Data-driven abstractions for the safe learning of nonlinear systems

Anas Makdesi

université PARIS-SACLAY

Data-driven abstractions for the safe learning of nonlinear systems

Abstractions basées sur les données pour l'apprentissage sécurisé de systèmes non linéaires

Thèse de doctorat de l'université Paris-Saclay

École doctorale n$^{(580)}$ Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Automatique
Graduate School : Informatique et sciences du numérique (ISN).
Référent : CentraleSupélec

Thèse préparée dans la unité de recherche **Laboratoire des signaux et systèmes (Université Paris-Saclay, CNRS, CentraleSupélec)**, sous la direction de **Antoine GIRARD**, directeur de recherche, et la co-direction de **Laurent FRIBOURG**, directeur de recherche.

Thèse soutenue à Paris-Saclay, le 07 Decembre 2023, par

Anas Makdesi

Composition du jury

Membres du jury avec voix délibérative

Sorin OLARU Professeur, CentraleSupélec	Président
Manuel Jr. MAZO Professeur associé, Technische Universiteit Delft	Rapporteur
Nacim RAMDANI Professeur des universités, Université d'Orléans	Rapporteur
Sadegh SOUDJANI Associate Professor, Newcastle University	Examinateur
Sylvie PUTOT Professeur, Ecole Polytechnique - Institut Polytechnique de Paris	Examinatrice

THESE DE DOCTORAT
NNT : 2023UPASG103

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Abstractions basées sur les données pour l'apprentissage sécurisé de systèmes non linéaires
Mots clés: À BASE DE DONNÉES, CONTRÔLE SYMBOLIQUE, SYSTÈMES NON LINÉAIRE

Résumé: Les méthodes de contrôle traditionnelles pour les systèmes non linéaires reposent fortement sur des modèles mathématiques précis, qui peuvent être difficiles, voire impossibles à obtenir dans certaines applications réelles. Pour résoudre ce problème, des techniques de contrôle basées sur les données ont émergé comme des alternatives prometteuses, utilisant des données d'entrée-sortie pour apprendre des politiques de contrôle directement à partir du comportement du système. Cependant, la sûreté reste une préoccupation essentielle dans le contrôle basé sur les données, car des modèles erronés peuvent avoir des conséquences catastrophiques. Ce travail propose une nouvelle approche basée sur les données pour contrôler les systèmes non linéaires, en mettant l'accent sur la sûreté pendant le processus d'apprentissage.

L'approche proposée utilise des sur-approximations de la dynamique du système, qui fournissent des représentations conservatives mais sûres du comportement du système. Ces sur-approximations sont apprises à partir de données d'entrée-sortie et sont ensuite utilisées pour construire des abstractions à états finis, qui capturent la dynamique essentielle du système sous une forme compacte et analysable. Cette abstraction est ensuite utilisée pour la synthèse du contrôleur, garantissant que le contrôleur maintient les propriétés et spécifications désirées tout au long du fonctionnement du système.

Une approche à deux modèles est introduite, où des modèles distincts sont construits pour la vérification de la sûreté et l'optimisation des performances. Le modèle de vérification de la sûreté est utilisé pour garantir que le contrôleur appris adhère aux contraintes de sûreté, tandis que le modèle d'optimisation des performances se concentre sur l'obtention des mesures de performance souhaitées. Cette séparation des préoccupations garantit que la sécurité est priorisée sans compromettre les performances.

L'approche proposée est rigoureusement analysée pour une large classe de systèmes non linéaires, y compris les systèmes monotones et les systèmes avec des fonctions dérivées bornées. Ces analyses fournissent des garanties théoriques pour la sûreté des contrôleurs appris et établissent la robustesse de la méthodologie de conception proposée.

L'efficacité des méthodes proposées est démontrée par une validation expérimentale approfondie sur divers systèmes non linéaires réels, notamment le contrôle de vitesse, la planification de trajectoires et les systèmes chaotiques (système de Lorenz). Ces expériences démontrent systématiquement la capacité de l'approche proposée à atteindre des performances satisfaisantes tout en maintenant une stricte adhésion aux contraintes de sûreté.

Title: Data-driven abstractions for the safe learning of nonlinear systems
Keywords: DATA-DRIVEN, SYMBOLIC CONTROL, NONLINEAR SYSTEMS

Abstract: Traditional control methods for nonlinear systems rely heavily on accurate mathematical models, which can be challenging or even impossible to obtain in some real-world applications. To address this issue, data-driven control techniques have emerged as promising alternatives, utilizing input-output data to learn control policies directly from system behavior. However, safety remains a critical concern in data-driven control, as erroneous models can lead to catastrophic consequences. This work proposes a novel data-driven approach to controlling nonlinear systems, with some emphasis on safety during the learning process.

The proposed approach utilizes over-approximations of system dynamics, which provide conservative yet safe representations of the system's behavior. These over-approximations are learned from input-output data and subsequently employed to construct finite-state abstractions, which capture the essential dynamics of the system in a compact and analyzable form. This abstraction is then utilized for controller synthesis, ensuring that the controller maintains the desired properties and specifications throughout the system's operation.

A two-model approach is introduced, where separate models are constructed for safety verification and performance optimization. The safety verification model is employed to guarantee that the learned controller adheres to safety constraints, while the performance optimization model focuses on achieving desired performance metrics. This separation of concerns ensures that safety is prioritized without compromising performance.

The proposed approach is rigorously analyzed for a broad class of nonlinear systems, including monotonic systems and systems with bounded derivative functions. These analyses provide theoretical guarantees for the safety of the learned controllers and establish the soundness of the proposed design methodology.

The effectiveness of the proposed methods is demonstrated through extensive experimental validation on various real-world nonlinear systems, including cruise control, trajectory planning, and chaotic systems (Lorenz system). These experiments consistently demonstrate the ability of the proposed approach to achieve satisfactory performance while maintaining strict adherence to safety constraints.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Antoine Girard and Laurent Fribourg, for their exceptional guidance, unwavering support, and invaluable insights throughout my research journey. Their expertise and dedication have been instrumental in shaping this thesis. Antoine Girard has been a constant source of inspiration, both in research and in life. Through my journey thus far, I have learned how hard it is to be a good mentor, and he has been an exceptional one. Saying that I am grateful for his mentorship would be an understatement. I am truly fortunate to have had the opportunity to work under his supervision. Laurent Fribourg's wisdom and guidance have been invaluable, and I am grateful for his mentorship and support. I am also grateful to him for his patience, kindness, and understanding throughout this journey.

I would also like to thank the members of the jury for their time and effort in evaluating this work. Their feedback and constructive criticism have greatly contributed to its improvement.

Next, I would like to express my gratitude to Sadegh Soudjani for providing me with the opportunity to visit his team at Newcastle University. His insights and feedback have been invaluable in shaping this research. I highly enjoyed the discussions and the collaboration with him and his team.

I am grateful to the members of the L2S laboratory for their support and encouragement. In particular, I would like to thank my colleagues for their friendship and camaraderie. Adel, Renato, Alissa, Dora, Georges, and many others have made my time at L2S truly memorable. Diego, thank you for your friendship and for always being there to help me when I needed it. Adnan and Elena, thank you for all the discussions, and for all the advice you have given me. You were the best senpai one could ask for. Finally, Jeanne and Fabian, thank you for being the best co-representatives ever.

I am grateful to my friends for their unwavering support and encouragement. Their belief in me has been a constant source of motivation. Seeing you all coming together to support me has been truly heartwarming, and I am grateful for your friendship. I genuinely believe that your friendship is the most valuable thing I have. Ragheed (without whom I would not have become the person I am today), Ayat, Jalal (who time and time again has shown me the true meaning of friendship), Ali Assi, M. Mhalla (Sada), Adam, Ghazal, Yazan (my flatmate and friend how helped me overcome the hardest times), Saeed, Fadi, Ghadeer, Hazem, Marah, Issa, Subhy, Sokrat, Ali Al-Abbas (The partner), Anaïs, Mouayd, Cécile, Anna, M. Darious, Tarek, and many others, thank you for your friendship and support.

Finally, I would like to express my deepest gratitude to my family for the endless love and support they have given me. Tony and Carol, I love you more than words can express. My father Philip, you seeded the love for science in me, and I am

forever grateful for that. Your kindness, wisdom, and love have been my guiding light, and I am eternally grateful for that. My mother Georget, your love and devotion have been my rock. You have always been there for me when I needed you, and I am truly blessed to have you as my mother.

Contents

List of Figures

1 - Introduction

1.1 . Literature review

1.1.1 . Safe learning

The integration of control theory and machine learning has ushered in an era of remarkable advances in autonomous systems. These systems are no longer confined to static rules or predefined behaviors but can adapt and learn from their experiences, thereby achieving unprecedented levels of autonomy and flexibility. Whether it's autonomous vehicles navigating complex road networks [12], industrial robots optimizing their tasks [47], or even humanoid robots learning to walk [68], learning-based control strategies have proven their mettle.

This was fueled by the recent advancements in data acquisition tools coupled with the ability to efficiently deal with the ever-increasing amount of data harnessed by those tools, which in turn gave rise to many new data-driven approaches and techniques in control theory. Those approaches have the benefit of being able to tackle unknown or hard-to-model systems, which justifies the increasing interest in them.

While some methods rely on finding the controller directly from the data [35, and references therein], others depend on finding data-driven models, which then can be used to find the controllers. A survey of both methods can be found in [15]. Learning the dynamics to find systems' models leverages the ability to use well-established, well-studied approaches to find controllers enforcing the desired behaviors. Moreover, data-driven models can sometimes come with formal guarantees regarding the approximation of the real system. Such guaranteed models can be categorized as stochastic or robust [34, and references therein]. The present work follows the latter approach, where the aim is to find bounds on the function representing the system (non-parametric approach) or bounds on some unknown parameters of the function representing the system (parametric approach).

With the rise of learning-based control strategies, a new set of challenges has emerged, particularly in the context of safety-critical applications [27, 5]. The reliability, predictability, and safety of autonomous systems are paramount in fields such as healthcare, aerospace, and autonomous driving, where human lives and valuable assets are at stake.

The ever-growing field of safe learning in control theory has emerged as a response to these concerns and challenges. Safe learning seeks to establish a principled framework for deploying learning algorithms while ensuring that the system's behavior remains within safe bounds, even when confronted with uncertainties and unforeseen conditions. The field of safe learning is still in its infancy, but it has captured the attention of many researchers, with many trying to tackle the challenges of safe learning in various ways [14, 24, 50, 61, 39, 73] from barrier functions

to reinforcement learning. The previous list is by no means exhaustive, and the literature on safe learning is growing rapidly. In what follows, we will try to explain the particular tools and methods used in this thesis to better understand the contribution of this work to the field of safe learning.

1.1.2 . Symbolic control

Symbolic control is a controller synthesis approach that uses abstraction to design provably correct controllers for complex systems. It is particularly well-suited for safety-critical systems, where it is important to be able to guarantee the correctness of the controller.

Abstraction is a process of simplifying a complex system by focusing on the aspects that are relevant to a particular task [71, 9]. In symbolic control, we use abstraction to construct a finite-state representation of the original system. This finite-state abstraction is characterized by a finite number of states and inputs. The transitions between states are determined by the inputs and the dynamics of the original system.

Once we have constructed a finite-state abstraction of the system, we can synthesize a controller for the abstraction using automata-theoretic techniques, such as game theory or model checking. The controller synthesized for the abstraction can then be refined to obtain a controller for the original system.

Symbolic control has been used to design controllers for a variety of safety-critical systems, including medical devices [13], aircraft [49], and nuclear power plants [16]. It has also been used to design controllers for systems with complex dynamics, such as robotic systems and hybrid systems.

Consider a simple example of a self-driving car. The car needs to be controlled to stay within a lane and avoid obstacles. The car can be modeled as a dynamical system with a continuous state space. The state of the system includes the car's position, velocity, and acceleration. The inputs to the system are the car's steering wheel angle and throttle.

Designing a controller for this system is a challenging task. The controller needs to be able to handle a variety of different scenarios, such as curves, obstacles, and other vehicles. It is also important to ensure that the controller is safe, i.e., that it will not cause the car to crash.

Symbolic control can be used to design a provably correct controller for the self-driving car. The first step is to construct a finite-state abstraction of the car. This can be done by discretizing the car's state space and inputs. For example, we could divide the car's position into three regions: left, center, and right. We could also discretize the car's velocity and acceleration into a few different levels.

Once we have constructed a finite-state abstraction of the car, we can synthesize a controller for the abstraction using automata-theoretic techniques. The controller will specify which inputs to apply to the car in each state in order to achieve the desired goal, such as staying within the lane and avoiding obstacles.

Finally, we can refine the controller for the finite-state abstraction to obtain a

controller for the original car. This refinement step is necessary to account for the error introduced by the abstraction.

The resulting controller will be provably correct, meaning that it will guarantee that the car stays within the lane and avoids obstacles, regardless of the initial conditions and the disturbances that the car encounters.

The range of systems that can be modeled using symbolic control is very broad. It includes linear systems such as [74, 65], piecewise affine systems [76], and nonlinear systems [71, 64, 77],

The range of specifications that can be handled by symbolic control is also very broad. It includes reach-avoid specifications [71, 29, 64, 77], safety specifications [71, 29], and full LTL [72].

Symbolic control, typically regarded as a model-driven technique, has recently witnessed the emergence of data-driven approaches. These innovative strategies present alternative avenues for addressing control challenges. For instance, Meyer and Girard introduced methods in [56, 32] that rely on system dynamics sampling on a predetermined grid. Similarly, the work in [77] aligns with this paradigm. Furthermore, the integration of data-driven abstractions into the probably approximately correct (PAC) statistical framework is explored in [25], highlighting the adaptability of symbolic control to modern statistical principles. PAC guarantees are also leveraged in [22] to facilitate data-driven abstractions using the scenario approach. This approach is further examined in [46] and [41], where it provides probabilistic guarantees for data-driven abstractions. A distinct focus on data-driven control design with regular language specifications is evident in [63], particularly for plants described as abstract systems. Additionally, for handling unknown aspects of nonlinear systems, Gaussian processes are harnessed as modeling tools. Subsequently, these models play a pivotal role in constructing symbolic abstractions, as demonstrated in [33].

These emerging data-driven approaches within the realm of symbolic control introduce an exciting dimension to the field, offering new perspectives and strategies for addressing complex control challenges.

1.1.3 . Over-approximation and reachability analysis

Reachability analysis is a technique for analyzing the behavior of a dynamical system. It is used to determine whether a system can reach a given set of states from a given set of initial states. Reachability analysis is used in a variety of applications, including control theory, robotics, and computer science [3]. This is vital for the calculation of the finite-state abstraction of a system in symbolic control. As we saw, to find the finite-state abstraction of a system, we discretize the state space and inputs of the system. Then, we use reachability analysis to determine which states can be reached from the initial states under the given inputs. Finding the reachable set of a system is a challenging task, especially for nonlinear systems. Instead, we can try to find an over-approximation of the reachable set. An over-approximation is a conservative approximation of a system's

behavior. In Chapter 2 we give a formal definition of over-approximation in the context of maps that describe the dynamics of a system. To review the literature on over-approximation, we can mention the following works. To over-approximate the reachable set of a linear system, we can use polytopes [21], ellipsoids [45], zonotopes [28], or support functions [31]. To over-approximate the reachable set of a nonlinear system, we can use Hamilton-Jacobi-Isaacs equations [58], Taylor models [20], or interval arithmetics [37]. A comprehensive overview of interval-based reachability analysis techniques can be found in [54].

In recent years, there has been a growing interest in using data-driven approaches to find over-approximations of the reachable set of a system. In [1], algorithms for computing over-approximated reachable sets based on matrix zonotopes are proposed. Data-driven reachability and support estimation with Christoffel functions is studied in [26]. Finally, we mention Set Membership Estimation (SME) [60], which is a data-driven approach to find over-approximations of the reachable set of a system. This method in particular is discussed in more detail in Chapter 3. Basically, in the set membership approach, we rely on the Libshitz constant of the system to find the least conservative over-approximation of the reachable set of the system from data.

