory computation is much simpler for differentially flat systems than for general non-linear systems, because of their equivalence to trivial systems in terms of their flat output (Greco et al. 2022). This is also the case for trajectory optimization, see e.g. Hagenmeyer and Delaleau (2008), Beaver and Malikopoulos (2024). However, for flat hybrid systems, we should not expect the same properties. Additionally, the FHA models can have very different structures, as we have shown for the examples in Section 6. This also leads to different optimization targets and problem structures.

Trajectory generation and optimization for differentially flat systems have been studied extensively. Most works follow one of two approaches to represent trajectories: They are either approximated by polynomial functions, e.g., in Milam et al.

(2000), Kolar et al. (2017), Suryawan et al. (2012), Oldenburg and Marquardt (2002), or they are discretized in the dynamics, see e.g. Greeff and Schoellig (2018), Faulwasser et al. (2014), Petit et al. (2001), Agrawal et al. (2022). A noteworthy exception is the work by Beaver and Malikopoulos (2024), in which the authors solve the problem by computing Lagrange multipliers of movement primitives. In Greco et al. (2022), the authors develop an approach to approximate the set of all feasible constrained trajectories by approximating them as Bézier polynomials. Furthermore, most path planning algorithms from robotics can be applied to differentially flat systems as well, when they rely on inverse dynamics formulations. These formulations are based on the flatness property of fully actuated mechanical systems. A particularly interesting approach for our work was developed in Marcucci et al. (2023, 2024b), which uses sets of convex boxes and Bézier curves to formulate a mixed-integer convex program that can be solved efficiently. The mixed-integer formulation can naturally model hybrid systems, as the authors show in Marcucci et al. (2024b).

The literature on hybrid system optimization is rather diverse, reflecting the diversity of hybrid system model types. The different treatments of the continuous dynamics, the discontinuities, and the transitions give rise to a number of approaches. A rather complete overview of approaches can be found in Barton et al. (2006). Many publications pair the trajectory optimization with a model predictive controller. The optimization approaches are nevertheless quite similar. There is also a number of publications on optimal control and optimized controller design for hybrid systems, which face very different challenges, and we do not discuss them in the present work. An important body of literature investigates methods for non-smooth differential equations, which can also be modeled as hybrid systems. These works mainly develop specialized tools to deal with the discontinuities and switching. They mostly use numerical integration schemes to solve the differential equations, combined with some method to encode and compute the discontinuities. The work by Nurkanovic (2023) gives a great account of theory, methods, and applications. For hybrid systems in particular, the approaches in the literature mostly rely on some form of integer variables to encode discrete inputs and transitions, resulting in mixed integer optimization problems. Concerning the continuous dynamics, many works use a full discretization of the hybrid dynamics, which for linear dynamics is formalized in the mixed logical dynamical (MLD) framework (Bemporad and Morari (1999), Sanfelice et al. (2007)).

There are a few publications describing the combination of hybrid systems, flatness, and optimization. However, these only apply flatness to the continuous dynamics, not considering the whole system's flatness. In Sreenath et al. (2013), trajectories for quadrotors with cable-suspended payload are computed and followed. The authors model the system as a hybrid system with two modes, depending on whether the cable is under tension or not. Although they explicitly describe the combination of flatness and hybrid systems, their control approach does not take the hybrid nature into account. The work of Marcucci et al. (2024b) integrates graphs and trajectory computation based on Bézier curves. The approach is tailored to fully actuated robotic systems, which are differentially flat. The problem they derive is called the Shortest Path Problem for Graphs of Convex Sets (SPP-GCS). The authors also demonstrate how to apply their method to piecewise-affine hybrid systems, which can also be modeled as FHA.

It is important to note that optimization for FHA is not categorically different from that of other hybrid systems. Any general approach to trajectory computation and optimization for HA can also be applied to FHA. Therefore, the problems we solve in this chapter can be solved without using our FHA model. However, FHA offer unique advantages because of their special properties, which we describe and leverage in this chapter. Furthermore, the guaranteed properties of FHA allow for a systematic evaluation of approaches, which is not possible for very general model classes such as HA or HS.

In the present chapter, we demonstrate the possible advantages of FHA over general HA for trajectory generation for two examples, which represent two different types of systems. We begin by stating the general optimization problem combining the optimization over the sequence of locations and over the continuous trajectories. Based on the general problem, we first investigate the setting in which the sequence is fixed, and we compute trajectories for the robotic thrower in this setting. The second type of problems we look at includes both the optimized computation of the sequence and of the continuous trajectories. These problems are much harder to solve. We present an example of line switching extending the example from Section 6.2.2. All material in this chapter is unpublished.

7.1 Optimization Problem

In most cases, trajectory generation is implemented with some form of numerical optimization, as we described above. Therefore, we investigate trajectory generation for FHA as an optimization problem. There are many possible combinations of constraints and optimization objectives, representing different application types. The different approaches we present below are tailored to different applications, depending on the types of transitions and the optimization objectives.

Before we dive into specific problem formulations, we first define the general problem on a conceptual level. For a given FHA and initial state (l^0, z^0) and a final state (l^f, z^f), we want to compute a sequence of locations $(\mathsf{l}^j)_{j\in\{0,\ldots,f\}}$ of unknown length f, and a corresponding sequence of flat output trajectories (z_{l}^*). Transitions can be computed from the sequence by $\mathsf{e} = (\mathsf{l}, \mathsf{l}')$. We can compute a trajectory for an FHA by solving an optimization problem of the abstract form:

$$\begin{aligned}
\min_{(\mathsf{l}^j)_j,(z_{\mathsf{l}}(\cdot))} \quad & \sum_{(\mathsf{l}^j)_j} c(\mathsf{l}^j, z_{\mathsf{l}}) && \text{(7.1a)}\\
\text{s.t.} \quad & (\mathsf{l}^j)_j \in \mathsf{L}^* && \text{(7.1b)}\\
& z_{\mathsf{l}}(\cdot) \in \mathsf{Flow}_{\mathsf{l}} && \text{(7.1c)}\\
& \text{Transition constraints} && \text{(7.1d)}\\
& \text{Initial and final state constraints} && \text{(7.1e)}
\end{aligned}$$

Here, L^* denotes the set of all admissible sequences of the automaton. In this very generalized formulation, the sequence of locations $(\mathsf{l}^j)_j$ and the corresponding sequence of flat output trajectories $(z_{\mathsf{l}}^j)_j$ minimize some cost function $c(\cdot)$. The constraints (7.1b)-(7.1d) restrict the feasible set to be actual trajectories of the given FHA. The constraints (7.2e) specify the initial conditions, and also allow for imposing a final state. The final state may not always be known in advance, and it can also be included in implicit form in the cost function instead, as it is common in many MPC formulations.

For the Problem (7.1), we can already deduce the potential complexity of the numerical solution algorithms: We have a mixture of discrete variables l, and functions z_{l} with potentially non-linear constraints. Thus, in its general form, the problem is intractable. This is obviously an overly pessimistic assessment. First of all, we generally do not need to optimize over all functions for the continuous trajectories. Therefore, we make the assumption that we can instead optimize over a set of continuous variables by choosing appropriate techniques. To this end, we can, for example, use Bézier curves to represent the trajectories, as it is described in much of the aforementioned literature, e.g. Marcucci et al. (2024b). We have introduced these in Section 5. This transforms the problem above to a Mixed-Integer non-linear Program (MINLP). A similar formulation is possible for non-flat hybrid automata via numerical integration schemes. These MINLPs are among the most difficult optimization problems to solve, being provably NP-complete (Bürger et al. 2023). Although some progress in solver development has been made in recent years (Bestuzheva et al. 2025), for automation and control, we want to avoid this type of problem, if possible.

The structure of the optimization problem depends a lot on the type of guards that are present in the model. As also mentioned in Barton et al. (2006), the optimization problems for models with only controlled or only autonomous transitions have different structures: In the case of controlled transitions, there are no constraints on the guard sets (7.1d). Therefore, we have to optimize over the sequence of locations and the continuous trajectories together. However, the sequence of locations is determined by the discrete inputs only. When only autonomous transitions are present, the continuous trajectories and the sequence of locations are much more coupled: The end points of the continuous trajectories are constrained by the guard sets. Thus, the sequence of locations only depends on the continuous trajectories. The FHA model also allows for mixtures of the two extremes. Concerning numerical optimization, these extremes come with different challenges and opportunities. These differences motivate the different approaches in Sections 7.2 and 7.3.

The flatness property of our FHA models offers ways to simplify the problem in two dimensions: On the one hand, we can separate the path planning through the graph and the optimization of the continuous trajectories. We present this approach in Section 7.2. On the other hand, the differential flatness property of the continuous dynamics can allow for a reduction of complexity if the transitions have certain properties. In that case, Problem (7.1) can become a Mixed-Integer Convex Program (MICP), or even a Mixed-Integer Linear Program (MILP), which allows for direct computation of hybrid trajectories. We investigate this possibility in Section 7.3.

7.2 Decoupled Path and Trajectory Computation

Computing trajectories for HA with purely autonomous transitions is generally challenging because the feasible sequences of locations depend on the feasible continuous trajectories. A sequence is only feasible if all corresponding guard sets can be reached. Therefore, the location sequence and the continuous trajectories of the hybrid trajectories for HA have to be computed together. The flatness property of FHA solves this problem and allows for the separate computation of location sequences and continuous trajectories. In Definition 12, the reachability of the guard sets is defined as a property of FHA. For all sequences $(\mathsf{l}^j)_j$ that are allowed by the automaton graph, we can thus guarantee the existence of continuous trajectories and discrete inputs to realize them. This property allows for computing the sequence of locations and the continuous trajectories separately.

Depending on the system at hand, we can either choose sequences that represent the tasks or operation modes, or we can compute an optimized sequence. Choosing a sequence was demonstrated for the examples in Section 6.2. For now, we assume that a fixed sequence of length N is given, as well as initial and final states z^0, z^f. In this case, the optimization problem (7.1) reduces to an NLP, for which we can use approaches for differentially flat system trajectory computation referenced above.

$$\begin{aligned}
\min_{(z_i(\cdot)^i)_i, \{T_i\}} \quad & \sum_{i=1}^{N} c_i(z_i, T_i) && \text{(7.2a)}\\
\text{s.t.} \quad & \text{for all } i \in \{1, \ldots, N\} \\
& z_i(t) \in \mathsf{Flow}_{\mathsf{l}_i}, \quad \forall t \in [0, T_i] && \text{(7.2b)}\\
& z_{i+1}(0) = L_{\mathsf{e}}(z_i(T_i)), \quad \mathsf{e} = (\mathsf{l}_i, \mathsf{l}_{i+1}) && \text{(7.2c)}\\
& z_i(T_i) \in \mathcal{G}_{\mathsf{e}}, \quad \mathsf{e} = (\mathsf{l}_i, \mathsf{l}_{i+1}) && \text{(7.2d)}\\
& z_1(0) = z^0, z_N(T_N) = z^f && \text{(7.2e)}
\end{aligned}$$

In this formulation, we optimize over the continuous trajectories z_i, and the time T_i between transitions. The choices of the cost functions c_i and the representation of the continuous trajectories in (7.2b) have a major influence on the complexity of the optimization problem, as well as the type of reset functions (7.2c) and guard sets (7.2d).

When the flow set can be described as a polyhedron, constraint (7.2b) can be formulated as a linear inequality, using Bézier curves or exact discretization, as described in Chapter 5. If, in addition to that, the reset functions and the guard sets can be described by affine equations, the proper choice of the cost function can result in a simple LP or QP. Otherwise, the optimization problem is an NLP, which can also be solved with state-of-the-art solvers. Therefore, the setting in which the sequence of locations is given or precomputed, the optimization for the continuous trajectories is not of high complexity. In the following section, we demonstrate the computation in an example.

7.2.1 Example: 1-DoF Robot Thrower

We apply the decoupled trajectory generation to the robotic thrower example from Section 6.1.3. To compute a trajectory, we define optimization problems for the pick and the throw actions. These can be combined in arbitrary order, thanks to the flatness property of the model.

The robot has 4 different possible tasks, which are combinations of the two pick locations and the two target buckets. The model allows us to choose a sequence of these combinations in arbitrary order and to compute corresponding trajectories and inputs, as we demonstrate in the following. Each task consists of one pick action and one throw action. For each action, an optimized trajectory can be computed with a simple generic optimization problem. We formulate these optimization problems using Bézier curves. Each action is characterized by a vector of $N+1$ control points γ, representing the trajectory of the angle θ, the system's flat output. We constrain the trajectories to belong to the flow set and to be smoothly connected to their predecessor. Additionally, we include the constraints of the guard sets. All constraints can be formulated with the relations from Section 5.3.1. The optimization for the trajectory of the pick action is described as

$$\min_{\{\gamma\},\{T\}} \quad c(\gamma, T) \tag{7.3a}$$

$$\text{s.t.} \quad \gamma \in [\theta_{\text{min}}, \theta_{\text{max}}] \tag{7.3b}$$

$$\gamma^{(2)} \in [\ddot{\theta}_{\text{min}}, \ddot{\theta}_{\text{max}}] \tag{7.3c}$$

$$\gamma_0 = \gamma_N^-, \quad \gamma_0^{(1)} = \gamma_N^{(1)-}, \quad \gamma_0^{(2)} = \gamma_N^{(2)-} \tag{7.3d}$$

$$\gamma_N = \theta_{\text{pick}}, \quad \gamma_N^{(1)} = 0, \quad \gamma_N^{(2)} = 0 \tag{7.3e}$$

The notation $\gamma_i^{(j)}$ describes the i-th control point of the curves j-th derivative, as defined in Section 5.3.1. The constraint (7.3b) is the flow set constraints. The constraints on the second derivative in (7.3c) encode the constraints on the input torque τ. It is thus different for each location, depending on the weight of the object. Instead of including the input directly, we choose a convex inner approximation of the sets pictured in Figure 6.10, and described by Equation (6.11). Equations

(7.3d) and (7.3e) are the initial and final conditions. The initial condition is the connection to the previous trajectory.

For the throw action, we have a similar problem:

$$\min_{\{\gamma\},\{T\}} \quad c(\gamma, T) \tag{7.4a}$$

$$\text{s.t.} \quad \gamma \in [\theta_{\text{min}}, \theta_{\text{max}}] \tag{7.4b}$$

$$\gamma^{(2)} \in [\ddot{\theta}_{\text{min}}, \ddot{\theta}_{\text{max}}] \tag{7.4c}$$

$$\gamma_0 = \gamma_N^-, \quad \gamma_0^{(1)} = \gamma_N^{(1)-}, \quad \gamma_0^{(2)} = \gamma_N^{(2)-} \tag{7.4d}$$

$$L^2 \gamma_N^{(1)} \sin(2(\pi - \gamma_N)) + g d_i = 0 \tag{7.4e}$$

$$\gamma_N \in [\frac{3}{5}\pi, \frac{2}{5}\pi], \quad \gamma_N^{(1)} \leq 0 \tag{7.4f}$$

Equations (7.4e) and (7.4f) encode the non-linear throwing guard set. The cost functions for the two problems can include the control points, the derivatives, and the durations. Note that the durations T enter the problem via the derivatives $\gamma^{(j)}$, as described in Equation (5.13).

With these two optimization problems, we can form any sequence of tasks defined by the automaton graph in Figure 6.9. We choose a sequence of pick and throw tasks: pick 1, throw 2, pick 2, throw 1, pick 2, throw 2, pick 1, throw 1. The number denotes locations or buckets as defined in Section 6.1.3. From this task sequence, we obtain the corresponding sequence of locations: $\{\mathsf{l}_1, \mathsf{l}_2, \mathsf{l}_5, \mathsf{l}_1, \mathsf{l}_3, \mathsf{l}_6, \mathsf{l}_1, \mathsf{l}_3, \mathsf{l}_7, \mathsf{l}_1, \mathsf{l}_2, \mathsf{l}_4, \mathsf{l}_1\}$. The initial and final states are steady states at $\theta = \frac{\pi}{2}$. With these, build an optimization for the whole sequence, concatenation versions of Problems (7.3) and (7.4). As a cost function, we simply take the durations of each phase, i.e., $c(T_i) = T_i$. We thus compute the minimum time trajectory for the given sequence of tasks. The optimization problem is solved numerically with the open source solver IPOPT (Wächter and Biegler 2006).

The results are shown in Figure 7.1. We can see that the pick locations, throw angle, and velocities are reached as specified, while respecting the constraints on the input τ. The discrete inputs to trigger the transitions to the throwing tasks are not included in the plots because they are trivial.