1.2 . Motivation and Contribution

A major question of this thesis is: How can we learn models of a system from data that is suitable for controller synthesis and safe to implement? To answer this question, the main contribution of this thesis can be summarized as follows:

- Learning tight over-approximations of monotone systems from data.
- Learning over-approximations of bounded derivative systems from data.
- Using the learned over-approximations to build finite-state abstractions of the systems, which can be used to synthesize controllers.
- Introduce a two-model approach to safe learning. The first model is a discrete model that is used to synthesize a safety controller. Thus, ensuring that the learned system is safe. The second model is a compatible estimation of the dynamics, which is used to achieve desired performance.

We presented a method to learn data-driven over-approximation of monotone maps from input-output data. We showed that the resulting piecewise interval-valued monotone map is the tightest map that is consistent with the data (the notion of tightness and consistency will be properly defined later) [51]. We presented a method to efficiently learn an over-approximation map on a predefined grid of the input space [52], and showed that the resulting map is the tightest interval-valued map on this predefined grid. Then, we presented a method to

learn data-driven over-approximation of maps described as the sum of a bounded derivative function and a disturbance term sampled from a bounded set [53]. In the case of bounded derivative functions, we presented several techniques to compute such an over-approximation such as transforming the data into a monotone case or finding growth cones that over-approximate the map, and using them to learn the over-approximating map. We also discussed how to update the over-approximation locally when new data points are added. We show how to use the learned maps to build finite-state abstractions of the systems, which can be used to synthesize controllers. By introducing the concept of compatible estimation with data-driven over-approximation, we presented a two-model approach to safe learning. The first model is the discrete model that is used to synthesize a safety controller. Thus, ensuring that the learned system is safe. The second model is a compatible estimation of the dynamics, which is used to achieve desired performance. We presented a method to learn a compatible estimation of the dynamics of the system from data. We showed that the resulting estimation is a piecewise multi-affine estimation of the dynamics of the system. We presented a method to update the estimation locally when new data points are added. We presented a method to update the over-approximation locally when new data points are added. We showed how to use the methods of updating the over-approximation and the compatible estimation locally to update the models of the system online when new data points are added. We showed the effectiveness of the proposed methods by studying several examples relating to path planning and obstacle avoidance.

In the case of bounded derivative functions, we presented various techniques to compute an over-approximation. These techniques include transforming the data into a monotone case, or finding growth cones that over-approximate the map, and using them to learn a unified over-approximating map. We also discussed how to update the over-approximation locally when new data points are added.

To build finite-state abstractions of the systems and synthesize controllers, we utilized the learned maps. We introduced the concept of compatible estimation with data-driven over-approximation, which enabled a two-model approach to safe learning. The first model, a discrete model, was used to synthesize a safety controller, ensuring the learned system's safety. The second model, a compatible estimation of the dynamics, was employed to achieve desired performance.

We presented a method to learn a compatible estimation of the system dynamics from data. The resulting estimation was shown to be a piecewise multi-affine estimation. Additionally, we demonstrated how to update the estimation and over-approximation locally when new data points are added. These methods allowed for online model updates as new data points became available.

The effectiveness of the proposed methods was validated through the study of various examples related to path planning and obstacle avoidance.

Our research contributes to the field of data-driven control by providing a novel and rigorous approach to learning discrete models of monotone systems

from data. Our method can handle uncertainty and disturbances in the data and preserve the monotonicity property of the system. Our method can also be applied to systems with bounded derivative functions, which are common in many engineering applications. Our method enables controller synthesis for systems that are otherwise difficult or impossible to model analytically.

1.3 . Thesis outline

1.3.1 . Data-driven over-approximation of monotone systems

In this chapter, we present a data-driven over-approximation method for monotone systems. We first introduce a definition of monotone maps used to study monotone systems. We focus on studying how to learn monotone maps in general. The application of learning monotone maps to monotone dynamical systems is presented in Chapter 4.

We formulate the problem of learning a monotone map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$ of the form

$$\forall \mathbf{z} \in Z,\ F(\mathbf{z}) = f(\mathbf{z}) + D, \tag{1.1}$$

where $Z \subseteq \overline{\mathbb{R}}^n$, $f : Z \to \overline{\mathbb{R}}^m$ is a monotone function and $D = [\underline{\mathbf{d}}, \overline{\mathbf{d}}]$, $D \subseteq \mathbb{R}^m$ is a bounded interval of disturbances. The map F is learned from a finite set of input-output data generated by the map F.

$$\mathcal{D} = \{(\mathbf{z}_k, \mathbf{y}_k) \mid \mathbf{y}_k \in F(\mathbf{z}_k), k \in \mathbb{K}\},$$

where $\mathbb{K}$ is a finite set of indices. We call any map of the form (1.1) capable of generating the data, *consistent* with the $\mathcal{D}$, and we call any map containing or over-approximating all the consistent maps with the data, a *simulating* map. We define the problem of learning a monotone map by finding the tightest simulating map. We give a definition of the tightness of a simulating map (Definition 7) and provide a method to learn such a map in Theorem 1. The resulting map is piecewise interval-valued and monotone. Due to the high computational cost of calculating such a map and storing it, we propose a method to learn a map that is not necessarily tight but is still a simulating map on a predefined grid of the input space (Equation (2.14)). Moreover, we show that the resulting map is the tightest map on this predefined grid in Theorem 2. Finally, we present a method to learn this tight map on the grid efficiently (Proposition 5 and 6). We show the effectiveness of the proposed method by studying the learning of a monotone map and the effects of the number of data points and the size of partitions on the tightness of the resulting map, and on the computational cost of learning the map.

1.3.2 . Data-driven over-approximation of systems with bounded derivative functions

In this chapter, we study the problem of learning a map of the form 1.1 but instead of assuming that the function f is monotone, we assume that it is an un-

known bounded derivative function. The bounds of the derivatives of the function f and the interval D are known.

In this case, finding the tightest over-approximation of the map F is computationally intractable. Such a tight over-approximation can be found using the method developed in the set membership approach [60], we present this method (Theorem 8) and its limitations in calculating an over-approximation of the map F efficiently, especially with a large number of data points. Instead, we propose several methods to learn a map that is not necessarily tight but is still over-approximating the map F. The first method depends on transforming the data points using the bounds of the derivative function f such that the resulting data points can be seen as generated by a monotone map. Then, we use the method presented in Chapter 2 to learn a monotone map that is consistent with the transformed data points. Several such transformations can be carried out, and the resulting maps can be combined and transformed back to form a map that is over-approximating the map F (Proposition 9). The second method depends on calculating growth cones that have apexes at the data points and are bounded by the derivatives of the function f. Those growth cones are defined in Definition 8 and in Proposition 7 we show that each of these growth cones over-approximates the map F. Finding the values of the growth cones on the vertices of the partition grid allows us to learn a map that is over-approximating the map F (Proposition 10). Similar to the way we present an efficient method to learn a monotone map in Chapter 2, we present an efficient method to calculate the value of the growth cones on the vertices of the partition grid (Proposition 11 and 12). Then we present a method to update the over-approximation locally when new data points are added (Proposition 13). In the final section, we investigate how to find the bounds on the derivatives of the function f and the interval D from the data points. The presented approach is based on the scenario approach [18] for robust convex optimization. Using the scenario approach allows us to provide probabilistic guarantees on the found bounds. In the presented two-step approach, we sample a subset of pairs from the data and write a convex optimization problem that finds the bounds on the derivatives of the function f and the interval D. In the second step, we only update the bounds on the disturbance interval D using a larger subset of the data points. The reason for this is that, while the convex optimization problem can be solved for a moderately large data set, finding the bounds on the disturbance interval D can be done efficiently using a very large number of data points. Hence, better probabilistic guarantees can be achieved. Finally, similar to the previous chapter, we show the effectiveness of the proposed method by studying the learning of a map with bounded derivative functions and the effects of the number of data points and the size of partitions on the tightness of the resulting map, and on the computational cost of learning the map.

1.3.3 . Data-driven abstraction

In this chapter, we present how to use the learned maps in the previous chapters to synthesize a controller for a dynamical system. We first present a definition of transition systems, the framework we use to model and study systems (definition 9). Then, we talk about the concept of system relations and how to use them to model the behavior of a system with a simpler system. If we want to use a simpler model to synthesize a controller for the original system, we need to ensure that a relation between the two systems exists. This relation is called *alternating simulation* (Definition 11), which implies that any controller synthesized for the simpler system can be refined to control the original system. After that, we present a definition of the abstraction of a transition system which is the simpler system that we use to synthesize a controller for the original system. It is possible to abstract a continuous-state system into a discrete-state one using a partition of the state and input spaces. opening the door for using discrete controller synthesis to find controllers for a variety of complex specifications like safety and reachability specifications, or specifications given by temporal logic. We explain in particular how to find safety and reachability controllers for a discrete-state system. Finally, after defining all the required concepts, we show how we can use the learned maps in the previous chapters to find an abstraction of the interval-valued over-approximation of unknown systems using a set of transitions data generated by the system (Theorem 3). We study a general case where part of the system is known and part is unknown. We describe how to find over-approximations of the system and how to use those over-approximations to find an abstraction of the system. we prove that the resulting over-approximation is alternatingly simulating the original system. Also, we show that the resulting abstraction is alternatingly simulating the over-approximation and hence the original system. In the case where the system is completely unknown, we show that the resulting abstraction is alternatingly bisimulating the over-approximation. This means that working with the resulting abstraction is equivalent to working with the over-approximation and provides no more conservatism. Finally, we show the effectiveness of the proposed method by studying several examples. The first example is a simple example of a system with a known part and an unknown monotone part, which is a cruise control system. We want to find a controller that ensures that the velocity of a follower car remains within a safe interval of velocity and distance from the leader car. The second example is a path-planning problem for a car-like robot (unicycle model). We want to find a controller that ensures that the robot reaches a goal region while avoiding obstacles. In this chapter, we focus on the obstacle avoidance part of the problem (safety controller). The problem of choosing the best control action to reach the goal region is studied in Chapter 5. The third example is the chaotic Lorenz system. We start by implementing the approach presented in the previous chapters to find the bounds on the derivatives of the system and the disturbance interval. Then, we use the learned map to find an abstraction of the system and synthesize a safety controller for the system.

1.3.4 . Safe learning

The final chapter discusses the problem of safe learning, which is how to learn a model of a system from data while ensuring safety and performance. The chapter proposes a method that uses two models to achieve this goal. The first model is a discrete model that is used to synthesize a safety controller. Thus, ensuring that the learned system is safe. The second model is a compatible estimation of the dynamics, which is used to achieve desired performance. In previous chapters, we showed how to learn a discrete model of a system from data and how to use this model to synthesize a safety controller for the system. In this chapter, we start by presenting the framework that uses the two models to achieve safe learning. Theorem 4 presents the learning-based MPC scheme based on the two models. Then, we present a method to learn a compatible estimation of the dynamics of the system from data. This estimation is a piecewise multi-affine estimation of the dynamics of the system. In Section 5.2, we present what is a (piecewise) multi-affine function, its properties on interval domains, and how to learn such a function from data. Then, we use this class of functions to learn a compatible estimation of the dynamics of the system (Proposition 5.7). We then move to study how to update the estimation when new data points are added. In order to do that, we present a method to update the estimation locally (Proposition 19). Section 5.4 deals with how to use the methods of updating the over-approximation and the compatible estimation locally to update the models of the system online when new data points are added. Finally, we show the effectiveness of the proposed method by studying several examples relating to path planning and obstacle avoidance.

1.3.5 . Conclusion

The final chapter of the thesis provides a summary of the findings and suggests potential avenues for future research.

1.4 . Publications

This work led to the following publications:

- **Makdesi, A., Girard, A., Fribourg, L.** (2021). "Data-driven abstraction of monotone systems." Learning for Dynamics and Control.
- **Makdesi, A., Girard, A., Fribourg, L.** (2021). "Efficient data-driven abstraction of monotone systems with disturbances." Conference on Analysis and Design of Hybrid Systems.
- **Makdesi, A., Girard, A., Fribourg, L.** (2023). "Safe Learning-Based Model Predictive Control Using The Compatible Models Approach." European Journal of Control.
- **Makdesi, A., Girard, A., Fribourg, L.** (2021). "Data-Driven Models of Monotone Systems." Submitted to the Transactions on Automatic Control.

2 - Data-Driven Over-approximation of Monotone Systems

In this chapter, we study the problem of learning models of monotone systems. Using a set of data generated by the system, we aim to learn a model that is an over-approximation of the unknown system that generated the data. An over-approximation is a set-valued map that contains the unknown system. This over-approximation will be then used to synthesize a controller for the system in later chapters. The quality of the controller depends on the tightness of the over-approximation. Therefore, we aim to learn the tightest over-approximation of the unknown system. We introduce the notion of minimal simulating maps, which are the tightest over-approximations of the unknown system. We provide a constructive proof of the existence of minimal simulating maps. Then, we propose a method for learning a minimal simulating map from data efficiently.

The class of monotone systems is the class of systems whose trajectories preserve a partial order on the state and input spaces. Among the classical references about monotone systems are the textbook by Smith [70] and the work in [6]. This class of systems is very rich and includes many systems of interest in practice, such as adaptive cruise control [36], temperature regulation systems [55] or power networks [79]. In [57], models that enforce a notion of monotonicity are learned following the argument that, for many machine learning problems, some inputs relate to the output monotonically, such as house pricing. Data-driven approaches are used in [40] for monotone systems reduction. In this chapter, we are interested in how to use the monotonicity property to learn over-approximations of monotone systems from data. Existing approaches for data-driven over-approximation require assumptions similar to those considered in Chapter 3, namely that the system is Libschitz continuous or has bounded derivatives. The discussion about those approaches is postponed to Section 3.1.

The chapter is organized as follows. In Section 2.1, we introduce the notion of monotone systems. In Section 2.2, we introduce the notion of minimal simulating maps and provide proof of the existence of minimal simulating maps. In Section 2.3, we propose a method for learning a minimal simulating map from data efficiently. In Section 2.5, we illustrate the proposed method on a numerical example. Finally, in Section 2.6, we conclude the chapter and discuss future work.

2.1 . Monotone maps/systems

In this section, we introduce the notion of monotone maps and systems. We start by introducing the notion of partial order, which is the mathematical structure used to define monotonicity. Then, we introduce the notion of monotone maps

and systems.

2.1.1 . Partial Order

A binary relation $\preceq_{\mathcal{L}} \subseteq \mathcal{L} \times \mathcal{L}$ is a partial order if and only if for all $l_1, l_2, l_3 \in \mathcal{L}$ we have:

- $l_1 \preceq_{\mathcal{L}} l_1$ (reflexivity);
- If $l_1 \preceq_{\mathcal{L}} l_2$ and $l_2 \preceq_{\mathcal{L}} l_1$ then $l_1 = l_2$ (antisymmetry);
- If $l_1 \preceq_{\mathcal{L}} l_2$ and $l_2 \preceq_{\mathcal{L}} l_3$ then $l_1 \preceq_{\mathcal{L}} l_3$ (transitivity).

We denote by $\preceq_{\mathcal{L}}$ the partial order on $\mathcal{L}$. If neither $l_1 \preceq_{\mathcal{L}} l_2$ nor $l_2 \preceq_{\mathcal{L}} l_1$ then we write $l_1 \not\preceq_{\mathcal{L}} l_2$, and we say that l_1 and l_2 are incomparable.