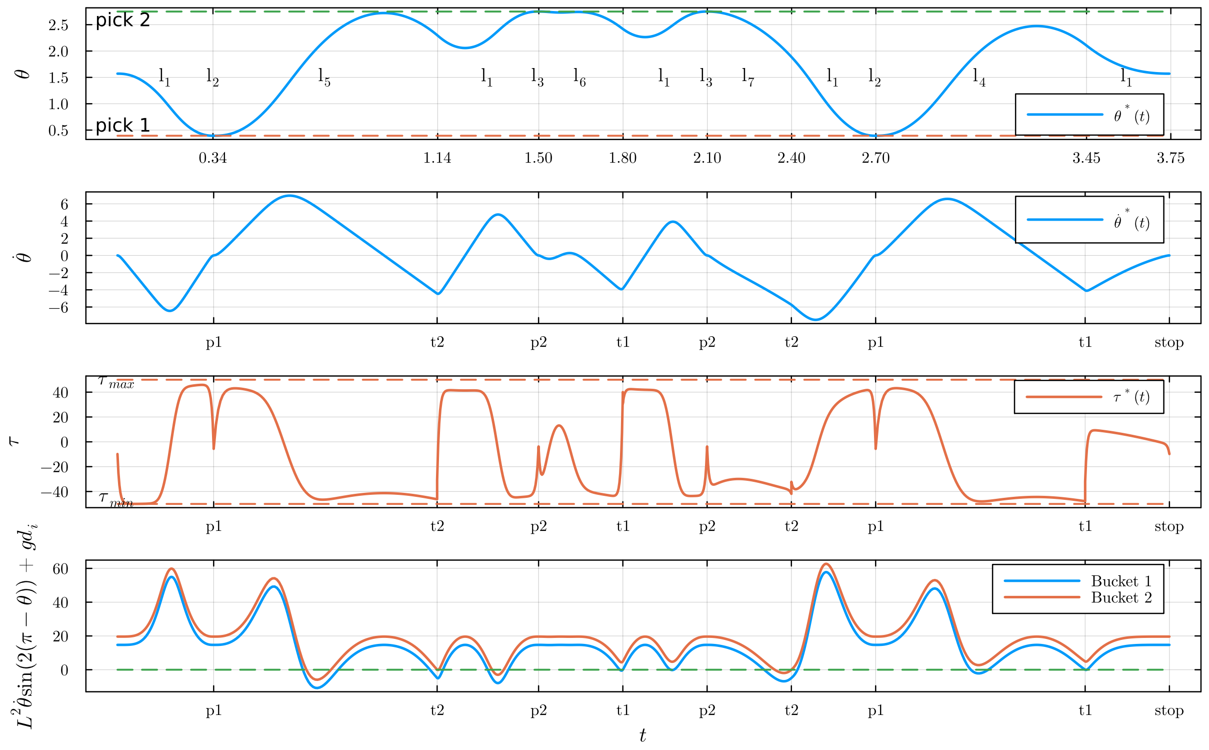


Abbildung 7.1: Optimized trajectory of the throwing robot. The bottom plot depicts the value of the throwing constraint. At the time of release, the value is always zero.

7.2.2 Discussion

We demonstrated how the flatness property of FHA allows for decoupled trajectory computation. For the sequence of locations, we manually chose the tasks and the order. This could also be automated for applications, leveraging the fact that all paths through the automaton graph are feasible. Computing optimized sequences is commonly known as the shortest path problem in graph theory. There are very efficient state-of-the-art algorithms to compute shortest paths, see e.g. Mehlhorn and Sanders (2008). However, these computations need some cost or distance measure for each transition, which depends on the optimization target. In a spatial navigation application, the costs may, for example, be approximations of physical distance, as proposed in Marcucci et al. (2024a). In the present work, we do not investigate this type of problem further.

Another way to compute sequences is the combination with control methods from discrete-event systems. Adopting a hierarchical control framework, it may be possible to combine supervisory control for the sequences of locations with numerical optimization for the continuous trajectories. Such approaches may be investigated in future work.

The decoupled computation approach is restricted to problems for which the sequences of locations and transitions can be precomputed. A significant advantage in this setting is that the length of the sequence is known before solving the optimization problem. Using Bézier curves as trajectories also allows the avoidance of smoothing and the detection of transitions, which are common in numerical integration for non-smooth ODEs (Nurkanovic 2023). In our approach using Bézier curves, the transitions are encoded at the junctions of the curves, and their durations give their transition times.

7.3 Integrated Path and Trajectory Computation

In some applications, it may not make sense to predefine the sequence of locations. When all transitions of a system are controlled, one may want to optimize over the discrete inputs and the continuous trajectories at the same time. We can again exploit the differential flatness property of the continuous dynamics, using discretization or Bézier curves. The general problem formulation for FHA for such an optimization is stated in Problem (7.1) above. In this case, we have to include the discrete decisions in the optimization problem, resulting in Mixed Integer Problems (MIP). For FHA with linear continuous dynamics discretized in time and with affine reset functions L_e and affine constraints, the problem can be formulated in the MLD framework (Bemporad and Morari (1999)), resulting in a Mixed Integer Quadratic Program (MIQP). This is guaranteed because of the differential flatness of the continuous dynamics, which allows for a linear representation (see Sections 2 and 5). If we need continuous solutions for the continuous dynamics, we have to use different trajectory representations. Combining Bézier curves and affine transition rules and constraints, we can transform the problem into an MICP, as described in Marcucci et al. (2024b). Both of these approaches can also handle autonomous transitions, as long as the guards can be formulated as affine equalities. However, if either the constraints or the transitions include non-linear functions, the problem remains a MINLP. Although these are hard to solve, for FHA we know at least that solutions exist, as opposed to trajectory generation problems for HA in general.

7.4 Example: Dynamic Optimal Transmission Switching

We return to the line switching example from Section 6.2.2. In the example, we did not address the questions of how to choose the line to switch and what an optimal trajectory is. As we stated above, switching lines can increase stability and lead to more efficient transmission. Computing whether switching a line improves one of these metrics leads to different mathematical formulations. In the following, we focus on the question of efficient operation through the lens of operational cost. In this context, we investigate the combination of two optimization problems: We can optimize over the topology, i.e., which line we switch, and over the trajectories for smooth line switching. Based on our approach to switch lines when they do not carry any load, we propose a combination of the two computations.

We choose the following set-up: We are given a standard grid model with generators, loads in the form of a case file from the power grid library benchmark repository[1]. To derive an FHA model of the grid, we modify the grid to make all nodes controllable: As motivated in Section 6.2.2 above, we assume all nodes to contain some form of energy storage as controllable power sources. Based on the load data in the case file, we create a load curve discretized over standard 15-minute intervals. For each interval, we compute the optimal topology, as we described below. Based on the output of the computation, we compute the corresponding continuous trajectories with or without switching. For the optimal trajectories, we use the general idea from Section 6.2.2, and we combine it with Bézier curve representations of the angle trajectories. We will provide further details for the example below.

The first problem, computing the optimal generation and corresponding topology, can be solved for different objectives. As we stated above, our objective is to optimize efficiency in terms of operational cost. For a given set of loads, we want to compute the optimal topology and the corresponding generation. This problem is known as Optimal Transmission Switching (OTS) (Taheri and Molzahn 2025). The OTS is based on the Optimal Power Flow (OPF) problem, which optimizes for the cheapest possible power generation for a given set of loads in steady-state operation. The two components of the OPF are the cost function, which is a function of generator output powers and their respective prices, and a model of the grid. The OTS introduces additional discrete variables into the grid model,

[1] `https://github.com/power-grid-lib/pglib-opf`, version 23.07

representing the switching of lines. The complexity of the OTS depends on the structure of the grid model. One widely applied approximation is the linear DC power flow approximation, resulting in an MILP for the OTS. For an overview of the different formulations, see Taheri and Molzahn (2025). In the following section, we present the mathematical formulations for the two problems.

Problem Formulation

We restate the dynamic second-order model with switching introduced above:

$$M_i\ddot{\delta}_i + D_i\dot{\delta}_i = P_i - \sum_{j=1, j\neq i}^{n} \mathsf{v}_{ij} b_{ij} \sin\left(\delta_i - \delta_j\right) \quad i \in \{1, \ldots, n\}.$$

In a steady state and for small angle differences, we can linearize the model to

$$0 = P_i - \sum_{j=1, j\neq i}^{n} \mathsf{v}_{ij} b_{ij} (\delta_i - \delta_j) \quad i \in \{1, \ldots, n\}, \tag{7.5}$$

which corresponds to the DC power flow equations (see e.g. Taheri and Molzahn (2025)). We solve an OTS with the DC power flow model as

$$\begin{aligned}
\min_{P_i^g, \mathsf{v}_{ij}, \delta_i} \quad & \sum_{i}^{n} c_i P_i^g + c_{switch} \sum_{j=1, j\neq i}^{n} \mathsf{v}_{ij} && \text{(7.6a)}\\
\text{s.t.} \quad & 0 = P_i^g - L_i - \sum_{j=1, j\neq i}^{n} \mathsf{v}_{ij} b_{ij} (\delta_i - \delta_j) \quad i \in [1, n] && \text{(7.6b)}\\
& 0 \leq P_i^g \leq \overline{P}_i^g && \text{(7.6c)}\\
& \underline{S}_{ij} \leq \mathsf{v}_{ij} b_{ij} (\delta_i - \delta_j) \leq \overline{S}_{ij} && \text{(7.6d)}\\
& \underline{\delta} \leq \delta_i \leq \overline{\delta} && \text{(7.6e)}\\
& \mathsf{v}_{ij} \in \{0, 1\} && \text{(7.6f)}\\
& \delta_1 = 0 && \text{(7.6g)}
\end{aligned}$$

with generator powers P_i^g, loads L_i, line power limits $\underline{S}_{ij}, \overline{S}_{ij}$ and costs c_i for generation and c_{switch} for switching. Equation (7.6b) is the DC power flow model derived above. Equations (7.6c), (7.6a), and (7.6e) constrain the generator powers, the line flows, and the angles. The maximum and minimum values, as well as the line parameters b_{ij}, the load and generator locations and the generator costs are given in the grid case files. The last constraint (7.6g) sets the reference angle of the bus with index one to zero. This formulation is adapted from Taheri and Molzahn (2025). For the solution of the problem, we eliminate the non-linear terms by adopting the reformulation and the big-M method described in Taheri and Molzahn (2025).

For the continuous trajectories, we solve one NLP for each time interval. We could also solve a single NLP aggregating all intervals, but we opt for a modular problem formulation. We represent the trajectory over a time interval as a concatenation of two Bézier curves for each bus angle. The two curves are smoothly joined at the switching time t^s. The initial and final angles are determined by the powers calculated from the OTS. The initial and the final times are determined by the intervals. We fix the switching time to be near the end of the interval at $t^s = 14$min. The optimization problem also includes the same constraints on angles and the line flows. A major difference is the introduction of virtual powers P_{virt} at each bus, representing the controllable power sources introduced above. The set points from the OTS at each bus are denoted $P_i^* = P_i^g - L_i$. In order to allow for an FHA model, we assume controllable bus power as motivated above, based on the microgrid assumption, see Section 6.2.2. We model this additional flexibility as virtual power P_{virt}, again representing a combination of storage and power control. To stay close to the power set points, we penalize deviations from these, thereby minimizing the usage of the virtual power. For the mathematical formulation of the optimization problem, we use a single trajectory $\delta_i(t)$, instead of the two Bézier curves, for the sake of readability. Based on these considerations, we derive the following optimization problem:[2]

$$\min_{\delta_i(\cdot)} \quad \sum_{i=1}^{N} w(\dot{\delta}_i(\cdot)^2 + \ddot{\delta}_i(\cdot)^2) + (P^* - P_{\text{virt}})^2 \tag{7.7a}$$

$$s.t. \quad \underline{S}_{ij} \leq \mathsf{v}_{ij}^{-} b_{ij} \sin(\delta_i(t) - \delta_j(t)) \leq \overline{S}_{ij} \tag{7.7b}$$

$$\underline{S}_{ij} \leq \mathsf{v}_{ij}^{+} b_{ij} \sin(\delta_i(t) - \delta_j(t)) \leq \overline{S}_{ij} \tag{7.7c}$$

$$\underline{\delta} \leq \delta_i(t) \leq \overline{\delta} \tag{7.7d}$$

$$\dot{\delta}_i(t^0), \ddot{\delta}_i(t^0), \dot{\delta}_i(t^f), \ddot{\delta}_i(t^f) = 0 \tag{7.7e}$$

$$\delta_i(t^s) - \delta_j(t^s) = 0, \quad \text{if } \mathsf{v}_{ij}^{-} - \mathsf{v}_{ij}^{+} \neq 0 \tag{7.7f}$$

$$\delta_1(t) = 0 \tag{7.7g}$$

The cost function aims to minimize deviations from the power set points and the first derivatives of the angles, in order to minimize the dynamic power changes. The angle derivatives are scaled by a weighting coefficient c to tune behavior. The constraints (7.7b),(7.7c),(7.7d), and (7.7g) are analogous to the constraints in the OTS (7.6). The constraints (7.7e) ensure steady state operation at the beginning and the end of the interval. The line flow for the switched lines is constrained by (7.7f). Note that because of the line constraints, in critical load situations, it may be impossible to make one line entirely load-free. In that case, we can adapt the optimization problem by making the angle difference constraint a soft constraint, i.e., replacing the constraint with a term in the cost function.

[2] Again, we use a simplified notation for the sake of simplicity. In a precise formulation, we would specify the times for which each constraint has to be satisfied and add that $i \in \{1, \ldots, n\}$.

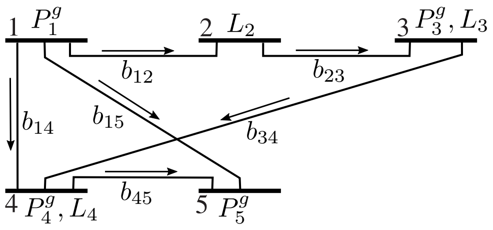


Abbildung 7.2: 5-bus grid topology.

5-Bus Example

We demonstrate the approach for the 5-bus grid from the pg-lib repository[3], which was first described by Li and Bo (2010), and is depicted in Figure 7.2. The grid has generators at busses 1, 3, 4 and 5, and loads at 2, 3 and 4. We use all parameters as stated in the case file with one small change: We aggregate the two generators at bus one to a single generator. The line parameters we use are given in Table 7.1. The admittances b_{ij} are computed from r and x given in the case file by the formula $b = \Im\left(\frac{-1}{r+\mathbf{j}x}\right)$ (Taheri and Molzahn 2025). The generator maximum powers and cost coefficients are given in Table 7.2. The base power for the scaling of the parameters to p.u. quantities is 100MVA. The generator parameters M_i and D_i are not included in the case file, and we choose them at random in the range provided by Dörfler and Bullo (2012).

In this configuration with the virtual power sources, the grid can be modeled as FHA, like the 3-bus example in Section 6.2.2. We also apply the constraint that only one line at a time can be opened. This constraint is not necessary, but it allows for a more compact model. The automaton graph thus resembles that of the 3-bus example above, with a central node in which all lines are closed, and with six leaf nodes representing the configuration with one opened line each. In this 5-bus example, all OTS solutions have at most one line opened in the

[3] `https://github.com/power-grid-lib/pglib-opf/blob/master/pglib_opf_case5_pjm.m`, version 23.07

Tabelle 7.1: Line parameters of 5-bus grid.

from i	to j	b_{ij} in p.u.	$\underline{S}_{ij}$ in MVA	$\overline{S}_{ij}$ in MVA
1	2	35.235	-400	400
1	4	32.569	-426	426
1	5	154.703	-426	426
2	3	91.676	-426	426
2	4	33.337	-426	426
4	5	33.337	-240	240

Tabelle 7.2: Generator parameters of 5-bus grid.

bus i	$\overline{P}_i^g$ in MW	c_i in €/MW
1	210	15
3	520	30
4	200	40
5	600	10

load situations we simulated anyway. For larger grids, one should consider less restrictive constraints.

We choose a sequence of loads for the three load buses over 7 time intervals, as depicted in Figure 7.3. For each interval, we solve the OTS with the mixed integer solver Gurobi (Gurobi Optimization 2024) and the NLP solver IPOPT (Wächter and Biegler 2006) for the trajectory computation problems as described above. The resulting trajectories are depicted in Figure 7.4. There are two intervals, in which lines are opened: At minute 14, the line between buses 3 and 4 is opened, before being closed again at minute 29. After a two-minute interval without switching, the line between buses 2 and 3 is opened at minute 59, before being closed again at minute 74. We see in the angle plots that the corresponding angles at the switching times are equal, as we optimized for. We can further observe that the optimized trajectories lead to minimal changes in the power outputs, which for

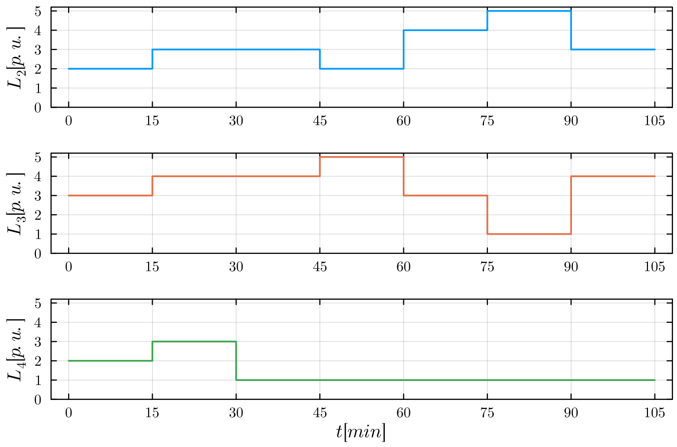


Abbildung 7.3: Load curves for loads at busses 2, 3 and 4.

the first switching even blends nicely with the necessary set point change between the intervals.

7.5 Summary and Discussion

We investigate the computation of trajectories for FHA with numerical optimization methods. To this end, we first formulate the general optimization problem, and we simplify it by using Bézier curves for the continuous trajectories. Based on this formulation, we discuss the optimization problem for fixed sequences of locations. We illustrate the theory with the robotic thrower example introduced in Section 6.1.3, computing optimized trajectories for a given sequence of tasks. We also describe the optimization problem for the simultaneous computation of sequences and continuous trajectories. This problem is potentially much harder to solve, and we analyze different problem settings. We end by revisiting the line switching example from Section 6.2.2 and optimizing over both the decision which line to switch and the corresponding trajectories.