We say that $l_1 \prec_{\mathcal{L}} l_2$ if $l_1 \preceq_{\mathcal{L}} l_2$ and $l_1 \neq l_2$. We define $\succeq_{\mathcal{L}}$ and $\succ_{\mathcal{L}}$ similarly. Given a partially ordered set $\mathcal{L}$, for $l_1, l_2 \in \mathcal{L}$, we define the interval $[l_1, l_2]$ as the set $\{l \in \mathcal{L} \mid l_1 \preceq_{\mathcal{L}} l \preceq_{\mathcal{L}} l_2\}$. For $a \in \mathcal{L}$, we define the lower and upper closure of a as $\downarrow a = \{l \in \mathcal{L} \mid l \preceq_{\mathcal{L}} a\}$ and $\uparrow a = \{l \in \mathcal{L} \mid a \preceq_{\mathcal{L}} l\}$ respectively. For $A \subseteq \mathcal{L}$, we define the lower and upper closure of A as $\downarrow A = \bigcup_{a \in A} \downarrow a$ and $\uparrow A = \bigcup_{a \in A} \uparrow a$ respectively. A subset $A \subseteq \mathcal{L}$ is said to be lower-closed (respectively upper-closed) if $\downarrow A = A$ (respectively $\uparrow A = A$).
In this chapter, we will be interested in the case where $\mathcal{L} = \overline{\mathbb{R}}^n$ is the n-dimensional extended real space. With some abuse of notation, we will use the symbol $\preceq$ and drop the subscript $\mathcal{L}$. Given two vectors $\mathbf{z}_1, \mathbf{z}_2 \in \overline{\mathbb{R}}^n$, we define the partial order $\preceq$ on $\overline{\mathbb{R}}^n$ to be $\mathbf{z}_1 \preceq \mathbf{z}_2$ if and only if $z_1^i \leq z_2^i$ for all $i = 1, \ldots, n$. We use indifferently $\mathbf{z}_1 \preceq \mathbf{z}_2$ and $\mathbf{z}_2 \succeq \mathbf{z}_1$. We define $\max(\mathbf{z}_1, \mathbf{z}_2)$, or $\min(\mathbf{z}_1, \mathbf{z}_2)$, to be the vector $\mathbf{z}$ whose components are $z^i = \max(z_1^i, z_2^i)$, or $z^i = \min(z_1^i, z_2^i)$ respectively. $\inf Z$ and $\sup Z$ denote the *infimum* and the *supremum* of Z, i.e. the greatest lower and least upper bounds of Z relative to partial order $\preceq$ on $\overline{\mathbb{R}}^n$.

2.1.2 . Monotone systems definition

Continuous time

We consider a continuous-time system of the form

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \tag{2.1}$$

where $\mathbf{x} \in \overline{\mathbb{R}}^{n_x}$ is the state ($\overline{\mathbb{R}} = [-\infty, +\infty]$ is the set of *extended real numbers*), $\mathbf{u} \in \overline{\mathbb{R}}^{n_u}$ is the input, and $f : \overline{\mathbb{R}}^{n_x} \times \overline{\mathbb{R}}^{n_u} \to \overline{\mathbb{R}}^{n_x}$. We denote a control policy by $\tilde{\mathbf{u}} : \mathbb{R}_0^+ \to \overline{\mathbb{R}}^{n_u}$. We write $\tilde{\mathbf{u}} \preceq \tilde{\mathbf{u}}'$ if and only if $\tilde{\mathbf{u}}(t) \preceq \tilde{\mathbf{u}}'(t)$ for all $t \in \mathbb{R}_0^+$.

Definition 1. *System* (2.1) *is* monotone *if for all* $\mathbf{x}, \mathbf{x}' \in \overline{\mathbb{R}}^{n_x}$*, for all* $t \in \mathbb{R}_0^+$*,*

$$\mathbf{x} \preceq \mathbf{x}', \tilde{\mathbf{u}} \preceq \tilde{\mathbf{u}}' \Rightarrow \phi(\mathbf{x}, \tilde{\mathbf{u}}) \preceq \phi(\mathbf{x}', \tilde{\mathbf{u}}'),$$

where $\phi(\mathbf{x}, \tilde{\mathbf{u}})$ *is the solution of* (2.1) *with initial condition* $\mathbf{x}$ *and input* $\tilde{\mathbf{u}}$*.*

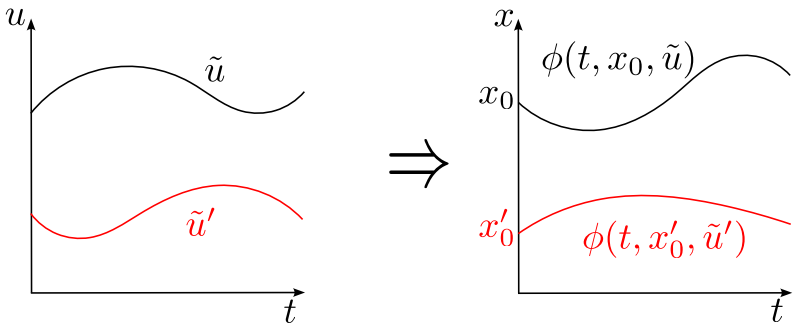


Figure 2.1: Example of a scalar monotone system.

Discrete time

Now let us consider the case of discrete-time systems of the form

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{2.2}$$

where $\mathbf{x}_k \in \overline{\mathbb{R}}^{n_x}$ is the state, $\mathbf{u}_k \in \overline{\mathbb{R}}^{n_u}$ is the input, and $f : \overline{\mathbb{R}}^{n_x} \times \overline{\mathbb{R}}^{n_u} \to \overline{\mathbb{R}}^{n_x}$. In this case, the monotonicity of the system is defined using the vector field f.

Definition 2. *System* (2.1) *is* monotone *if for all* $\mathbf{x}_k, \mathbf{x}_k' \in \overline{\mathbb{R}}^{n_x}$*, for all* $\mathbf{u}_k, \mathbf{u}_k' \in \overline{\mathbb{R}}^{n_u}$*,*

$$\mathbf{x}_k \preceq \mathbf{x}_k', \mathbf{u}_k \preceq \mathbf{u}_k' \Rightarrow f(\mathbf{x}_k, \mathbf{u}_k) \preceq f(\mathbf{x}_k', \mathbf{u}_k').$$

Therefore, for discrete-time systems, the study of the monotonicity of the vector field f is sufficient to determine the monotonicity of the system.
As all the systems studied in this work are written as discrete-time systems, we will first shift the study to monotone vector fields. Also, we are interested in studying systems more general than (2.2), where disturbances are taken into account. We consider the following system

$$\mathbf{x}_{k+1} \in f(\mathbf{x}_k, \mathbf{u}_k) + D, \tag{2.3}$$

where $D \subseteq \overline{\mathbb{R}}^{n_x}$ is a set describing the disturbance. Signaling the need to define the monotonicity of set-valued maps, which will be done in the next section.

2.1.3 . Monotone maps

To study systems of the form (2.3), we need to define the monotonicity of set-valued maps.

Definition 3. *The map* $F : Z \rightrightarrows Y$*, with* $Z \subseteq \overline{\mathbb{R}}^n$ *and* $Y \subseteq \overline{\mathbb{R}}^m$*, is monotone if for all* $\mathbf{z}, \mathbf{z}' \in Z$ *with* $\mathbf{z} \preceq \mathbf{z}'$*,*

$$\forall \mathbf{y} \in F(\mathbf{z}),\ \exists \mathbf{y}' \in F(\mathbf{z}'),\ \mathbf{y} \preceq \mathbf{y}',\ \textit{and}$$
$$\forall \mathbf{y}' \in F(\mathbf{z}'),\ \exists \mathbf{y} \in F(\mathbf{z}),\ \mathbf{y} \preceq \mathbf{y}'.$$

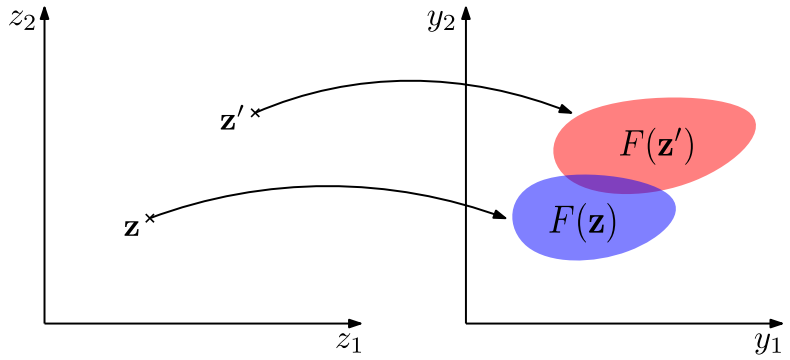


Figure 2.2: Example of a set-valued monotone map $F : Z \rightrightarrows Y$.

An illustration of the notion of monotone maps is shown in Figure 2.2. Let us remark that in the case of a function $f : Z \to Y$, Definition 3 coincides with the usual definition of monotone functions, that is

$$\forall \mathbf{z}, \mathbf{z}' \in Z,\ \mathbf{z} \preceq \mathbf{z}' \implies f(\mathbf{z}) \preceq f(\mathbf{z}').$$

We will focus on a class of maps given by monotone functions with additive bounded disturbances. Formally, let us consider a map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$ where $Z \subseteq \overline{\mathbb{R}}^n$ and such that

$$\forall \mathbf{z} \in Z,\ F(\mathbf{z}) = f(\mathbf{z}) + D, \tag{2.4}$$

where $f : Z \to \overline{\mathbb{R}}^m$ is a monotone function and $D = [\underline{\mathbf{d}}, \overline{\mathbf{d}}]$, $D \subseteq \mathbb{R}^m$ is a bounded interval of disturbances. The following property of F is a straightforward consequence of (2.4) and of Definition 3 and is therefore stated without proof.

Claim 1. *The map F given by (2.4), where f is a monotone function and $D \subseteq \mathbb{R}^m$, is a monotone map.*

Example 1

Let us consider the map $F : \overline{\mathbb{R}}^+ \rightrightarrows \overline{\mathbb{R}}$ given for all $z \in \overline{\mathbb{R}}^+$ by

$$F(z) = \left\{ y \in \overline{\mathbb{R}} \mid y \in z^2 + [-0.5, 0.5] \right\},$$

where $[-0.5, 0.5]$ can represent the interval of disturbances. Then, F is a monotone map.

For the rest of the chapter, we will focus on monotone maps of the form (2.4). hi

2.2 . Minimal simulating maps

In this section, We study how to learn models of monotone maps from data. The learned model is an over-approximation of the unknown map that generated the data. We search for the tightest over-approximations, which we call minimal simulating maps. We introduce those maps and provide constructive proof of their existence. First, let us formally define the notion of over-approximation.

Definition 4. *An over-approximation of a map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$ is a map $S : Z \rightrightarrows \overline{\mathbb{R}}^m$ such that for all $\mathbf{z} \in Z$, $F(\mathbf{z}) \subseteq S(\mathbf{z})$.*

Let us consider a map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$ of the form (2.4) where the monotone function $f : Z \to \overline{\mathbb{R}}^m$ is **unknown** and the disturbances lower and upper bounds $\underline{\mathbf{d}}$, $\overline{\mathbf{d}} \in \mathbb{R}^m$ are **known**. A set of data $\mathcal{D} \subseteq Z \times \overline{\mathbb{R}}^m$ generated using the map F:

$$\mathcal{D} = \{(\mathbf{z}_k, \mathbf{y}_k) \mid \mathbf{y}_k \in F(\mathbf{z}_k), k \in \mathbb{K}\}, \tag{2.5}$$

where $\mathbb{K}$ is a finite set of indices.
Given the data $\mathcal{D}$, the bounds $\underline{\mathbf{d}}$, $\overline{\mathbf{d}}$, and under the sole assumption that f is monotone, we aim at computing an over-approximation of the map F that is as "tight" as possible.

To formalize the notion of tightness, we introduce the following several notions. A map will be said to be consistent if it is of the form (2.4) and is capable of generating the data $\mathcal{D}$.

Definition 5. *A map $\tilde{F} : Z \rightrightarrows \overline{\mathbb{R}}^m$ is consistent with the data $\mathcal{D}$ if the following hold:*

1. *There exists a monotone function $\tilde{f} : Z \to \overline{\mathbb{R}}^m$ such that, for all $\mathbf{z} \in Z$, $\tilde{F}(\mathbf{z}) = \tilde{f}(\mathbf{z}) + D$;*
2. *For all $(\mathbf{z}, \mathbf{y}) \in \mathcal{D}$, $\mathbf{y} \in \tilde{F}(\mathbf{z})$.*

We denote the set of maps consistent with the data $\mathcal{D}$ by $\mathbb{C}_{\mathcal{D}}$.

Obviously, there is at least one map that is consistent with $\mathcal{D}$, which is the map F that generated $\mathcal{D}$. In general, there could be more.

Example 2

Let us consider the same map $F : \overline{\mathbb{R}}^+ \rightrightarrows \overline{\mathbb{R}}$ introduced in Example 2.1.3. Let us consider the data $\mathcal{D} = \{(1, 1.2), (2, 4.3), (3, 8.6)\}$ generated using the map F. Then, all the following maps are consistent with $\mathcal{D}$: For all $z \in \overline{\mathbb{R}}^+$

- The true map $F(z) = \{y \in \overline{\mathbb{R}} \mid y \in z^2 + [-0.5, 0.5]\}$,

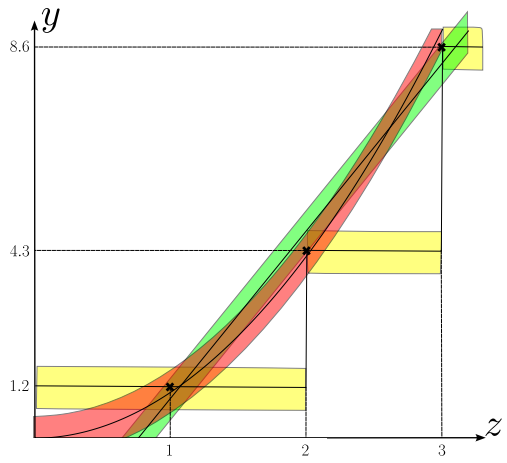


Figure 2.3: Map F in red. Map $\tilde{F}_1$ in yellow. Map $\tilde{F}_2$ in green.

- The piecewise-defined map built according to the data $\mathcal{D}$:

$$\tilde{F}_1(z) = \left\{ y \in \overline{\mathbb{R}} \mid y \in \begin{cases} 1.2 + [-0.5, 0.5] & \text{if } z \in [0, 2), \\ 4.3 + [-0.5, 0.5] & \text{if } z \in [2, 3), \\ 8.6 + [-0.5, 0.5] & \text{if } z \in [3, +\infty). \end{cases} \right\}$$

- The map built using linear regression

$$\tilde{F}_2(z) = \left\{ y \in \overline{\mathbb{R}} \mid y \in 3.7z - 2.7 + [-0.5, 0.5] \right\}$$

A map over-approximating all the consistent maps is called a simulating map.

Definition 6. *A map $S : Z \rightrightarrows \overline{\mathbb{R}}^m$ is a* simulating map *of the data $\mathcal{D}$ if for all $\tilde{F} \in \mathbb{C}_{\mathcal{D}}$, for all $\mathbf{z} \in Z$, $\tilde{F}(\mathbf{z}) \subseteq S(\mathbf{z})$. We denote the set of all simulating maps of $\mathcal{D}$ by $\mathbb{S}_{\mathcal{D}}$.*

Example 3

Let us consider the same data $\mathcal{D}$ as in Example 2.2. Then, the following maps are simulating maps of $\mathcal{D}$:

- A trivial and useless example of a simulating map is the map given for all $z \in \overline{\mathbb{R}}^+$ by $S_1(\mathbf{z}) = \overline{\mathbb{R}}$.
- The map

$$S_2(z) = \left\{ y \in \overline{\mathbb{R}} \mid y \in \begin{cases} [-\infty, 2.2] & \text{if } z \in [0, 1), \\ [0.2, 5.3] & \text{if } z \in [1, 2), \\ [3.3, 9.6] & \text{if } z \in [2, 3), \\ [7.6, \infty] & \text{if } z \in [3, +\infty). \end{cases} \right\},$$

The maps $\tilde{F}_1$ and $\tilde{F}_2$ introduced in Example 2.2 are not simulating maps of $\mathcal{D}$, because $\tilde{F}_1$ and $\tilde{F}_2$ are not an over-approximation of F (Do not include F). Also the true map F is not a simulating map of $\mathcal{D}$ because it does not over-approximate all the consistent maps (Example: there exists $z \in \overline{\mathbb{R}}^+$, such that $\tilde{F}_1(z) \nsubseteq F(z)$). The map S_2 is the tightest simulating map of $\mathcal{D}$, which will be apparent why shortly.