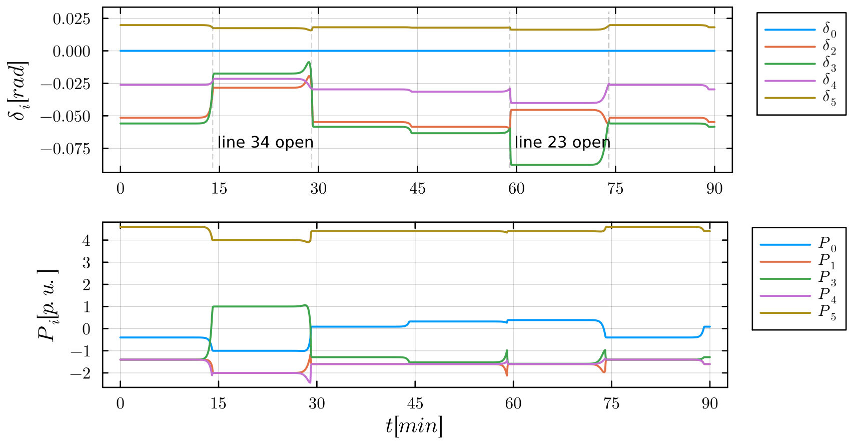


Abbildung 7.4: Angles and net powers for each bus. The switching times are depicted as vertical dashed lines.

Although we show that the flatness property of FHA can greatly simplify the trajectory computation via numerical optimization, the nature of the optimization problem depends on the specific mathematical structures of the associated constraints. Furthermore, whether the sequence of locations is fixed or free greatly affects the complexity of the computation. For some FHA models and the associated optimization tasks, the problem remains quite challenging. Not only does the complexity of the problem depend on these factors, but also the type of optimization we can use changes. We only present two approaches for two specific examples, but in reality, a variety of methods may be suitable for different FHA applications. We suppose that some existing methods may be applied to FHA, like for example approaches from supervisory control (Moor and Raisch 1999).

8 Summary and Outlook

The aim of this thesis is to apply the concept of differential flatness to HA. To this end, we cover three key areas, which are a theory of flatness for HA, its application to examples, and algorithms for trajectory generation and inversion. We start with an introduction to continuous dynamics described by ODE and revisit differential flatness from two perspectives: a classic definition and a definition in the behavioral framework. We emphasize that differentially flat systems can be characterized by trajectories of their flat output, from which the corresponding inputs can be computed by an inversion. The freeness of the trajectories and the inversion are the guiding principles for our development of a flatness definition for HA. In a first step, we define HA and all its different parts, leading to a definition of HA that we use to derive our definition of flatness. Since a definition of flatness for HA cannot be equivalent to differential flatness due to their different structure, our definition is built on a partitioning of the HA into three domains: The continuous dynamics, the discrete dynamics, and the transitions. For each domain, we develop a notion of flatness, which we add together to form a new model class, called Flat Hybrid Automata (FHA). A separate contribution to this development is a definition of flatness for automata, which uses the flatness definition of the behavioral framework. The resulting FHA model is based on properties of the three domains, including the reachability of guards causing transitions. This property is generally not easy to verify.

To simplify this reachability problem, we leverage the flatness property of the continuous dynamics of FHA. Specifically, we built on the fact that any sufficiently smooth function can be a flat output trajectory to provide theoretical results and numerical computations to evaluate reachability of guards. The theoretical results cover the situation where initial states and guards are steady states. In that case, the evaluation of reachability is fairly simple. For other scenarios, we propose two methods based on numerical optimization. These are particularly simple when all constraints are convex, but they can also be applied in non-convex settings. The numerical methods and the theoretical results are applied to an academic example and to the examples we present in Chapters 6 and 7.

From the theory of FHA, we build a bridge into applications by presenting examples from different engineering domains. The examples are an electrical DC circuit, a tank, and a robotic manipulator. For these examples, we show in detail how to model a system as FHA and what design choices can be made. We emphasize different aspects of design and reachability computation. FHA allow for inversion to compute inputs from trajectories, just like differentially flat continuous systems. We describe the associated steps and apply the inversion to two examples, namely the tank example and an electrical AC grid.

In order to compute inversions, we need a feasible trajectory. Trajectory computation for differentially flat systems is often much simpler than for general non-linear systems. We exploit this fact and build on existing methods from the literature to formulate the trajectory computation as an optimization problem. The complexity of the corresponding problems depends heavily on the specific systems and the planning task. We give an overview of the associated challenges and categorize their sources. Additionally, for two specific types of problems we present the particular optimization problem and solve it for one example each. The first type is problems in which the sequence of locations is fixed For these, the trajectories can be easily computed as a standard NLP. This scenario is demonstrated for the robotic manipulator example. The second type, which combines computing the optimal sequence and the corresponding continuous trajectories together, is more challenging. We present a solution to this problem for Optimal Transmission Switching (OTS). More precisely, instead of solving the whole MINLP, we decompose the solutions into two steps and apply this approach to a 5-bus example from the literature.

Summarizing, this thesis is the first step towards a theory of FHA. We present a foundational theory and demonstrate its application to examples. This opens the door to many more open questions and potential further research, which will hopefully bridge the gap into real-world applications.

Future Work

The results and ideas of our work give rise to many open questions and directions. In the following, we present our thoughts and ideas in the same order as the content of this thesis: We start with the theory, continue with modeling and applications, and finish with algorithms and control.

In our work, we provide a definition of FHA and discuss the resulting properties. For further research on control for FHA, the model needs to include real-world phenomena such as model mismatch, measurement noise, and actuation errors. Therefore, in future work, a model including noise and disturbances has to be derived, and the stability of the feedforward control has to be analyzed. Further, it should be investigated how to apply concepts from control theory, such as observers, and robustness, in the FHA model.

To allow the model to represent more realistic systems, the FHA definition could potentially be extended to systems that have locations with autonomous or other non-flat continuous dynamics, while preserving flatness of the automaton. If, for example, the autonomous dynamics are provably converging to a guard, one could integrate them into an FHA, even though they are technically not flat. Such an approach could make the model more versatile.

Another line of research may investigate how to apply the FHA theory to other hybrid system models such as piecewise-affine systems, MLD, or switched systems. In this way, results from our work and results from these other hybrid system classes could be combined.

Modeling FHA for a given technical system requires writing down all parts of the HA and checking their properties. For industrial applications, it would be beneficial to automate the modeling process at least in part, as well as to check the flatness properties. This may include data-driven approaches, which have already been applied to differentially flat continuous systems and to HA in the literature. In industrial environments, one also frequently encounters many interconnected systems that interact with each other. The interconnection of multiple FHA could be investigated, as has been done for discrete-event systems and hybrid systems in the literature.

An important next step for FHA is the experimental validation of the approach. To this end, the modeling and trajectory computation approaches we present have to be tested in experiments. This certainly requires more work on the specific control architectures. One direction of research could be the extension of the presented optimization methods for trajectory computation to a receding horizon controller in form of an MPC. Besides the presented approaches, there is potential to apply other approaches, like supervisory control and other hierarchical schemes. Because of the properties of FHA, we can separate the discrete and the continuous dynamics, which allows for separate control approaches. This may be exploited by applying hierarchical control schemes, by forming appropriate abstractions for supervisory control, or by creating specialized multi-layer algorithms to solve the problem efficiently. Another line of research can investigate methods based on machine learning, such as reinforcement learning, to conquer the complexity of the mixed-integer problems.

The unique advantages of FHA lie in the modular treatment of the locations, the reachability of the automaton graph and the flatness of the continuous dynamics. It is certainly possible to exploit these advantages in many different ways, many of which may already exist. Future research will have to find these approaches and test their effectiveness in practice.

Abbildungsverzeichnis

2.1 Schema of a planar one-degree-of-freedom robotic manipulator. . . . 17

3.1 Depiction of basic elements of our hybrid automaton model. 32
3.2 Two options for the replacement of a mixed guard set. 33

4.1 Example of equivalent paths in a strongly connected graph $\{\mathsf{L}, \mathsf{E}\}$ and a complete graph $\{\mathsf{D}, \hat{\mathsf{E}}\}$. 42
4.2 Illustration of Example 2. The grey areas are the guard sets. 45

5.1 Flow set and obstacles for mobile robot example. 60
5.2 Left: Topology and path for mobile robot example. Right: Velocities and accelerations for the trajectory, with constraints as dotted red lines. 61
5.3 Valid initial states for Example 4 in blue. The red shaded area shows initial states for which no trajectory exists that does not violate the constraints. 63
5.4 Grid points for reachability computation of area in box constraints. . 73
5.5 Feasible initial points for different parameters. The solutions for the initial values marked as large red dots are depicted in Figure 5.6 below. 74
5.6 Trajectory plots: (a) and (c) show the effects of increasing the order N for Bézier curves. (b) shows a discretized solution for large time steps and the continuous-time solution for the corresponding piecewise constant input (dashed lines), which violates the constraints. (d) shows a solution for small step sizes. 76

6.1 DC network with loads R_1, R_2, R_3, capacitor C, inductance L, power sources $V_{\text{in},1}, V_{\text{in},2}$ and two switches $\mathsf{v}_1, \mathsf{v}_2$. 80
6.2 Automaton graph of the DC network model. 83
6.3 Left: Trajectory of the flat outputs in the state space. The dotted lines indicate the switching sets. Right: Trajectories of the flat outputs and the continuous inputs over time (Solid lines: v_C, u_1; dashed lines: i_L, u_2). 86
6.4 Automaton graph reduction for the DC network model. 87
6.5 One-tank system setup. 88
6.6 Automaton graph and plot of the locations depending on the flat output and the discrete input of the tank example. 89
6.7 Depiction of possible derivatives $\dot{l}$ for all locations and levels l 92
6.8 Schema of the throwing robot. Positive directions of θ and g are indicated by arrows. 94
6.9 Automaton graph of the throwing robot. 96
6.10 Input constraints of robotic thrower mapped to the $\theta - \ddot{\theta}$ plane for free movement, and movement with one of the two objects. . . . 98
6.11 Reachability computation based on Bézier curves for robotic thrower for different locations. 99
6.12 Liquid level and inputs for the one-tank example. The green line shows the discrete input v, the red line shows the continuous input u. The dashed gray lines mark the times of the state switching. . 102
6.13 Simulation of feedforward of the computed inputs with model mismatch and disturbance. 103
6.14 Simulation of feedforward with PI-controller for tank example. The simulation includes model mismatch and disturbance. 104
6.15 a) 3 bus AC microgrid with power injections P_i, angles δ_i and line susceptances b_{ij}. The arrows denote the direction of positive power transmitted by the lines. b) Automaton graph of the FHA model of the power network. 106
6.16 Trajectories of the angles, input signals and line loads for the line switching feedforward example. 110
6.17 Transient response to opening line 12 at $t = 0$. 111

7.1 Optimized trajectory of the throwing robot. The bottom plot depicts the value of the throwing constraint. At the time of release, the value is always zero. 123
7.2 5-bus grid topology. 130
7.3 Load curves for loads at busses 2, 3 and 4. 132
7.4 Angles and net powers for each bus. The switching times are depicted as vertical dashed lines. 133

Tabellenverzeichnis

6.1 Guards for the electrical DC switched circuit. 84
6.2 Parameters of the throwing robot. 96

7.1 Line parameters of 5-bus grid. 131
7.2 Generator parameters of 5-bus grid. 131

List of Publications

Kleinert, T., Zahn, F., and Hagenmeyer, V. (2020). On complexity reduction of the discrete-event subsystem of flat hybrid automata for control design. *at - Automatisierungstechnik*, 68(7):529–540.

Zahn, F., Kleinert, T., and Hagenmeyer, V. (2020). On optimal control of flat hybrid automata. *IFAC-PapersOnLine*, 53(2):6800–6805. 21st IFAC World Congress.

Zahn, F. and Hagenmeyer, V. (2022). Modelling power systems as flat hybrid automata for controlled line switching. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, e-Energy '22, page 302–306, New York, NY, USA. Association for Computing Machinery.

Zahn, F., Kleinert, T., and Hagenmeyer, V. (2022). Assessing the combination of differential flatness and deterministic automata for controllable hybrid systems. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2612–2619.

Zahn, F., Kleinert, T., and Hagenmeyer, V. (2024). Feedforward control of flat hybrid automata: a behavioral systems theory approach. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 6238–6244.

Bibliography

Agrawal, D. R., Parwana, H., Cosner, R. K., Rosolia, U., Ames, A. D., and Panagou, D. (2022). A Constructive Method for Designing Safe Multirate Controllers for Differentially-Flat Systems. *IEEE Control Systems Letters*, 6:2138–2143.

Althoff, M., Frehse, G., and Girard, A. (2021). Set Propagation Techniques for Reachability Analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1).

Althoff, M., Stursberg, O., and Buss, M. (2010). Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249.

Alur, R. (2011). Formal verification of hybrid systems. In *Proceedings of the Ninth ACM International Conference on Embedded Software*, pages 273–278, Taipei Taiwan. ACM.

Alur, R., Courcoubetis, C., Henzinger, T. A., and Ho, P. H. (1993). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Goos, G., Hartmanis, J., Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, *Hybrid Systems*, volume 736, pages 209–229. Springer Berlin Heidelberg, Berlin, Heidelberg.

Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235.

Bang-Jensen, J. and Gutin, G. (2007). Theory, algorithms and applications. *Springer Monographs in Mathematics, Springer-Verlag London Ltd., London*, 101.

Barton, P. I., Lee, C. K., and Yunt, M. (2006). Optimization of hybrid systems. *Computers & Chemical Engineering*, 30(10-12):1576–1589.

Beaver, L. E. and Malikopoulos, A. A. (2024). Optimal control of differentially flat systems is surprisingly easy. *Automatica*, 159:111404.

Bemporad, A., Ferrari-Trecate, G., and Morari, M. (2000). Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45(10):1864–1876.

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427.

Bergen, A. R. and Hill, D. J. (1981). A structure preserving model for power system stability analysis. *IEEE transactions on power apparatus and systems*, (1):25–35.

Berman, A. and Plemmons, R. (1979). *Nonnegative Matrices in the Mathematical Sciences*. Academic Press, New York.

Bernard, P. and Sanfelice, R. G. (2020). Hybrid dynamical systems with hybrid inputs: Definition of solutions and applications to interconnections. *International Journal of Robust and Nonlinear Control*, 30(15):5892–5916.

Bernard, P. and Sanfelice, R. G. (2022). Observer design for hybrid dynamical systems with approximately known jump times. *Automatica*, 141:110225.

Bernard, P. and Sanfelice, R. G. (2024). Semiglobal High-Gain Hybrid Observer for a Class of Hybrid Dynamical Systems With Unknown Jump Times. *IEEE Transactions on Automatic Control*, 69(9):5804–5819.

Bestuzheva, K., Chmiela, A., Müller, B., Serrano, F., Vigerske, S., and Wegscheider, F. (2025). Global optimization of mixed-integer nonlinear programs with SCIP 8. *Journal of Global Optimization*, 91(2):287–310.

Bindel, R., Nitsche, R., Rothfuß, R., and Zeitz, M. (2000). Flachheitsbasierte Regelung eines hydraulischen Antriebs mit zwei Ventilen für einen Großmanipulator (Flatness Based Control of a Two Valve Hydraulic Joint of a Large Manipulator). 48(3):124.

Brailsford, S. C., Potts, C. N., and Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119(3):557–581.

Branicky, M., Borkar, V., and Mitter, S. (1998). A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45.

Bruno, S., D'Aloia, M., De Carne, G., and La Scala, M. (2012). Controlling transient stability through line switching. In *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, pages 1–7.

Bürger, A., Zeile, C., Altmann-Dieses, A., Sager, S., and Diehl, M. (2023). A Gauss–Newton-based decomposition algorithm for Nonlinear Mixed-Integer Optimal Control Problems. *Automatica*, 152:110967.

Buss, M. and Schlegl, T. (2000). A discrete-continuous control approach to dextrous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 276–281, San Francisco, CA, USA. IEEE.

Caines, P. E. and Lemch, E. S. (2002). On the Global Controllability of Nonlinear Systems: Fountains, Recurrence, and Applications to Hamiltonian Systems. *SIAM Journal on Control and Optimization*, 41(5):1532–1553.

Diwold, J., Kolar, B., and Schöberl, M. (2022). A Trajectory-Based Approach to Discrete-Time Flatness. *IEEE Control Systems Letters*, 6:289–294.

Dörfler, F. and Bullo, F. (2012). Synchronization and Transient Stability in Power Networks and Nonuniform Kuramoto Oscillators. *SIAM Journal on Control and Optimization*, 50(3):1616–1642.

Doyen, L., Frehse, G., Pappas, G. J., and Platzer, A. (2018). Verification of Hybrid Systems. In Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., editors, *Handbook of Model Checking*, pages 1047–1110. Springer International Publishing, Cham.

Egerstedt, M. (2000). Behavior Based Robotics Using Hybrid Automata. In Goos, G., Hartmanis, J., van Leeuwen, J., Lynch, N., and Krogh, B. H., editors, *Hybrid Systems: Computation and Control*, volume 1790, pages 103–116. Springer Berlin Heidelberg, Berlin, Heidelberg.

Egerstedt, M. and Hu, X. (2002). A hybrid control approach to action coordination for mobile robots. *Automatica*, 38(1):125–130.

Engell, S., Kowalewski, S., Schulz, C., and Stursberg, O. (2000). Continuous-discrete interactions in chemical processing plants. *Proceedings of the IEEE*, 88(7):1050–1068.

Ezzine, J. and Haddad, A. H. (1989). Controllability and Observability of Hybrid Systems. *International Journal of Control*, 49(6):2045–2055.