Out of the maybe infinite number of simulating maps, we are interested in the tightest one, which we call a minimal simulating map. Those maps are the least conservative over-approximations of the unknown map F that generated the data $\mathcal{D}$. Formally, they are defined as follows:

Definition 7. *A map $S_m : Z \rightrightarrows \overline{\mathbb{R}}^m$ is a minimal simulating map of the data $\mathcal{D}$ if the following hold:*

1. *$S_m \in \mathbb{S}_{\mathcal{D}}$ (Simulation);*
2. *For almost all $\mathbf{z} \in Z$, for all $S \in \mathbb{S}_{\mathcal{D}}$, $S_m(\mathbf{z}) \subseteq S(\mathbf{z})$ (Minimality).*

Remark 1. *Although minimality holds only almost everywhere, it is required that the simulation property holds for all $\mathbf{z} \in Z$. Therefore, a minimal simulating map is always an over-approximation of the unknown map F that generated the data $\mathcal{D}$. We require minimality only almost everywhere to make it possible to give a simple construction of the minimal simulating map as shown in subsection 2.2.*

In subsection 2.2, we will provide a constructive proof of the existence of minimal simulating maps. The following proposition describes the relation between minimal simulating maps.

Proposition 1. *Given two maps $S_m, S'_m : Z \rightrightarrows \overline{\mathbb{R}}^m$, the following properties hold:*

- *If S_m and S'_m are minimal simulating maps of $\mathcal{D}$, then $S_m = S'_m$ a.e.;*
- *If S_m is a minimal simulating map of $\mathcal{D}$, $S'_m \in \mathbb{S}_{\mathcal{D}}$ and $S_m = S'_m$ a.e., then S'_m is a minimal simulating map of $\mathcal{D}$.*

Proof. We prove the first part of the proposition. According to Definition 7, there exists a set $Z_0 \subseteq Z$ of measure zero such that, for all $\mathbf{z} \in Z \setminus Z_0$, $S_m(\mathbf{z}) \subseteq S'_m(\mathbf{z})$, and there exists a set $Z'_0 \subseteq Z$ of measure zero such that, for all $\mathbf{z} \in Z \setminus Z'_0$, $S'_m(\mathbf{z}) \subseteq S_m(\mathbf{z})$. Hence, for all $\mathbf{z} \in Z \setminus (Z_0 \cup Z'_0)$, $S'_m(\mathbf{z}) = S_m(\mathbf{z})$. The set $Z_0 \cup Z'_0$ being of measure zero, it follows that $S_m = S'_m$ a.e..

Now, let us show the second part of the proposition. For S'_m to be a minimal simulating map, S'_m should meet the two requirements of Definition 7.

The first one is satisfied by assumption. The second one can be shown as follows. There exists a set $Z_0 \subseteq Z$ of measure zero such that, for all $\mathbf{z} \in Z \setminus Z_0$, for all $S \in \mathbb{S}_{\mathcal{D}}$, $S_m(\mathbf{z}) \subseteq S(\mathbf{z})$. Also, there exists a set $Z_0' \subseteq Z$ of measure zero such that, for all $\mathbf{z} \in Z \setminus Z_0'$, $S_m'(\mathbf{z}) = S_m(\mathbf{z})$. Hence, for all $\mathbf{z} \in Z \setminus (Z_0 \cup Z_0')$, for all $S \in \mathbb{S}_{\mathcal{D}}$, $S_m'(\mathbf{z}) \subseteq S(\mathbf{z})$. Because $Z_0 \cup Z_0'$ is of measure zero, the second requirement of Definition 7 is satisfied, and S_m' is a minimal simulating map. □

Proposition 1 essentially states that minimal simulating maps are uniquely determined up to a set of measure zero. We can now formally state the problem under consideration in this section:

The following problem is the one we aim to solve in this section:

Problem 1. *Given the data $\mathcal{D}$ and the disturbance bounds $\underline{\mathbf{d}}, \overline{\mathbf{d}} \in \mathbb{R}^m$, compute $S_m : Z \rightrightarrows \overline{\mathbb{R}}^m$, a minimal simulating map of $\mathcal{D}$.*

Characterization and properties

In the following, we describe a solution to Problem 1. We first establish a characterization of a minimal simulating map and prove some of its properties. Then, a practical approach is provided for computing a minimal simulating map.

The following theorem provides an effective characterization of a minimal simulating map:

Theorem 1. *Let $S_m : Z \rightrightarrows \overline{\mathbb{R}}^m$ be the map given for all $\mathbf{z} \in Z$ by:*

$$\begin{aligned} S_m(\mathbf{z}) = &\left(\bigcap_{k \in \mathbb{K}^-(\mathbf{z})} \left\{ \mathbf{y} \in \overline{\mathbb{R}}^m \;\middle|\; \mathbf{y}_k + \underline{\mathbf{d}} - \overline{\mathbf{d}} \preceq \mathbf{y} \right\} \right) \\ &\cap \left(\bigcap_{k \in \mathbb{K}^+(\mathbf{z})} \left\{ \mathbf{y} \in \overline{\mathbb{R}}^m \;\middle|\; \mathbf{y} \preceq \mathbf{y}_k + \overline{\mathbf{d}} - \underline{\mathbf{d}} \right\} \right), \end{aligned} \tag{2.6}$$

where

$$\mathbb{K}^-(\mathbf{z}) = \{k \in \mathbb{K} \mid \mathbf{z}_k \preceq \mathbf{z}\},\ \mathbb{K}^+(\mathbf{z}) = \{k \in \mathbb{K} \mid \mathbf{z} \preceq \mathbf{z}_k\}. \tag{2.7}$$

Then, S_m is a minimal simulating map of $\mathcal{D}$.

Figure 2.4 shows the construction of the map S_m. The map S_2 in Example 2.2 offers an example of such a map in the scalar case. Let us remark that Theorem 1 also provides a constructive proof of the existence of minimal simulating maps. It follows from (2.6) that the minimal simulating map S_m is interval-valued. Indeed, for all $\mathbf{z} \in Z$, $S_m(\mathbf{z}) = [\underline{S}_m(\mathbf{z}), \overline{S}_m(\mathbf{z})]$ with upper and lower bounds

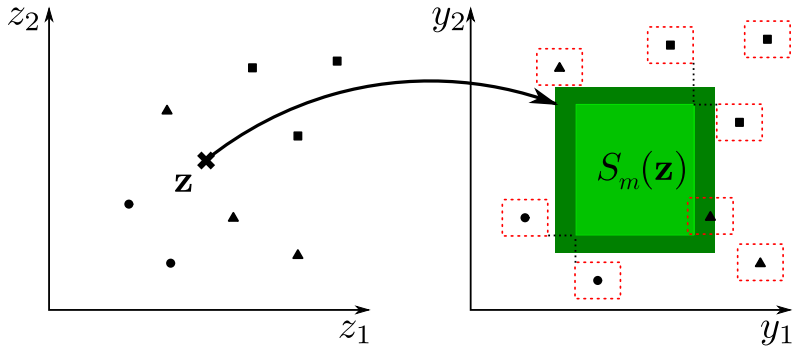


Figure 2.4: Construction of a minimal simulating map. Left: data points in the input space $z_k \in Z$, $k \in \mathbb{K}$, points larger than ($k \in \mathbb{K}^+(\mathbf{z})$) / lesser than ($k \in \mathbb{K}^-(\mathbf{z})$) / incomparable to a given $\mathbf{z} \in Z$ are represented with squares /circles /triangles, respectively. Right: data points in the output space $y_k \in \overline{\mathbb{R}}^m$, $k \in \mathbb{K}$ are surrounded by red intervals representing the bounded disturbance, and the green interval represents the minimal simulating map $S_m(\mathbf{z}) \subseteq \overline{\mathbb{R}}^m$ given by (2.6). The light green area represents the approximation of the function f as mentioned in Remark 2.

$\overline{S}_m(\mathbf{z})$, $\underline{S}_m(\mathbf{z})$ given by

$$\overline{S}_m(\mathbf{z}) = \inf \left\{ \mathbf{y}_k + \overline{\mathbf{d}} - \underline{\mathbf{d}} \mid k \in \mathbb{K}^+(\mathbf{z}) \right\}, \tag{2.8}$$

$$\underline{S}_m(\mathbf{z}) = \sup \left\{ \mathbf{y}_k + \underline{\mathbf{d}} - \overline{\mathbf{d}} \mid k \in \mathbb{K}^-(\mathbf{z}) \right\}. \tag{2.9}$$

We now prove Theorem 1.

Proof. Let us first show that $S_m \in \mathbb{S}_{\mathcal{D}}$. We should prove that the map S_m contains any consistent map, so let us consider $\tilde{F} \in \mathbb{C}_{\mathcal{D}}$, then there exists a monotone function $\tilde{f} : Z \to \overline{\mathbb{R}}^m$ such that, $\tilde{F}(\mathbf{z}) = \tilde{f}(\mathbf{z}) + D$. Let $\mathbf{z} \in Z$, $\mathbf{y} \in \tilde{F}(\mathbf{z})$, then, $\mathbf{y} \preceq \tilde{f}(\mathbf{z}) + \overline{\mathbf{d}}$. From the monotonicity of $\tilde{f}$, we have $\tilde{f}(\mathbf{z}) \preceq \tilde{f}(\mathbf{z}_k)$ for all $k \in \mathbb{K}^+(\mathbf{z})$. Therefore,

$$\mathbf{y} \preceq \tilde{f}(\mathbf{z}_k) + \overline{\mathbf{d}}, \ \forall k \in \mathbb{K}^+(\mathbf{z}).$$

Moreover, $\tilde{F} \in \mathbb{C}_{\mathcal{D}}$ implies $\mathbf{y}_k \in \tilde{F}(\mathbf{z}_k)$, for all $k \in \mathbb{K}$. Therefore, we have

$$\tilde{f}(\mathbf{z}_k) + \underline{\mathbf{d}} \preceq \mathbf{y}_k, \ \forall k \in \mathbb{K}.$$

Hence, we get from the inequalities above that

$$\mathbf{y} \preceq \mathbf{y}_k + \overline{\mathbf{d}} - \underline{\mathbf{d}}, \ \forall k \in \mathbb{K}^+(\mathbf{z}).$$

Then, from (2.8), $\mathbf{y} \preceq \overline{S}_m(\mathbf{z})$. Similarly, it can be shown that $\mathbf{y} \succeq \underline{S}_m(\mathbf{z})$. Therefore, we have $\mathbf{y} \in S_m(\mathbf{z})$; hence, S_m is a simulating map of $\mathcal{D}$.

Now, we prove minimality. For any arbitrary point included in S_m, we will build a consistent map containing this point, and show that this map is a subset of S_m. We will partition the input space Z into three regions, the first region contains all the data points smaller than the point of interest, the second region contains all the data points larger than the point of interest, the final region contains only the point of interest. Then, we will build a consistent map, which on the first region is the smallest possible, on the second region is the largest possible, and on the third region contains the point of interest, all while being included in S_m. But first, we remove the grid built by the data points, which is a set of measure zero. Consider the set Z_0 of measure zero, defined as

$$Z_0 = \{\mathbf{z} \in Z \mid \exists i \in \{1, \dots, n\}, k \in \mathbb{K}, z^i = z^i_k\}. \tag{2.10}$$

This set consists of the union of finitely many hyperplanes. Each of those hyperplanes is defined by one of the components of a data point. Let $\mathbf{z}^\star \in Z \setminus Z_0$ and $\mathbf{y}^\star \in S_m(\mathbf{z}^\star)$. This is allowed under our definition of minimal simulating maps. Let us define a partition of Z in 3 regions defined by

$$Z_1 = Z^+(\mathbf{z}^\star) \setminus Z^-(\mathbf{z}^\star),\ Z_2 = Z^+(\mathbf{z}^\star) \cap Z^-(\mathbf{z}^\star),$$
$$Z_3 = Z \setminus (Z_1 \cup Z_2),$$

where

$$Z^-(\mathbf{z}^\star) = \bigcap_{i=1}^{n} \left\{ \mathbf{z} \in Z \mid z^i > \sup\{z^i_k \mid z^i_k \leq z^{\star i}, k \in \mathbb{K}\} \right\},$$

$$Z^+(\mathbf{z}^\star) = \bigcap_{i=1}^{n} \left\{ \mathbf{z} \in Z \mid z^i < \inf\{z^i_k \mid z^i_k \geq z^{\star i}, k \in \mathbb{K}\} \right\}.$$

A representation of the regions can be found in Figure 2.5. Let us remark that by construction $\mathbf{z}_k \in Z_1 \cup Z_3$ for all $k \in \mathbb{K}$ and that $\mathbf{z}^\star \in Z_2$ because $\mathbf{z}^\star \in Z \setminus Z_0$.

Then, let us consider the map $F^\star : Z \rightrightarrows \overline{\mathbb{R}}^m$ given for all $\mathbf{z} \in Z$ by $F^\star(\mathbf{z}) = f^\star(\mathbf{z}) + W$ where $f^\star : Z \to \overline{\mathbb{R}}^m$ is defined as follows

$$f^\star(\mathbf{z}) = \begin{cases} \underline{S}_m(\mathbf{z}) - \underline{\mathbf{d}} & \text{if } \mathbf{z} \in Z_1, \\ \max(\min(\mathbf{y}^\star, \overline{S}_m(\mathbf{z}^\star) - \overline{\mathbf{d}}), \underline{S}_m(\mathbf{z}^\star) - \underline{\mathbf{d}}) & \text{if } \mathbf{z} \in Z_2, \\ \overline{S}_m(\mathbf{z}) - \overline{\mathbf{d}} & \text{if } \mathbf{z} \in Z_3. \end{cases} \tag{2.11}$$

In Figure 2.6, a representation, in the one-dimensional case, of the maps $F, F^\star$, and S_m can be found. Since $\mathbf{z}^\star \in Z_2$ and since $\mathbf{y}^\star \in S_m(\mathbf{z}^\star) = [\underline{S}_m(\mathbf{z}^\star), \overline{S}_m(\mathbf{z}^\star)]$, it follows from (2.11) that $\mathbf{y}^\star \in F^\star(\mathbf{z}^\star)$. We will now prove that $F^\star \in \mathbb{C}_{\mathcal{D}}$.

We first prove that function $f^\star$ is monotone. Let $\mathbf{z}, \mathbf{z}' \in Z$ such that $\mathbf{z} \preceq \mathbf{z}'$. There are several possibilities for the position of $\mathbf{z}$ and $\mathbf{z}'$: $i)$ $\mathbf{z}, \mathbf{z}' \in Z_1$, $ii)$ $\mathbf{z}, \mathbf{z}' \in Z_2$, $iii)$ $\mathbf{z}, \mathbf{z}' \in Z_3$, $iv)$ $\mathbf{z} \in Z_1, \mathbf{z}' \in Z_2$, $v)$ $\mathbf{z} \in Z_2, \mathbf{z}' \in Z_3$, $vi)$ $\mathbf{z} \in Z_1, \mathbf{z}' \in Z_3$. We want to verify that $f^\star(\mathbf{z}) \preceq f^\star(\mathbf{z}')$ in all those cases.

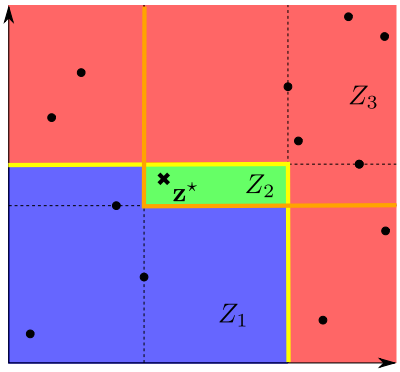


Figure 2.5: Partition of Z used to define the function $f^\star$ in (2.11). The area bounded from above by the yellow line represents the set $Z^+(\mathbf{z}^\star)$. The area bounded from below by the orange line represents the set $Z^-(\mathbf{z}^\star)$. The area Z_1 in blue contains all the data points smaller than $\mathbf{z}^\star$. The area Z_3 in red contains all the data points larger than $\mathbf{z}^\star$ or incomparable with $\mathbf{z}^\star$. The area Z_2 is colored in green.

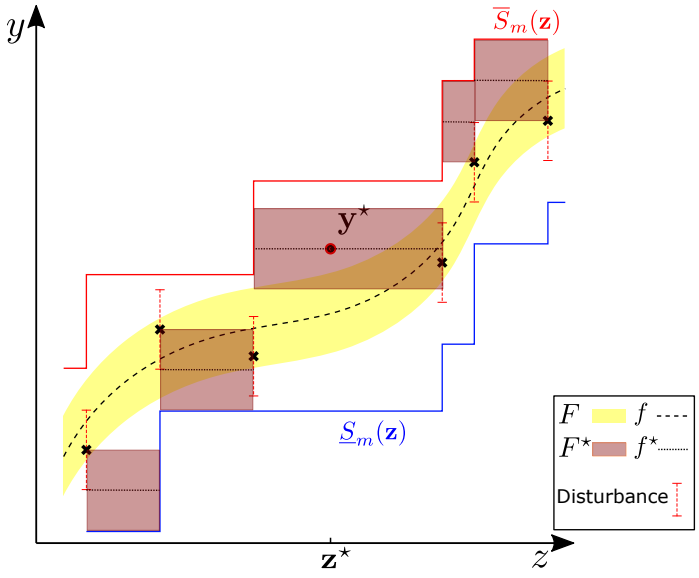


Figure 2.6: Construction of the map $F^\star$ according to (2.11).

- Case (i): Since $\mathbf{z} \preceq \mathbf{z}'$, we have $\mathbb{K}^-(\mathbf{z}) \subseteq \mathbb{K}^-(\mathbf{z}')$, which implies from (2.9), $\underline{S}_m(\mathbf{z}) \preceq \underline{S}_m(\mathbf{z}')$. Then, from (2.11), $f^\star(\mathbf{z}) \preceq f^\star(\mathbf{z}')$.
- Case (ii): From (2.11), we have $f^\star(\mathbf{z}) = f^\star(\mathbf{z}')$.
- Case (iii): Similar to case (i).
- Case (iv): It follows from the definition of Z_2 that $\mathbb{K}^-(\mathbf{z}') = \mathbb{K}^-(\mathbf{z}^\star)$. Then, since $\mathbf{z} \preceq \mathbf{z}'$, we have $\mathbb{K}^-(\mathbf{z}) \subseteq \mathbb{K}^-(\mathbf{z}')$ and thus $\mathbb{K}^-(\mathbf{z}) \subseteq \mathbb{K}^-(\mathbf{z}^\star)$. Hence, from (2.9), we get $\underline{S}_m(\mathbf{z}) \preceq \underline{S}_m(\mathbf{z}^\star)$. Since $\mathbf{z} \in Z_1$ and $\mathbf{z}' \in Z_2$ we get from (2.11),
$$f^\star(\mathbf{z}) = \underline{S}_m(\mathbf{z}) - \underline{\mathbf{d}} \preceq \underline{S}_m(\mathbf{z}^\star) - \underline{\mathbf{d}} \preceq f^\star(\mathbf{z}').$$
- Case (v): Similar to case (iv).
- Case (vi): Since $S_m \in \mathbb{S}_\mathcal{D}$ and $F \in \mathbb{C}_\mathcal{D}$, we get that $\underline{S}_m(\mathbf{z}) - \underline{\mathbf{d}} \preceq f(\mathbf{z})$ and $f(\mathbf{z}') \preceq \overline{S}_m(\mathbf{z}') - \overline{\mathbf{d}}$. Since f is monotone and $\mathbf{z} \preceq \mathbf{z}'$, we get $f(\mathbf{z}) \preceq f(\mathbf{z}')$. Finally, since $\mathbf{z} \in Z_1$ and $\mathbf{z}' \in Z_3$, we get from (2.11),
$$f^\star(\mathbf{z}) = \underline{S}_m(\mathbf{z}) - \underline{\mathbf{d}} \preceq \overline{S}_m(\mathbf{z}') - \overline{\mathbf{d}} = f^\star(\mathbf{z}').$$

Hence, we established the monotonicity of function $f^\star$.

We now prove that the second requirement of Definition 5 is satisfied. Let us consider $k \in \mathbb{K}$; we already know that $\mathbf{z}_k \in Z_1 \cup Z_3$. In the case where $\mathbf{z}_k \in Z_1$, we have from (2.9) and (2.11),
$$\mathbf{y}_k \preceq \underline{S}_m(\mathbf{z}_k) - \underline{\mathbf{d}} + \overline{\mathbf{d}} = f^\star(\mathbf{z}_k) + \overline{\mathbf{d}}.$$
Moreover, since $S_m \in \mathbb{S}_\mathcal{D}$, we have
$$\mathbf{y}_k \succeq \underline{S}_m(\mathbf{z}_k) = f^\star(\mathbf{z}_k) + \underline{\mathbf{d}}.$$
Therefore, $\mathbf{y}_k \in F^\star(\mathbf{x}_k)$. A similar statement can be shown when $\mathbf{z}_k \in Z_3$. It follows that $F^\star \in \mathbb{C}_\mathcal{D}$.

Hence, we have proved that for all $\mathbf{z}^\star \in Z \setminus Z_0$ and for all $\mathbf{y}^\star \in S_m(\mathbf{z}^\star)$, there exists a map $F^\star \in \mathbb{C}_\mathcal{D}$ such that $\mathbf{y}^\star \in F^\star(\mathbf{z}^\star)$. Hence, for all map $S \in \mathbb{S}_\mathcal{D}$, we have $\mathbf{y}^\star \in S(\mathbf{z}^\star)$. Hence, S_m is a minimal simulating map. □

We present some useful properties of minimal simulating maps. The first property establishes the monotonicity of the minimal simulating map defined in Equation (2.6)

Proposition 2. *The minimal simulating map S_m, as defined in Equation* (2.6)*, exhibits the monotonicity property.*

Proof. Consider $\mathbf{z}$ and $\mathbf{z}' \in Z$ such that $\mathbf{z} \preceq \mathbf{z}'$. This implies that $\mathbb{K}^+(\mathbf{z}') \subseteq \mathbb{K}^+(\mathbf{z})$ and $\mathbb{K}^-(\mathbf{z}) \subseteq \mathbb{K}^-(\mathbf{z}')$, which, in turn, leads to $\overline{S}_m(\mathbf{z}) \preceq \overline{S}_m(\mathbf{z}')$ and $\underline{S}_m(\mathbf{z}) \preceq \underline{S}_m(\mathbf{z}')$, following Equations (2.8) and (2.9) respectively. Consequently, for any $\mathbf{y} \in S_m(\mathbf{z})$, we have $\mathbf{y} \preceq \overline{S}_m(\mathbf{z}) \preceq \mathbf{y}'$, where $\mathbf{y}' = \overline{S}_m(\mathbf{z}') \in S_m(\mathbf{z}')$. Similarly, for any $\mathbf{y}' \in S_m(\mathbf{z}')$, we have $\mathbf{y}' \succeq \underline{S}_m(\mathbf{z}') \succeq \mathbf{y}$, where $\mathbf{y} = \underline{S}_m(\mathbf{z}) \in S_m(\mathbf{z})$. According to Definition 1, this confirms that S_m is monotonic. □

Next, we investigate the properties of the minimal simulating map obtained from combining two datasets.

Proposition 3. *Consider two datasets $\mathcal{D}'$ and $\mathcal{D}''$, and define the dataset $\mathcal{D} = \mathcal{D}' \cup \mathcal{D}''$. Let S_m, S'_m, and S''_m be the minimal simulating maps of $\mathcal{D}$, $\mathcal{D}'$, and $\mathcal{D}''$, respectively, as given by Equation* (2.6)*. Then, for any $\mathbf{z} \in Z$, we have:*

$$S_m(\mathbf{z}) = S'_m(\mathbf{z}) \cap S''_m(\mathbf{z}).$$

Proof. The data set $\mathcal{D}$ can be expressed as:

$$\mathcal{D} = \{(\mathbf{z}_k, \mathbf{y}_k) \mid \mathbf{y}_k \in F(\mathbf{z}_k), k \in \mathbb{K}_{\mathcal{D}}\},$$

with $\mathbb{K}_{\mathcal{D}} = \mathbb{K}_{\mathcal{D}'} \cup \mathbb{K}_{\mathcal{D}''}$ where $\mathbb{K}_{\mathcal{D}'}$ and $\mathbb{K}_{\mathcal{D}''}$ are the sets of indices of $\mathcal{D}'$ and $\mathcal{D}''$ respectively. Then, for all $\mathbf{z} \in Z$, we have

$$\mathbb{K}^-_{\mathcal{D}}(\mathbf{z}) = \mathbb{K}^-_{\mathcal{D}'}(\mathbf{z}) \cup \mathbb{K}^-_{\mathcal{D}''}(\mathbf{z}),\ \mathbb{K}^+_{\mathcal{D}}(\mathbf{z}) = \mathbb{K}^+_{\mathcal{D}'}(\mathbf{z}) \cup \mathbb{K}^+_{\mathcal{D}''}(\mathbf{z}).$$

Substituting into Equation (2.6), we can readily establish that for all $\mathbf{z} \in Z$, $S_m(\mathbf{z}) = S'_m(\mathbf{z}) \cap S''_m(\mathbf{z})$.

□

Proposition 3 has several implications. First, it means that the computation of a minimal simulating map can be easily parallelized. This is because the map can be computed for each data set independently, and then the results can be combined. This makes it possible to speed up the computation for large data sets. Second, Proposition 3 implies that pre-computed minimal simulating maps can be easily refined when new data becomes available. This is because the new data can be used to compute a new minimal simulating map, which can then be intersected with the pre-computed map. This will result in a new, more accurate, minimal simulating map.

Remark 2. *In this section, we provided a solution for computing a minimal simulating map of $\mathcal{D}$, which provides a tight over-approximation of the map F. Note that similar results can be obtained if we seek an over-approximation of the function f as it can be shown that*

$$\forall \mathbf{z} \in Z,\ f(\mathbf{z}) \in [\underline{S}_m(\mathbf{z}) - \underline{\mathbf{d}}, \overline{S}_m(\mathbf{z}) - \overline{\mathbf{d}}],$$

which can be seen in Figure 2.4. If we want to calculate a single-valued approximation of the function f, we can use a similar approach to the one used in the set membership technique [60] and build an estimation of f using the calculated bounds on the minimal simulating map

$$f(\boldsymbol{z}) = \frac{\overline{S}_m(\mathbf{z}) - \overline{\mathbf{w}} + \underline{S}_m(\mathbf{z}) - \underline{\mathbf{d}}}{2}.$$

It will be apparent from what follows that this estimation is piecewise constant and not continuous. Continuous estimation of f, included in the over-approximation map S_m, is derived in chapter 5 using the class of multi-affine functions.

Computation

Having characterized minimal simulating maps, let us now delve into the practical computation of these maps. Our approach and the used notations are illustrated in Figure 2.7, where a six data points case is demonstrated.

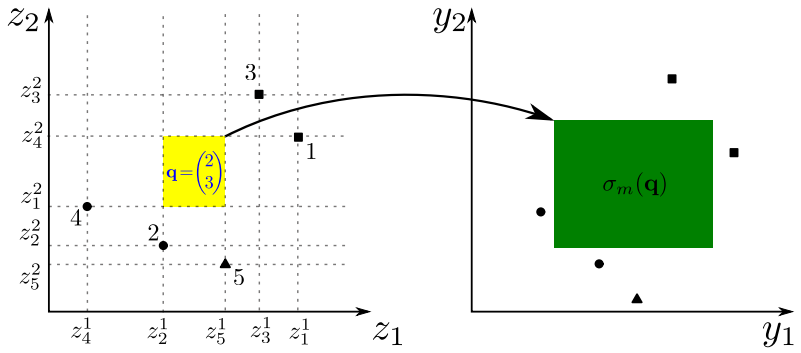


Figure 2.7: The figure shows the rectangular partition of the input set Z based on the collected data points and an example of one of the discrete variables **q**. All the points in yellow, aggregated in **q**, have the same output.

For that purpose, we introduce a rectangular partition of the input set Z, induced by the data $\mathcal{D}$. For simplicity, let us assume that Z is a closed interval of $\overline{\mathbb{R}}^n$, i.e. $Z = [\underline{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha}}]$. This is without loss of generality since it is always possible to embed Z in such a set. For each $i \in \{1, \ldots, n\}$, we sort the i^{th} components of all the data points in the input space (i.e., z_k^i, for $k \in \mathbb{K}$), so we have $z^i_{k_i^1} \leq \cdots \leq z^i_{k_i^{|\mathbb{K}|}}$. The sorted values are then used to define the finite partitions $(Z^i_{q^i})_{q^i \in Q^i}$ of $[\underline{\alpha}^i, \overline{\alpha}^i]$ where $Q^i = \{0, \ldots, |\mathbb{K}|\}$ and

$$\begin{cases} Z^i_0 &= [\underline{\alpha}^i, z^i_{k_i^1}), \\ Z^i_{q^i} &= [z^i_{k_i^{q^i}}, z^i_{k_i{}^{q^i+1}}),\ q^i = 1, \ldots, |\mathbb{K}| - 1, \\ Z^i_{|\mathbb{K}|} &= [z^i_{k_i^{|\mathbb{K}|}}, \overline{\alpha}^i]. \end{cases}$$

Then, let us define $Q = Q^1 \times \cdots \times Q^n$, and let the finite rectangular partition $(Z_{\mathbf{q}})_{\mathbf{q}\in Q}$ of Z be given for $\mathbf{q} = (q^1, \ldots, q^n)$ by $Z_{\mathbf{q}} = Z^1_{q^1} \times \cdots \times Z^n_{q^n}$.

Lemma 1. *Let S_m be the minimal simulating map of $\mathcal{D}$ given by* (2.6)*. Then, for all $\mathbf{q} \in Q$, for all $\mathbf{z}, \mathbf{z}' \in int\, Z_{\mathbf{q}}$, $S_m(\mathbf{z}) = S_m(\mathbf{z}')$.*

Proof. Because the partition is defined using the components of data points, we have, for all $\mathbf{q} \in Q$, for all $\mathbf{z}, \mathbf{z}' \in \operatorname{int} Z_{\mathbf{q}}$, $\mathbb{K}^-(\mathbf{z}) = \mathbb{K}^-(\mathbf{z}')$ and $\mathbb{K}^+(\mathbf{z}) = \mathbb{K}^+(\mathbf{z}')$. Subsequently, we obtain by (2.6), $S_m(\mathbf{z}) = S_m(\mathbf{z}')$. □

Building on the property presented in Lemma 1, we can introduce the new map $\sigma_m : Q \rightrightarrows \overline{\mathbb{R}}^m$ defined as follows

$$\forall \mathbf{q} \in Q,\ \sigma_m(\mathbf{q}) = S_m(\mathbf{z}),\ \text{for } \mathbf{z} \in \operatorname{int} Z_{\mathbf{q}}. \tag{2.12}$$

We also define a quantization function $\phi_m : Z \to Q$ associated to the finite data-induced partition $(Z_{\mathbf{q}})_{\mathbf{q}\in Q}$ as follows

$$\forall \mathbf{z} \in Z,\ \forall \mathbf{q} \in Q,\ \phi_m(\mathbf{z}) = \mathbf{q} \iff \mathbf{z} \in Z_{\mathbf{q}}. \tag{2.13}$$

Proposition 4. *Let σ_m and ϕ_m be given by* (2.12) *and* (2.13)*, then $\sigma_m \circ \phi_m$ is a minimal simulating map of $\mathcal{D}$.*

Proof. It follows directly from Lemma 1 and from the definition of σ_m and ϕ_m that $\sigma_m \circ \phi_m = S_m$ a.e.. Then, let us show that $\sigma_m \circ \phi_m$ is a simulation map. From the construction of the partition $(Z_{\mathbf{q}})_{\mathbf{q}\in Q}$, it follows that for all $\mathbf{q} \in Q$, for all $\mathbf{z} \in Z_{\mathbf{q}}$, for all $\mathbf{z}' \in \operatorname{int} Z_{\mathbf{q}}$, $\mathbb{K}^-(\mathbf{z}') \subseteq \mathbb{K}^-(\mathbf{z})$ and $\mathbb{K}^+(\mathbf{z}') \subseteq \mathbb{K}^+(\mathbf{z})$. Hence, from (2.6), $S_m(\mathbf{z}) \subseteq S_m(\mathbf{z}')$. Then, it results that $S_m(\mathbf{z}) \subseteq \sigma_m \circ \phi_m(\mathbf{z})$, for all $\mathbf{z} \in Z$. Since $S_m \in \mathbb{S}_{\mathcal{D}}$, we have that $\sigma_m \circ \phi_m \in \mathbb{S}_{\mathcal{D}}$. Therefore, according to Proposition 1, $\sigma_m \circ \phi_m$ is a minimal simulating map of $\mathcal{D}$. □

Hence, computing the finite partition $(Z_{\mathbf{q}})_{\mathbf{q}\in Q}$ and the map σ_m offers an effective way to compute a minimal simulating map and store it. Let us remark that the number of elements in Q is $(|\mathbb{K}|+1)^n$. Hence, it grows polynomially with the number of data points $|\mathbb{K}|$, and exponentially with the dimension n of the input space Z. Therefore, it is computationally prohibitive to use this approach for large data sets, particularly with high-dimensional input spaces. In the following, we tackle this problem by fixing a partition a priori and by finding the best simulating map on this partition.