Faessler, M., Franchi, A., and Scaramuzza, D. (2018). Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626.

Faulwasser, T., Hagenmeyer, V., and Findeisen, R. (2014). Constrained reachability and trajectory generation for flat systems. *Automatica*, 50(4):1151–1159.

Fliess, M. and Glad, S. T. (1993). An Algebraic Approach to Linear and Nonlinear Control. In Trentelman, H. L. and Willems, J. C., editors, *Essays on Control*, pages 223–267. Birkhäuser Boston, Boston, MA.

Fliess, M., Join, C., and Sira-Ramírez, H. (2005). Closed-Loop Fault-Tolerant Control for Uncertain Nonlinear Systems. In Meurer, T., Graichen, K., and Gilles, E. D., editors, *Control and Observer Design for Nonlinear Finite and Infinite Dimensional Systems*, volume 322, pages 217–233. Springer-Verlag, Berlin/Heidelberg.

Fliess, M., Levine, J., Martin, P., and Rouchon, P. (1995). Flatness and defect of nonlinear systems: Introductory theory and examples. *Int. Journal of Control*, 61(6):1327–1361.

Fliess, M., Levine, J., Martin, P., and Rouchon, P. (1999a). A Lie-Bäcklund approach to equivalence and flatness of nonlinear systems. *IEEE Transactions on Automatic Control*, 44(5):922–937.

Fliess, M., Lévine, J., Martin, P., and Rouchon, P. (1999b). Some open questions related to flat nonlinear systems. In Dickinson, B. W., Fettweis, A., Massey, J. L., Modestino, J. W., Sontag, E. D., Thoma, M., Blondel, V., Sontag, E. D., Vidyasagar, M., and Willems, J. C., editors, *Open Problems in Mathematical Systems and Control Theory*, pages 99–103. Springer London, London.

Fourlas, G., Kyriakopoulos, K., and Vournas, C. (2004). Hybrid systems modeling for power systems. *IEEE Circuits and Systems Magazine*, 4(3):16–23.

Gensior, A., Woywode, O., Rudolph, J., and Guldner, H. (2006). On Differential Flatness, Trajectory Planning, Observers, and Stabilization for DC-DC Converters. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(9):2000–2010.

Goebel, R., Sanfelice, R. G., and Teel, A. R. (2009). Hybrid dynamical systems. *IEEE Control Systems*, 29(2):28–93.

Greco, L., Mounier, H., and Bekcheva, M. (2022). An approximate characterisation of the set of feasible trajectories for constrained flat systems. *Automatica*, 144:110484.

Greeff, M. and Schoellig, A. P. (2018). Flatness-Based Model Predictive Control for Quadrotor Trajectory Tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6740–6745, Madrid. IEEE.

Gurobi Optimization, LLC. (2024). Gurobi Optimizer Reference Manual.

Hagenmeyer, V. and Delaleau, E. (2003a). Exact feedforward linearisation based on differential flatness: The SISO case. In Zinober, A. and Owens, D., editors, *Nonlinear and Adaptive Control*, volume 281, pages 161–170. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hagenmeyer, V. and Delaleau, E. (2003b). Exact feedforward linearization based on differential flatness. *International Journal of Control*, 76(6):537–556.

Hagenmeyer, V. and Delaleau, E. (2003c). Robustness analysis of exact feedforward linearization based on differential flatness. *Automatica*, 39(11):1941–1946.

Hagenmeyer, V. and Delaleau, E. (2008). Continuous-Time Non-Linear Flatness-Based Predictive Control: An Exact Feedforward Linearisation Setting with an Induction Drive Example. *International Journal of Control*, 81(10):1645–1663.

Hagenmeyer, V. and Delaleau, E. (2010). Robustness analysis with respect to exogenous perturbations for flatness-based exact feedforward linearization. *IEEE Transactions on automatic control*, 55(3):727–731.

Hagenmeyer, V. and Nohr, M. (2008). Flatness-based two-degree-of-freedom control of industrial semi-batch reactors using a new observation model for an extended Kalman filter approach. *International Journal of Control*, 81(3):428–438.

Hagenmeyer, V. and Zeitz, M. (2004). Flachheitsbasierter Entwurf von linearen und nichtlinearen Vorsteuerungen (Flatness-based Design of Linear and Nonlinear Feedforward Controls). *at - Automatisierungstechnik*, 52(1):3–12.

Hedlund, S. and Rantzer, A. (2002). Convex dynamic programming for hybrid systems. *IEEE Transactions on Automatic Control*, 47(9):1536–1540.

Henke, B., Rueß, A., Neumann, R., Zeitz, M., and Sawodny, O. (2014). Flatness-based MIMO control of hybrid stepper motors - design and implementation. In *2014 American Control Conference*, pages 347–352.

Hofmann, S. and Raisch, J. (2012). Solutions to inversion problems in preferential crystallization of enantiomers—part I: Batch crystallization in a single vessel. *Chemical Engineering Science*, 80:253–269.

Hofmann, S. and Raisch, J. (2013). Solutions to inversion problems in preferential crystallization of enantiomers—Part II: Batch crystallization in two coupled vessels. *Chemical Engineering Science*, 88:48–68.

Horn, J., Bamberger, J., Michau, P., and Pindl, S. (2003). Flatness-based clutch control for automated manual transmissions. *Control Engineering Practice*, 11(12):1353–1359.

Huang, Garng. M., Wang, W., and An, J. (2014). Stability Issues of Smart Grid Transmission Line Switching. *IFAC Proceedings Volumes*, 47(3):7305–7310.

Kastner, A., Gröll, L., and Hagenmeyer, V. (2023). Flatness of Some DC Microgrid Topologies. In *2023 8th IEEE Workshop on the Electronic Grid (eGRID)*, pages 1–6.

Keller, R., Baader, F. J., Bardow, A., Müller, M., and Peters, R. (2025). Experimental demonstration of dynamic demand response scheduling for PEM-electrolyzers. *Applied Energy*, 393:126014.

Khalil, H. (2002). *Nonlinear Systems*. Pearson Education. Prentice Hall.

Kleinert, T. and Lunze, J. (2002). A hybrid automaton representation of simulated counterflow chromatographic separation processes. In *15th IFAC World Congress Proceedings*, Barcelona, Spain.

Kleinert, T., Zahn, F., and Hagenmeyer, V. (2020). On complexity reduction of the discrete-event subsystem of Flat Hybrid Automata for control design. *at - Automatisierungstechnik*, 68(7):529–540.

Kolar, B., Rams, H., and Schlacher, K. (2017). Time-optimal flatness based control of a gantry crane. *Control Engineering Practice*, 60:18–27.

Koutsoukos, X., Antsaklis, P., Stiver, J., and Lemmon, M. (2000). Supervisory control of hybrid systems. *Proceedings of the IEEE*, 88(7):1026–1049.

Kundur, P. S. (2012). Power System Stability. In *Power System Stability and Control*. CRC Press, 3 edition.

Lemch, E., Sastry, S., and Caines, P. (2000). On the Global Controllability of Hybrid Systems: Hybrifolds and Fountains. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, volume 1, pages 981–986, Sydney, NSW, Australia. IEEE.

Levine, J. (2009). *Analysis and Control of Nonlinear Systems: A Flatness-Based Approach*. Springer Science & Business Media.

Lévine, J. (2011). On necessary and sufficient conditions for differential flatness. *Applicable Algebra in Engineering, Communication and Computing*, 22(1):47–90.

Li, F. and Bo, R. (2010). Small test systems for power system economic studies. In *IEEE PES General Meeting*, pages 1–4.

Liñán, M. B., Cortés, J., de Diego, D. M., Martínez, S., and Lecanda, M. M. (2020). Global Controllability Tests for Geometric Hybrid Control Systems. *Nonlinear Analysis: Hybrid Systems*, 38:100935.

Lunze, J. and Lamnabhi-Lagarrigue, F., editors (2009). *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge University Press, Cambridge, UK ; New York.

Lygeros, J., Tomlin, C., and Sastry, S. (1997). Multiobjective hybrid controller synthesis. In Goos, G., Hartmanis, J., Van Leeuwen, J., and Maler, O., editors, *Hybrid and Real-Time Systems*, volume 1201, pages 109–123. Springer Berlin Heidelberg, Berlin, Heidelberg.

Manetti, M. (2023). *Topology*, volume 153 of *UNITEXT*. Springer Nature Switzerland, Cham.

Marcucci, T. (2024). *Graphs of Convex Sets with Applications to Optimal Control and Motion Planning*. PhD thesis, Massachusetts Institute of Technology.

Marcucci, T., Nobel, P., Tedrake, R., and Boyd, S. (2024a). Fast Path Planning Through Large Collections of Safe Boxes. *IEEE Transactions on Robotics*, 40:3795–3811.

Marcucci, T., Petersen, M., von Wrangel, D., and Tedrake, R. (2023). Motion planning around obstacles with convex optimization. *Science Robotics*, 8(84):eadf7843.