2.2.1 . Simulating maps on fixed partitions

Instead of relying on the data set $\mathcal{D}$ to partition Z, which makes calculating the simulating map computationally expensive, we will assume that a rectangular partition of Z is given a priori. In the following, we characterize the tightest interval-valued simulating map on this partition. Our approach is illustrated in Figure 2.8

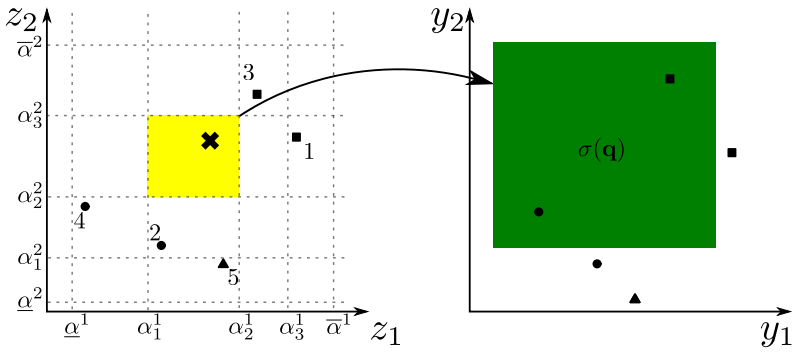


Figure 2.8: The figure shows the predefined partition of the input set Z. The function ϕ maps all the points inside a cell (e.g., square in yellow) to a discrete variable. The map σ is the minimal interval-valued simulating map.

Construction

For each coordinate $i \in \{1, \dots, n\}$, let be given finite partitions $(R^i_{q^i})_{q^i \in Q^i}$ of $[\underline{\alpha}^i, \overline{\alpha}^i]$ where $Q^i = \{0, \dots, K^i\}$ and

$$\begin{cases} R^i_0 & = [\underline{\alpha}^i, \alpha^i_1), \\ R^i_{q^i} & = [\alpha^i_{q^i}, \alpha^i_{q^i+1}),\ q^i = 1, \dots, K^i - 1, \\ R^i_{K^i} & = [\alpha^i_{K^i}, \overline{\alpha}^i], \end{cases}$$

where $\underline{\alpha}^i < \alpha^i_1 < \cdots < \alpha^i_{K^i} < \overline{\alpha}^i$. Note that the number of partition elements $K^i + 1$ can be different from one coordinate to another. Then, let us define $Q = Q^1 \times \cdots \times Q^n$, and let the finite rectangular partition $(R_{\mathbf{q}})_{\mathbf{q} \in Q}$ of Z be given for $\mathbf{q} = (q^1, \dots, q^n)$ by $R_{\mathbf{q}} = R^1_{q^1} \times \cdots \times R^n_{q^n}$.

Let us define the map $\sigma : Q \rightrightarrows \overline{\mathbb{R}}^m$ given for all $\mathbf{q} \in Q$ by

$$\begin{aligned} \sigma(\mathbf{q}) = & \left(\bigcap_{k \in \mathbb{K}^-(\underline{\mathbf{z}}_{\mathbf{q}})} \{\mathbf{y} \in \overline{\mathbb{R}}^m | \ \mathbf{y}_k + \underline{\mathbf{d}} - \overline{\mathbf{d}} \preceq \mathbf{y}\} \right) \\ & \cap \left(\bigcap_{k \in \mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}})} \{\mathbf{y} \in \overline{\mathbb{R}}^m | \ \mathbf{y} \preceq \mathbf{y}_k + \overline{\mathbf{d}} - \underline{\mathbf{d}}\} \right) \end{aligned} \tag{2.14}$$

where $\underline{\mathbf{z}}_{\mathbf{q}} = \inf R_{\mathbf{q}}$, $\overline{\mathbf{z}}_{\mathbf{q}} = \sup R_{\mathbf{q}}$, and $\mathbb{K}^-$, $\mathbb{K}^+$ are defined as in (2.7). From (2.8), (2.9) and (2.14), we get that σ is an interval-valued map: for all $\mathbf{q} \in Q$, $\sigma(\mathbf{q}) = [\underline{\sigma}(\mathbf{q}), \overline{\sigma}(\mathbf{q})]$ with $\overline{\sigma}(\mathbf{q}) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$ and $\underline{\sigma}(\mathbf{q}) = \underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}})$.

We also consider a quantization function $\phi : Z \to Q$ associated to the finite partition $(R_{\mathbf{q}})_{\mathbf{q} \in Q}$ and defined as

$$\forall \mathbf{z} \in Z,\ \forall \mathbf{q} \in Q,\ \phi(\mathbf{z}) = \mathbf{q} \iff \mathbf{z} \in R_{\mathbf{q}}. \tag{2.15}$$

Theorem 2. *Let σ and ϕ be given by* (2.14) *and* (2.15)*, then the following properties hold:*

1. *$\sigma \circ \phi \in \mathbb{S}_{\mathcal{D}}$ (Simulation);*
2. *For any interval-valued map $\sigma' : Q \rightrightarrows \overline{\mathbb{R}}^m$, such that $\sigma' \circ \phi \in \mathbb{S}_{\mathcal{D}}$, it holds for all $\mathbf{q} \in Q$, $\sigma(\mathbf{q}) \subseteq \sigma'(\mathbf{q})$ (Minimality).*

Proof. Let S_m be the minimal simulating map in (2.6). Let us prove the simulation property. Let $\mathbf{z} \in Z$, then $\underline{\mathbf{z}}_{\phi(\mathbf{z})} \preceq \mathbf{z} \preceq \overline{\mathbf{z}}_{\phi(\mathbf{z})}$, which yields

$$\mathbb{K}^-(\underline{\mathbf{z}}_{\phi(\mathbf{z})}) \subseteq \mathbb{K}^-(\mathbf{z}) \text{ and } \mathbb{K}^+(\overline{\mathbf{z}}_{\phi(\mathbf{z})}) \subseteq \mathbb{K}^+(\mathbf{z}).$$

Hence, by (2.6) and (2.14), $S_m(\mathbf{z}) \subseteq \sigma \circ \phi(\mathbf{z})$. Since $S_m \in \mathbb{S}_{\mathcal{D}}$, we get $\sigma \circ \phi \in \mathbb{S}_{\mathcal{D}}$.

Next, let us prove the minimality property. Let us consider an interval-valued map $\sigma' : Q \rightrightarrows \overline{\mathbb{R}}^m$ such that $\sigma' \circ \phi \in \mathbb{S}_{\mathcal{D}}$, and let $q \in Q$. From (2.7), it can be seen that there exists a neighborhood $\overline{N}_{\mathbf{q}}$ of $\overline{\mathbf{z}}_{\mathbf{q}}$ such that for all $\mathbf{z} \in \overline{N}_{\mathbf{q}}$ with $\mathbf{z} \preceq \overline{\mathbf{z}}_{\mathbf{q}}$ it holds $\mathbb{K}^+(\mathbf{z}) = \mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}})$, which yields by (2.6) $\overline{S}_m(\mathbf{z}) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$. Then, let $\mathbf{z}^\star \in (\overline{N}_{\mathbf{q}} \cap R_{\mathbf{q}}) \setminus Z_0$ with Z_0 the set of measure zero defined as in (2.10). Let us remark that $\mathbf{z}^\star \preceq \overline{\mathbf{z}}_{\mathbf{q}}$ and hence $\overline{S}_m(\mathbf{z}^\star) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$. From the proof of Theorem 1, we get that there exists $F^\star \in \mathbb{C}_{\mathcal{D}}$ such that $\overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}}) = \overline{S}_m(\mathbf{z}^\star) \in F^\star(\mathbf{z}^\star)$. Then, $\sigma' \circ \phi \in \mathbb{S}_{\mathcal{D}}$ gives us that $\overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}}) \in \sigma' \circ \phi(\mathbf{z}^\star)$. Moreover, $\mathbf{z}^\star \in R_{\mathbf{q}}$, gives us that $\sigma' \circ \phi(\mathbf{z}^\star) = \sigma'(\mathbf{q})$. Hence, $\overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}}) \in \sigma'(\mathbf{q})$. Similarly, we can show that $\underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}}) \in \sigma'(\mathbf{q})$. Then since σ' is interval-valued, we get that $[\underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}}), \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})] \subseteq \sigma'(\mathbf{q})$. Then, $\sigma(\mathbf{q}) = [\underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}}), \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})]$ gives us $\sigma(\mathbf{q}) \subseteq \sigma'(\mathbf{q})$. □

It follows from Theorem 2 that it is possible to define a notion of minimal simulating map of $\mathcal{D}$ relative to a given partition $(R_{\mathbf{q}})_{\mathbf{q} \in Q}$ of Z. It should be noticed that similar results to Propositions 2 and 3 hold for the simulating map $\sigma \circ \phi$. Finally, one can check that if the partition $(R_{\mathbf{q}})_{\mathbf{q} \in Q}$ of Z coincides with the data-induced partition defined in the previous section, $\sigma \circ \phi$ is minimal simulating map of $\mathcal{D}$. The difference between the maps $\sigma \circ \phi$ and $\sigma_m \circ \phi$ can be examined by comparing Figure 2.7 and Figure 2.8.

Remark 3. *For a given number of cells, typically smaller than data-induced partition, choosing an optimal partition $(D_{\mathbf{q}})_{\mathbf{q} \in Q}$ is a complicated problem. However, it can be easily shown that the optimal partition (achieving the minimal over-approximation volume) is aligned with some of the data points components. This makes it possible to rely on heuristics to choose good yet sub-optimal partitions. For instance, a possible approach would be to aggregate cells of the data-induced partition where a lot of data is available.*

2.3 . Efficent calculation of the minimal simulating map

Given the partition $(R_{\mathbf{q}})_{\mathbf{q}\in Q}$ of Z, computing the simulating map defined in the previous section amounts to computing the map $\sigma : Q \rightrightarrows \overline{\mathbb{R}}^m$ given by (2.14). A straightforward algorithm to compute σ is as follows. For each $\mathbf{q} \in Q$, we go through all the data points in $\mathcal{D}$ and determine $\mathbb{K}^-(\underline{\mathbf{z}}_{\mathbf{q}})$ and $\mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}})$. Then, one gets $\sigma(\mathbf{q})$ by (2.14). That makes the overall complexity $\mathcal{O}(|\mathbb{K}| \times |Q|)$, i.e., bilinear with respect to the number of data points and to the number of partition elements. In this section, we present a more efficient approach to calculating σ.

For simplicity, we assume in the following that for all $k \in \mathbb{K}$, $\mathbf{z}_k \in \operatorname{int} Z$, i.e., that no data point lies on the boundary of the input set. Then, since $\overline{\sigma}(\mathbf{q}) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$, it follows from (2.8) that for all $\mathbf{q} \in Q$ such that $q^i = K^i$ (recall that K^i is number of partitions on the i^{th} dimension), for some $i \in \{1, \dots, n\}$, $\overline{\sigma}(\mathbf{q}) = (+\infty, \dots, +\infty)$. Similarly, since $\underline{\sigma}(\mathbf{q}) = \underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}})$, it follows from (2.9) that for all for all $\mathbf{q} \in Q$ such that $q^i = 0$, for some $i \in \{1, \dots, n\}$, $\underline{\sigma}(\mathbf{q}) = (-\infty, \dots, -\infty)$.

Then, let us consider the following subsets of Q:

$$\overline{Q} = \{0, \dots, K^1 - 1\} \times \cdots \times \{0, \dots, K^n - 1\},$$
$$\underline{Q} = \{1, \dots, K^1\} \times \cdots \times \{1, \dots, K^n\}.$$

One needs to compute $\overline{\sigma}(\mathbf{q})$ and $\underline{\sigma}(\mathbf{q})$ for $\mathbf{q} \in \overline{Q}$ and $\mathbf{q} \in \underline{Q}$, respectively. For that purpose, let us first define the functions $\overline{\sigma}_0 : \overline{Q} \to \overline{\mathbb{R}}^m$ and $\underline{\sigma}_0 : \underline{Q} \to \overline{\mathbb{R}}^m$ as follows:

$$\overline{\sigma}_0(\mathbf{q}) = \inf \left\{ \mathbf{y}_k + \overline{\mathbf{d}} - \underline{\mathbf{d}} \mid \mathbf{z}_k \in \operatorname{cl} R_{\operatorname{up}(\mathbf{q})} \right\}, \tag{2.16}$$
$$\underline{\sigma}_0(\mathbf{q}) = \sup \left\{ \mathbf{y}_k + \underline{\mathbf{d}} - \overline{\mathbf{w}} \mid \mathbf{z}_k \in \operatorname{cl} R_{\operatorname{lo}(\mathbf{q})} \right\}, \tag{2.17}$$

where $\operatorname{up}(\mathbf{q}) = \mathbf{q} + \mathbb{1}_n$, $\operatorname{lo}(\mathbf{q}) = \mathbf{q} - \mathbb{1}_n$, and $\mathbb{1}_n = (1, \dots, 1)$. An illustration of the elements $\operatorname{up}(\mathbf{q})$ and $\operatorname{lo}(\mathbf{q})$ can be seen in Figure 2.9. To compute $\overline{\sigma}_0, \underline{\sigma}_0$, we start by initializing $\overline{\sigma}_0(\mathbf{q}) = (+\infty, \dots, +\infty)$ for all $\mathbf{q} \in \overline{Q}$, and $\underline{\sigma}_0(\mathbf{q}) = (-\infty, \dots, -\infty)$ for all $\mathbf{q} \in \underline{Q}$. Then, we go through all the points in the set $\mathcal{D}$; for each entry $(\mathbf{z}_k, \mathbf{y}_k)$, we find all $\mathbf{q}$ such that $\mathbf{z}_k \in \operatorname{cl} D_{\mathbf{q}}$. Then, we update the value of $\underline{\sigma}_0(\operatorname{up}(\mathbf{q}))$ and $\overline{\sigma}_0(\operatorname{lo}(\mathbf{q}))$ using (2.16) and (2.17). The partition is stored and sorted component-wise, so to find $\mathbf{q}$, we can do a binary search for each component of $\mathbf{z}_k$. Therefore, computing $\overline{\sigma}_0, \underline{\sigma}_0$ is done with a complexity $\mathcal{O}(|\mathbb{K}| \times \sum_i \log(K^i))$ or equivalently $\mathcal{O}(|\mathbb{K}| \times \log(|Q|))$.

We now present a result that will allow us to compute the map σ sequentially.

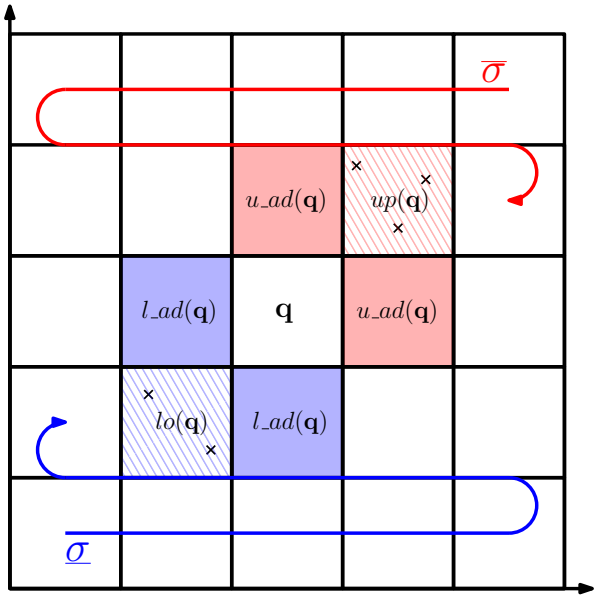


Figure 2.9: A representation of the sets $\mathrm{up}(\mathbf{q})$, $\mathrm{u\_ad}(\mathbf{q})$, $\mathrm{lo}(\mathbf{q})$, $\mathrm{l\_ad}(\mathbf{q})$ and the order in which we calculate $\overline{\sigma}$ and $\underline{\sigma}$.

Proposition 5. *Let σ be the interval-valued map given by (2.14), its upper and lower bounds $\overline{\sigma}, \underline{\sigma}$ satisfy:*

$$\forall \mathbf{q} \in \overline{Q},\ \overline{\sigma}(\mathbf{q}) = \min\left(\inf\left\{\overline{\sigma}(\mathbf{q}')|\ \mathbf{q}' \in u\_\, ad(\mathbf{q})\right\}, \overline{\sigma}_0(\mathbf{q})\right), \tag{2.18}$$

$$\forall \mathbf{q} \in \underline{Q},\ \underline{\sigma}(\mathbf{q}) = \max\left(\sup\left\{\underline{\sigma}(\mathbf{q}')|\ \mathbf{q}' \in l\_\, ad(\mathbf{q})\right\}, \underline{\sigma}_0(\mathbf{q})\right), \tag{2.19}$$

where

$$u\_\, ad(\mathbf{q}) = \{\mathbf{q}' \in Q|\ \exists k \in \{1, \ldots, n\}, \mathbf{q}' - \mathbf{q} = \mathbf{e}_k\},$$

$$l\_\, ad(\mathbf{q}) = \{\mathbf{q}' \in Q|\ \exists k \in \{1, \ldots, n\}, \mathbf{q} - \mathbf{q}' = \mathbf{e}_k\},$$

and $\mathbf{e}_k \in \mathbb{R}^n$ whose k^{th} component is 1 and all others are 0.

An illustration of the sets $\mathrm{u\_ad}(\mathbf{q})$ and $\mathrm{l\_ad}(\mathbf{q})$ can be seen in Figure 2.9.

Proof. We prove the property for the upper bound $\overline{\sigma}$, and the proof for the lower bound $\underline{\sigma}$ follows similarly. Let $q \in \overline{Q}$, it can be seen that

$$\mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}}) = \Big(\bigcup_{\mathbf{q}' \in \mathrm{u\_ad}(\mathbf{q})} \mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}'})\Big) \cup \{k \in \mathbb{K}|\ \mathbf{z}_k \in D_{\mathrm{up}}(\mathbf{q})\}.$$

Then, since $\overline{\sigma}(\mathbf{q}) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$, (2.18) follows directly from (2.8) and the equality above. □

From Proposition 5, we can see that to compute $\overline{\sigma}$ we go through all $\mathbf{q} \in Q$ sequentially in a decreasing order, starting from $\mathbf{q} = (K^1, \ldots, K^n)$ as represented

by Figure 2.9. For $\underline{\sigma}$ we start from $\mathbf{q} = (1, \ldots, 1)$ and though all $\mathbf{q} \in Q$ in an increasing order.

Proposition 6. *The map σ can be computed with complexity $\mathcal{O}(|\mathbb{K}| \times \log(|Q|) + |Q|)$.*

Proof. We already showed the complexity of computing $\overline{\sigma}_0$ and $\underline{\sigma}_0$ is $\mathcal{O}(|\mathbb{K}| \times \log(|Q|))$. To compute σ we should go through all the elements $\mathbf{q} \in Q$ twice, one in decreasing order to compute $\overline{\sigma}$ and one in increasing order to compute $\underline{\sigma}$. Therefore, the complexity of computing σ is $\mathcal{O}(|\mathbb{K}| \times \log(|Q|) + |Q|)$. □

Remark 4. *In the special case where the partition $(R_{\mathbf{q}})_{\mathbf{q} \in Q}$ of Z is uniform, and all the cells $R_{\mathbf{q}}$ have the same dimension, the binary search step is not needed and finding $\mathbf{q}$ can be done in constant time. Then, the complexity of computing σ is $\mathcal{O}(|\mathbb{K}| + |Q|)$.*

2.4 . Over-approximating unknown linear functions

In the case where the unknown function f is linear, we can apply a transformation to the data $\mathcal{D}$ to make it as if the data is generated by a monotone map. Let the unknown function f be Linear

$$f(\mathbf{z}) = A \cdot \mathbf{z} + \mathbf{b},$$

Where $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are unknown. Let A_i be the i-th row of A.

$$A = \begin{bmatrix} \vdots \\ A_i \\ \vdots \end{bmatrix}$$

We have that the i-th component of f is:

$$f^i(\mathbf{z}) = A_i \cdot \mathbf{z} + b_i.$$

Let us define the variable $\mathbf{z}' \in \overline{\mathbb{R}}^n$ as follows:

$$z'^j = \begin{cases} z^j & \text{if } A_{i,j} \geq 0 \\ -z^j & \text{if } A_{i,j} < 0 \end{cases}$$

Then we have that $f^i(\mathbf{z}) = A_i \cdot \mathbf{z}' + b_i$ is monotone because

$$\frac{\partial f^i}{\partial z'^j}(\mathbf{z}') = \begin{cases} A_{i,j} & \text{if } A_{i,j} \geq 0 \\ -A_{i,j} & \text{if } A_{i,j} < 0 \end{cases}$$

Therefore, we can apply the method presented in Section 2.1.2 to the data $\mathcal{D}$ to find an over-approximation of the map F, after applying the transformation to the data $\mathcal{D}$:

$$\mathcal{D}' = \{(\mathbf{z}'_k, \mathbf{y}_k) \mid (\mathbf{z}_k, \mathbf{y}_k) \in \mathcal{D},\ k \in \mathbb{K}\}.$$

The sign of the components of the matrix A can be inferred from the physics of the system. For example, in the case of a mass-spring-damper system, the signs of the components of the matrix A are dictated by its physics, as the tension in the spring is always trying to pull the mass toward the equilibrium position and the damping force is always trying to slow down the mass. In section 3.3, we will present a method to verify the monotonicity of unknown functions from data. This method can be used to verify the monotonicity of the transformed data $\mathcal{D}'$. Therefore, given that the function is linear, we can test all the possible transformations of the data $\mathcal{D}$ and find the one that makes the data as if it is generated by a monotone function. Then we can apply the method presented in Chapter 2 to the transformed data $\mathcal{D}'$ to find the least conservative over-approximation of the unknown map F.

Finally, in this section, we discussed how to learn linear functions from undisturbed data. Extending the approach to the case where there is an added bounded disturbance, is straightforward.

2.5 . Numerical examples

In this section, we present a numerical example to test the performance of the introduced over-approximation. We use interval domain, $Z = [\underline{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha}}]$, $\underline{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha}} \in \overline{\mathbb{R}}^n$ for the maps, we are trying to over-approximate. For $i \in \{1, \ldots, n\}$, we define the finite partition $(D^i_{q^i})_{q^i \in Q^i}$ as follows, $Q^i = \{0, \ldots, K^i\}$ and

$$\begin{cases} D^i_0 & = [\underline{\alpha}^i, \alpha^i_1) \\ D^i_{q^i} & = [\frac{q^i-1}{K^i-1}(\alpha^i_{K^i} - \alpha^i_1) + \alpha^i_1, \frac{q^i}{K^i-1}(\alpha^i_{K^i} - \alpha^i_1) + \alpha^i_1), \\ & \qquad\qquad q^i = 1, \ldots, K^i - 1 \\ D^i_{K^i} & = [\alpha^i_{K^i}, \overline{\alpha}^i] \end{cases} \tag{2.20}$$

We choose $\alpha^i_1 = \underline{\alpha}^i + c\ (\overline{\alpha}^i - \underline{\alpha}^i)$, $\alpha^i_{K^i} = \overline{\alpha}^i - c\ (\overline{\alpha}^i - \underline{\alpha}^i)$, and c is a constant specific to each example. This scheme of discretizing will be adopted throughout the thesis.

We present a set of experiments envisioned to test and visualize the algorithms, introduced in this chapter, for the over-approximation of set-valued maps.

To quantitatively measure the performance of the over-approximation, we can check the execution time and the conservatism in the resulting over-approximation. We calculate the conservatism of the over-approximation using the following per-

formance criterion

$$\mu(\mathcal{D}, Q) = \frac{\sum_{\mathbf{q} \in Q'} (vol(Z_{\mathbf{q}}) \times \sigma(\mathbf{q}))}{\sum_{\mathbf{q} \in Q'} (vol(Z_{\mathbf{q}}) \times vol(D))} \tag{2.21}$$

where

$$Q' = \{\mathbf{q} \in Q \big| \ -\infty < \underline{\sigma}^i(\mathbf{q}),\ \overline{\sigma}^i(\mathbf{q}) < \infty, \forall i \in \{1, \ldots, n\}\}.$$

The denominator of μ represents the volume of the graph of the unknown map for the part of space where we can find an over-approximation, whereas the numerator represents the volume of our over-approximation. μ can take its value in the interval $[1, \infty)$, and the smaller its value is, the less conservative the over-approximation is.

In this example, we consider a monotone set-valued map $F : [-\pi, \pi] \times [-\pi, \pi] \rightrightarrows \overline{\mathbb{R}}$ given by

$$F(\mathbf{z}) = \{2\ (\sin z^1 + \sin z^2 + z^1 + z^2) + d \mid d \in [-0.1, 0.1]\} \tag{2.22}$$

We use F to generate the sets of data $\mathcal{D}$ used in the subsequent experiments. First, we visualized the over-approximation. We sampled $|\mathbb{K}| = 10^6$ data points. The parameters of the partition are chosen as follows $K^1 = 30$, $K^2 = 30$, $c = 0.01$. Figure 2.10 shows the undisturbed function in solid and the over-approximation calculated from data. We see how the undisturbed function is included in the over-approximation.

2.5.1 . The effect of changing the number of data points

To study the effect of changing the number of data points, we chose and fixed a partition, $K^1 = 100$, $K^2 = 100$, $c = 0.01$. Then, for an increasing number of data points, we calculated the over-approximation of the map F and measured the execution time and the performance criterion. For each number of data points, we redo the experiment a hundred times using different randomly generated data sets. The results of this statistical study of changing the number of data points are shown in Figure 2.11. We can see from the figure the linear relation between the number of data points and the execution time as predicted in Proposition 6. We also see how the conservatism in the calculated over-approximation decreases with the increase in the number of data points. Note that even if the number of points increases toward infinity, the performance criterion μ will not reach one, because we are using a fixed partition. For μ to reach one, both the number of data points and the number of partition elements should go to infinity.

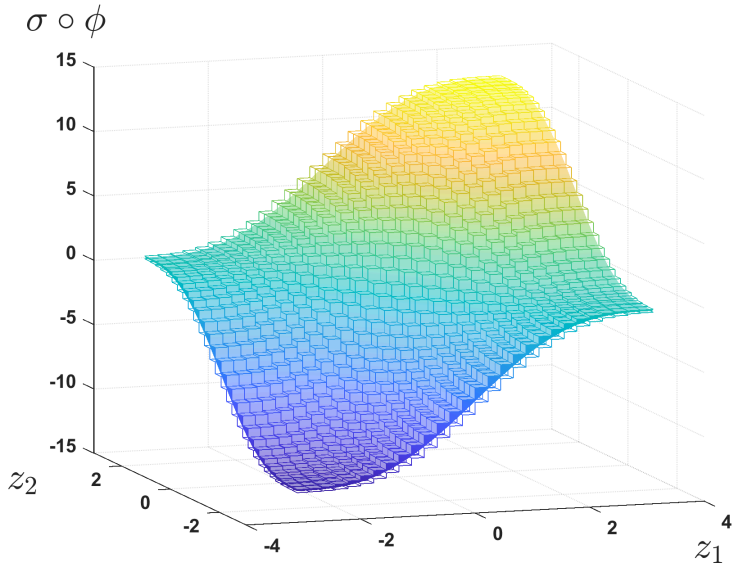


Figure 2.10: Map $F(\mathbf{z})$ with $w = 0$ everywhere is represented in solid. The upper and lower bounds of the over-approximation are represented in transparency

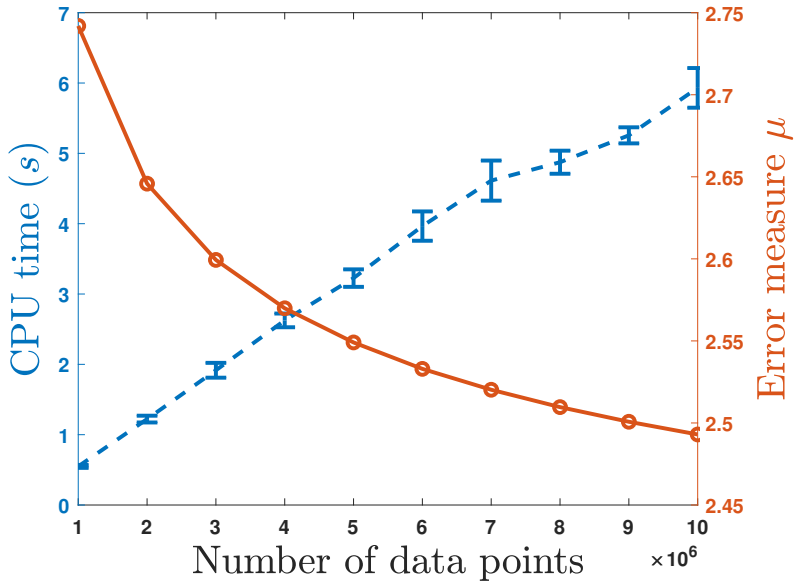


Figure 2.11: Line in blue represents the average time of execution with respect to the number of data points, with bars representing the standard deviation. The line in orange shows the relation between the number of points and the performance criterion $\mu(\mathcal{D}, Q)$.

2.5.2 . The effect of changing the size of the partition

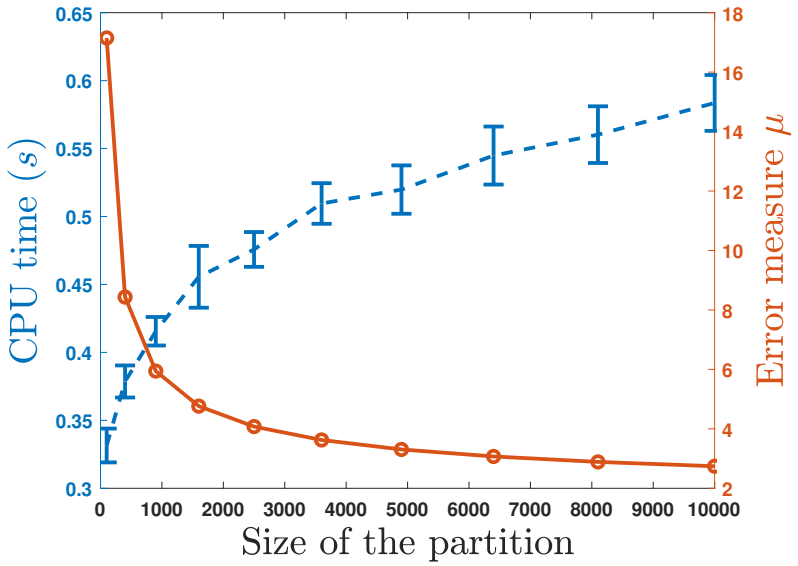


Figure 2.12: Line in blue represents the average time of execution with respect to the number of cells in the partitions, with bars representing the standard deviation. The line in orange represents the performance criterion $\mu(\mathcal{D}, Q)$ with respect to the number of cells in the partitions.

In the second experiment, we fixed the number of data points $|\mathbb{K}| = 10^6$, and changed the size of the partition. For each size considered, the partition is chosen such that $K^1 = K^2$, and $c = 0.01$. We also redo the calculation for each size a hundred times using different randomly sampled data sets. Although we are using a uniform partition, we do not make use of this fact in the calculation of the over-approximation, and we still use binary search to find the partition element that contains the data point. Figure 2.12 shows the results of the experiment. Time of execution increases logarithmically when $|Q|$ the number of partition elements is small. Then, the relation becomes linear for big values of $|Q|$. This behavior can be justified by the complexity relation introduced in Proposition 6. Similar to the first experiment, the performance criterion decreases with the increase in the number of partition elements.