Marcucci, T., Umenberger, J., Parrilo, P., and Tedrake, R. (2024b). Shortest paths in graphs of convex sets. *SIAM Journal on Optimization*, 34(1):507–532.

Mehlhorn, K. and Sanders, P. (2008). *Algorithms and Data Structures*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Milam, M., Mushambi, K., and Murray, R. (2000). A new computational approach to real-time trajectory generation for constrained mechanical systems. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, volume 1, pages 845–851, Sydney, NSW, Australia. IEEE.

Millerioux, G. and Daafouz, J. (2007). Invertibility and Flatness of Switched Linear Discrete-Time Systems. In *Hybrid Systems: Computation and Control: 10th International Workshop, HSCC 2007, Pisa, Italy, April 3-5, 2007. Proceedings 10*, pages 714–717. Springer.

Millerioux, G. and Daafouz, J. (2009). Flatness of Switched Linear Discrete-Time Systems. *IEEE Transactions on Automatic Control*, 54(3):615–619.

Moor, T., Davoren, J. M., and Raisch, J. (2001). Modular supervisory control of a class of hybrid systems in a behavioural framework. In *2001 European Control Conference (ECC)*, pages 870–875, Porto, Portugal. IEEE.

Moor, T. and Raisch, J. (1999). Supervisory control of hybrid systems within a behavioural framework. *Systems & Control Letters*, 38(3):157–166.

Mrochen, M. A. and Sawodny, O. (2019). Flatness-based Powertrain Control for Engine Start Applications in Hybrid Dual-Clutch Transmissions. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1562–1567.

Nenchev, V., Belta, C., and Raisch, J. (2015). Optimal motion planning with temporal logic and switching constraints. In *2015 European Control Conference (ECC)*, pages 1141–1146.

Nerode, A. and Greenberg, N. (2022). *Algebraic Curves and Riemann Surfaces for Undergraduates: The Theory of the Donut*. Springer International Publishing, Cham.

Nicolau, F., Respondek, W., and Barbot, J.-P. (2020). Flat Inputs: Theory and Applications. *SIAM Journal on Control and Optimization*, 58(6):3293–3321.

Noda, Y., Zeitz, M., Sawodny, O., and Terashima, K. (2011). Flow rate control based on differential flatness in automatic pouring robot. In *2011 IEEE International Conference on Control Applications (CCA)*, pages 1468–1475.

Nurkanovic, A. (2023). Numerical methods for optimal control of nonsmooth dynamical systems.

Oldenburg, J. and Marquardt, W. (2002). Flatness and higher order differential model representations in dynamic optimization. *Computers & chemical engineering*, 26(3):385–400.

Petit, N., Milam, M. B., and Murray, R. M. (2001). Inversion Based Constrained Trajectory Optimization. *IFAC Proceedings Volumes*, 34(6):1211–1216.

Platzer, A. (2010). *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Platzer, A. (2018). *Logical Foundations of Cyber-Physical Systems*. Springer International Publishing, Cham.

Raisch, J. and Moor, T. (2005). Hierarchical Hybrid Control Synthesis and its Application to a Multiproduct Batch Plant. In Meurer, T., Graichen, K., and Gilles, E. D., editors, *Control and Observer Design for Nonlinear Finite and Infinite Dimensional Systems*, volume 322, pages 199–216. Springer-Verlag, Berlin/Heidelberg.

Raisch, J. and O'Young, S. (1998). Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 43(4):569–573.

Ramadge, P. and Wonham, W. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98.

Rouchon, P. and Sira-Ramirez, H. (2003). Control of the walking toy: A flatness approach. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 3, pages 2018–2023, Denver, CO, USA. IEEE.

Rungger, M. and Stursberg, O. (2011). A numerical method for hybrid optimal control based on dynamic programming. *Nonlinear Analysis: Hybrid Systems*, 5(2):254–274.

Sanfelice, R. G., Teel, A. R., and Sepulchre, R. (2007). A hybrid systems approach to trajectory tracking control for juggling systems. In *2007 46th IEEE Conference on Decision and Control*, pages 5282–5287.

Scherer, P., Irscheid, A., Rizzello, G., and Rudolph, J. (2020). Flatness-Based Trajectory-Tracking Control of Dielectric Elastomer Actuators. *IFAC-PapersOnLine*, 53(2):8757–8762.

Schiffer, J., Zonetti, D., Ortega, R., Stanković, A. M., Sezi, T., and Raisch, J. (2016). A survey on modeling of microgrids—From fundamental physics to phasors and voltage sources. *Automatica*, 74:135–150.

Shaikh, M. S. and Caines, P. E. (2003). On the Optimal Control of Hybrid Systems: Optimization of Trajectories, Switching Times, and Location Schedules. In Goos, G., Hartmanis, J., Van Leeuwen, J., Maler, O., and Pnueli, A., editors, *Hybrid Systems: Computation and Control*, volume 2623, pages 466–481. Springer Berlin Heidelberg, Berlin, Heidelberg.

Sira-Ramirez, H. and Silva-Ortigoza, R. (2002). On the Control of the Resonant Converter: A Hybrid-Flatness Approach. In *15th Int. Symp. on Mathematical Theory of Networks and Systems (MTNS)*. Notre Dame, Indiana (USA).

Sira Ramírez, H. J. and Agrawal, S. K. (2004). *Differentially Flat Systems*. Control Engineering. Marcel Dekker, New York.

Sontag, E. D. (1998). *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, volume 6. Springer Science & Business Media.

Spong, M. W., Hutchinson, S., and Vidyasagar, M. (2020). *Robot Dynamics and Control*. John Wiley & Sons.

Sreekumar, C. and Agarwal, V. (2008). A Hybrid Control Algorithm for Voltage Regulation in DC–DC Boost Converter. *IEEE Transactions on Industrial Electronics*, 55(6):2530–2538.

Sreenath, K., Michael, N., and Kumar, V. (2013). Trajectory generation and control of a quadrotor with a cable-suspended load - A differentially-flat hybrid system. In *2013 IEEE International Conference on Robotics and Automation*, pages 4888–4895.

Stumper, J.-F., Hagenmeyer, V., Kuehl, S., and Kennel, R. (2015). Deadbeat Control for Electrical Drives: A Robust and Performant Design Based on Differential Flatness. *IEEE Transactions on Power Electronics*, 30(8):4585–4596.

Suryawan, F., De Doná, J., and Seron, M. (2012). Splines and polynomial tools for flatness-based constrained motion planning. *International Journal of Systems Science*, 43(8):1396–1411.

Susuki, Y., Koo, T. J., Ebina, H., Yamazaki, T., Ochi, T., Uemura, T., and Hikihara, T. (2012). A Hybrid System Approach to the Analysis and Design of Power Grid Dynamic Performance. *Proceedings of the IEEE*, 100(1):225–239.

Taheri, B. and Molzahn, D. K. (2025). AC-Informed DC Optimal Transmission Switching Problems via Parameter Optimization. *IEEE Transactions on Power Systems*, pages 1–12.

Tarjan, R. (1972). Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160.

Tomlin, C., Lygeros, J., and Shankar Sastry, S. (2000). A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970.

Torrisi, F. and Bemporad, A. (2004). HYSDEL—A Tool for Generating Computational Hybrid Models for Analysis and Synthesis Problems. *IEEE Transactions on Control Systems Technology*, 12(2):235–249.

Trentelman, H. (2004). On flat systems behaviors and observable image representations. *Systems & Control Letters*, 51(1):51–55.

Treuer, M., Weissbach, T., and Hagenmeyer, V. (2011). Flatness-Based Feedforward in a Two-Degree-of-Freedom Control of a Pumped Storage Power Plant. *IEEE Transactions on Control Systems Technology*, 19(6):1540–1548.

van Schuppen, J. H. (1998). A sufficient condition for controllability of a class of hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 374–383. Springer.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Willems, J. (1991). Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on Automatic Control*, 36(3):259–294.

Willems, J. C. (2007). The Behavioral Approach to Open and Interconnected Systems. *IEEE Control Systems Magazine*, 27(6):46–99.

Witsenhausen, H. (1966). A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions on Automatic Control*, 11(2):161–167.

Xu, J. and Li, Z. (2008). A Kinematic Model of Finger Gaits by Multifingered Hand as Hybrid Automaton. *IEEE Transactions on Automation Science and Engineering*, 5(3):467–479.

Zahn, F. and Hagenmeyer, V. (2022). Modelling power systems as flat hybrid automata for controlled line switching. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, pages 302–306, Virtual Event. ACM.

Zahn, F., Kleinert, T., and Hagenmeyer, V. (2022). Assessing the combination of differential flatness and deterministic automata for controllable hybrid systems. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2612–2619.

Zahn, F., Kleinert, T., and Hagenmeyer, V. (2024). Feedforward control of flat hybrid automata: A behavioral systems theory approach. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 6238–6244, Milan, Italy. IEEE.