2.6 . Conclusion

In this chapter, we defined the class of monotone systems and showed that the monotonicity property can be used to over-approximate the set-valued map, representing the system and disturbance from a set of data generated by the system. We introduced the data-driven approach to over-approximate set-valued maps. The approach is based on the idea of partitioning the input space and then finding a minimal simulating map for each partition element. We have shown that the minimal simulating map can be computed efficiently. We have also shown that

the minimal simulating map is the least conservative over-approximation of the set-valued map. Finally, we have presented a numerical example to illustrate the approach and to show its efficiency.

3 - Data-Driven Over-approximation of Systems with Bounded Derivative Functions

We will now shift focus to the case where the unknown function f is not necessarily monotone but has bounded derivatives with known upper and lower bounds.

Let us consider a map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$, where $Z \subseteq \overline{\mathbb{R}}^n$ and such that

$$\forall \mathbf{z} \in Z,\ F(\mathbf{z}) = f(\mathbf{z}) + D, \tag{3.1}$$

where $f : Z \to \overline{\mathbb{R}}^m$ is a function with bounded derivatives and $D \subseteq \overline{\mathbb{R}}^m$ is a bounded set. Let for all $\mathbf{z} \in Z$:

$$\frac{\partial f^i}{\partial z^j}(\mathbf{z}) \in [\underline{\gamma}_{ij}, \overline{\gamma}_{ij}],\ i \in \{1, \dots, m\},\ j \in \{1, \dots, n\},$$

where the bounds $\underline{\gamma}_{ij}, \overline{\gamma}_{ij} \in \mathbb{R}$ are known. Making the derivative of f bounded, Similar to (2.5), a dataset $\mathcal{D}$ is generated using the map F.

$$\mathcal{D} = \{(\mathbf{z}_k, \mathbf{y}_k) \mid \mathbf{y}_k \in f(\mathbf{z}_k) + D, k \in \mathbb{K}\} \tag{3.2}$$

where $\mathbb{K}$ is a finite index set. In this chapter, we will present a method to compute maps that over-approximate the unknown map F which generated the data $\mathcal{D}$.

When it comes to finding data-driven over-approximations of unknown functions that have bounded derivatives or are Lipschitz continuous, the literature contains several methods. We present the ones that find deterministic over-approximations, similar to our approach. The set membership approach offers an example of how we can find those deterministic over-approximations. In [60], the set membership approach is used for the identification of nonlinear systems. Set membership approaches are used in [19] to synthesize nonlinear model predictive controllers. In contrast to [19], our introduced approaches do not assume anything about the stability of the unknown systems. Also, the number of points that can be handled is greater than the number addressed in the set membership approach. The way set membership approaches over-approximate the unknown map F is a special case of the approach called kinky inferences, studied extensively [17]. The work in [17] aims to find a machine learning algorithm that defines an inference for unknown Holder continuous functions. The inference should exhibit the following properties: i) Conservatism: in the meaning that the inference should define uncertainty hyperrectangles guaranteed to contain the true value of the function, similar to our assumption that the resulting over-approximation should contain the true value of the unknown map F. ii) Monotonicity: Adding more data should not increase the size of the uncertainty hyperrectangles. iii) Convergence: The uncertainty hyperrectangles should converge to the true disturbance set D if we sample

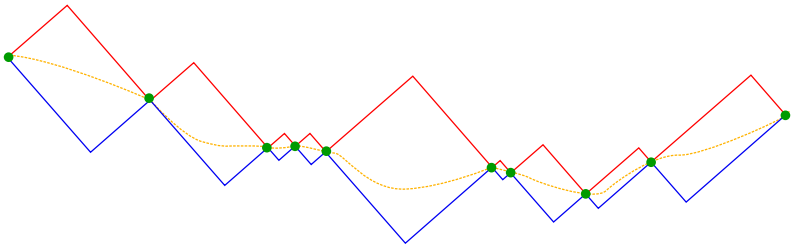


Figure 3.1: Set membership approach.

the domain densely. iv) Minimality (optimality): The uncertainty hyperrectangles should be as small as possible. The main difference between our approach and the approach in [17] is that we relax the demands regarding convergence and minimality (optimality) in favor of speed of computation. Instead of an inference that converges to the true map F, but found using an algorithm that needs to be repeated for each query point, we find an over-approximation that is calculated, on a predefined partition of the domain Z, once and can be used to over-approximate the unknown map F at any point in the domain Z in constant time. This means that for the over-approximation to converge to the true map F, we need to sample the domain densely and the size of the cells in the partition should go to zero. Regarding minimality (optimality), we were able to guarantee the minimality of the over-approximation on the predefined partition of the domain Z in the case of monotone maps (studied in Chapter 2). In the case of maps with bounded derivatives, finding optimal over-approximations is rather trickier, and requires costly optimization algorithms. Instead, we focused on finding over-approximations that are easy to compute while also being useful in practice.

Another example of robust over-approximation can be found in [42], where interval observers for partially unknown mixed-monotone nonlinear systems are found, assuming the unknown part is Lipschitz continuous. Lipschitz dynamical systems with known bounds on the Lipschitz constants are also studied in [66] to reach piecewise affine set-valued models. Zonotopes were used in [1] to compute reachable sets directly from noisy data generated by Lipschitz continuous systems.

The chapter is organized as follows. In Section 3.1, we will present a method to compute data-driven over-approximations of maps with bounded derivatives. In Section 3.2, we will present a method to compute online over-approximations of maps with bounded derivatives. We introduced a method to find the bounds on the derivatives of the unknown map f in Section 3.3. In Section 3.4, we will present the results of the experiments we conducted to test the introduced methods. Finally, we will conclude the chapter in Section 3.5.

3.1 . Over-approximating bounded derivative functions

Given only the bounds on the derivatives of the unknown function, the set membership approach offers the least conservative over-approximation of the unknown function [60]. The over-approximation is calculated out of a noise-corrupted dataset $\mathcal{D}$. A representation of such over-approximation is shown in Figure 3.1. The resulting set-valued over-approximation $\mathcal{F} : Z \rightrightarrows \overline{\mathbb{R}}^m$ is defined, componentwise, by its lower and upper bounds as follows [1] [60]:

$$\begin{aligned}\overline{\mathcal{F}}^i(\mathbf{z}) =& \min_{k\in\mathbb{K}} \left(y_k^i + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T \cdot (\mathbf{z}_k - \mathbf{z}) \right), \\ \underline{\mathcal{F}}^i(\mathbf{z}) =& \max_{k\in\mathbb{K}} \left(y_k^i + \underline{d}^i - \overline{d}^i + \underline{\mathbf{m}}_i{}^T \cdot (\mathbf{z}_k - \mathbf{z}) \right),\end{aligned} \tag{3.3}$$

where $\overline{\mathcal{F}}^i(\mathbf{z})$ and $\underline{\mathcal{F}}^i(\mathbf{z})$ are the i-th components of the upper and lower bounds of the set-valued map $\mathcal{F}$, respectively. The vectors $\overline{\mathbf{m}}_i$ and $\underline{\mathbf{m}}_i$ are function of $\mathbf{z}$ and $\mathbf{z}_k$ and are defined for all $i \in \{1, \ldots, m\}$ by:

$$\overline{m}_i^j(\mathbf{z}_k, \mathbf{z}) = \begin{cases} \underline{\gamma}_{ij} & \text{if } z_k^j < z^j, \\ \overline{\gamma}_{ij} & \text{if } z_k^j > z^j. \end{cases} \quad \underline{m}_i^j(\mathbf{z}_k, \mathbf{z}) = \begin{cases} \overline{\gamma}_{ij} & \text{if } z_k^j < z^j, \\ \underline{\gamma}_{ij} & \text{if } z_k^j > z^j. \end{cases} \tag{3.4}$$

With abuse of notation, where there is no ambiguity, we will write $\overline{\mathbf{m}}_i$ and $\underline{\mathbf{m}}_i$ instead of $\overline{\mathbf{m}}_i(\mathbf{z}_k, \mathbf{z})$ and $\underline{\mathbf{m}}_i(\mathbf{z}_k, \mathbf{z})$, respectively.

One can understand Equation (3.3) as follows: To find the over-approximation of the unknown map F at a given point $\mathbf{z}$, we iterate over all the data points $\mathbf{z}_k$. From each point, we calculate the upper and lower bounds of the growth cone of the unknown map F originated at the point $\mathbf{z}_k$, and defined using the bounds on the derivatives of the unknown map F.

Definition 8. *Let $\mathbf{z}, \mathbf{z}' \in Z$, $\mathbf{y} \in \mathbb{R}^m$, and $\underline{\mathbf{d}}, \overline{\mathbf{d}} \in \mathbb{R}^m$, such that $\mathbf{y} \in f(\mathbf{z}) + [\underline{\mathbf{d}}, \overline{\mathbf{d}}]$. The function f defined in Equation (3.1) is differentiable on Z. For all $i \in \{1, \ldots, m\}$, let $\underline{\mathbf{m}}_i(\mathbf{z}, \mathbf{z}'), \overline{\mathbf{m}}_i(\mathbf{z}, \mathbf{z}') \in \mathbb{R}^n$ be defined as in Equation (3.4). A growth cone of the unknown map F with an apex at the point $\mathbf{z}'$ is defined componentwise by its lower and upper bounds as follows:*

$$\begin{aligned}\overline{\mathcal{C}}^i_{\mathbf{z},\mathbf{y}}(\mathbf{z}') =& y^i + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T \cdot (\mathbf{z} - \mathbf{z}'), \\ \underline{\mathcal{C}}^i_{\mathbf{z},\mathbf{y}}(\mathbf{z}') =& y^i + \underline{d}^i - \overline{d}^i + \underline{\mathbf{m}}_i{}^T \cdot (\mathbf{z} - \mathbf{z}').\end{aligned} \tag{3.5}$$

Those growth cones are constructed in such a way as to contain the unknown map F. In other words, it over-approximates the unknown map.

Proposition 7. *Let $k \in \mathbb{K}$ such that $(\mathbf{z}_k, \mathbf{y}_k) \in \mathcal{D}$, The cone $\mathcal{C}_{\mathbf{z}_k,\mathbf{y}_k} : Z \rightrightarrows \overline{\mathbb{R}}^m$ defined in Equation (3.5) over-approximates the unknown map F.*

[1] The original set membership approach uses Lipschitz constants but it is straightforward to extend the approach to the case of bounded derivative functions.

Proof. Let us consider a point $\mathbf{z} \in Z$, and $\mathbf{y} \in F(\mathbf{z})$, For all $i \in \{1, \ldots, m\}$, we have that $y^i \in f^i(\mathbf{z}) + [\underline{d}^i, \overline{d}^i]$. We also have that $y^i_k \in f^i(\mathbf{z}_k) + [\underline{d}^i, \overline{d}^i]$. Due to the fact that f is differentiable on Z, and has bounded derivatives, we have that for all $i \in \{1, \ldots, m\}$,

$$f^i(\mathbf{z}) - f^i(\mathbf{z}_k) \in [\underline{\mathbf{m}}_i^T \cdot (\mathbf{z} - \mathbf{z}_k), \overline{\mathbf{m}}_i^T \cdot (\mathbf{z} - \mathbf{z}_k)],$$

which implies that for all $i \in \{1, \ldots, m\}$,

$$f^i(\mathbf{z}) \leq f^i(\mathbf{z}_k) + \overline{\mathbf{m}}_i^T \cdot (\mathbf{z} - \mathbf{z}_k),$$

hence, for all $i \in \{1, \ldots, m\}$,

$$\begin{aligned} y^i &\leq f^i(\mathbf{z}) + \overline{d}^i \leq f^i(\mathbf{z}_k) + \overline{\mathbf{m}}_i^T \cdot (\mathbf{z} - \mathbf{z}_k) + \overline{d}^i \\ &\leq y^i_k + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T \cdot (\mathbf{z} - \mathbf{z}_k) \\ &\leq \overline{\mathcal{C}}^i_{\mathbf{z}_k, \mathbf{y}_k}(\mathbf{z}). \end{aligned}$$

Similarly, we can show that for all $i \in \{1, \ldots, m\}$,

$$\underline{\mathcal{C}}^i_{\mathbf{z}_k, \mathbf{y}_k}(\mathbf{z}) \leq y^i.$$

Therefore, the cone $\mathcal{C}_{\mathbf{z}_k, \mathbf{y}_k} : Z \rightrightarrows \overline{\mathbb{R}}^m$ defined in Equation (3.5) over-approximates the unknown map F.

□

The next step in calculating the over-approximation, according to the set membership approach, is to take the minimum of the upper bounds and the maximum of the lower bounds of all the growth cones. As each growth cone over-approximates the unknown map F, the resulting set-valued map $\mathcal{F}$ over-approximates the unknown map F. Moreover, the resulting set-valued map $\mathcal{F}$ is the least conservative over-approximation of the unknown map F.

Proposition 8 (Theorem 2 [60]). *Let $\mathcal{F}$ be the set-valued map defined in Equation (3.3). Let $\mathcal{F}'$ be an over-approximation of the map F, such that for all $\mathbf{z} \in Z$,*

- *$\mathcal{F}'(\mathbf{z}) = \tilde{f}(z) + D$, where $\tilde{f}$ is differentiable on Z and has the same bounded derivatives as f,*
- *For all $k \in \mathbb{K}$, $\mathbf{y}_k \in \mathcal{F}'(\mathbf{z}_k)$.*

Then for all $\mathbf{z} \in Z$, $\mathcal{F}(\mathbf{z}) \subseteq \mathcal{F}'(\mathbf{z})$.

Finding this least conservative over-approximation is computationally expensive. It requires visiting all the data points $\mathbf{z}_k$ to find the value of the map, meaning the complexity of calculating the over-approximation at a given point is $\mathcal{O}(|\mathbb{K}|)$. Instead, we will focus on finding over-approximations that are easy to

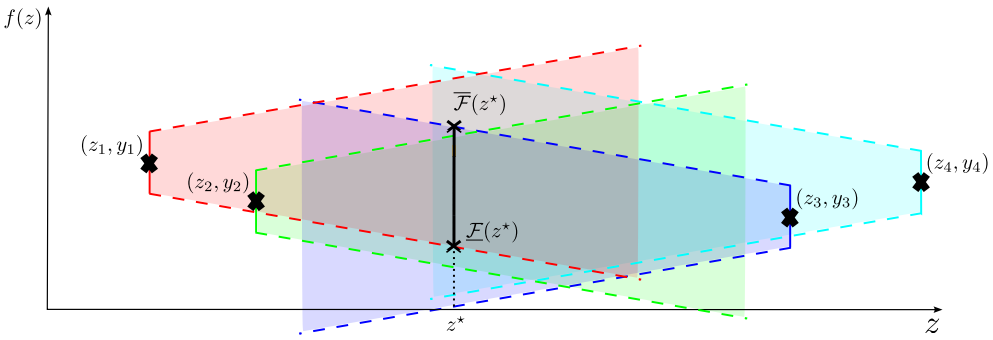


Figure 3.2: The set membership approach over-approximation. The colored regions are the growth cones.

compute and store. For that reason, we will consider interval-valued piecewise over-approximations on a fixed partition of the domain Z. Similar to the case of monotone maps, we will use the same fixed partition $(R_q)_{q\in Q}$ of the domain Z defined in Section 2.2.1, and the associated quantization function ϕ will be used. These over-approximations can be computed offline. Once calculated, they can be used to over-approximate the unknown map F at any point in the domain Z in constant time. Also, there is no need to store the data set $\mathcal{D}$ while online, and it is safe to assume that the number of data points will vastly exceed the number of partitions in practice. This means that the over-approximation will require less memory than storing the data set $\mathcal{D}$. The trade-off is that the over-approximation will be more conservative than the one calculated using the set membership approach.

3.1.1 . Over-approximating by transforming the data into a monotone case

We now show how the approach described in the previous chapter can be adapted to compute data-driven over-approximations for systems where the unknown dynamics f is not necessarily monotone but has bounded derivatives with known upper and lower bounds. Our construction is inspired by the approach presented in [75] for computing decomposition functions of mixed-monotone functions.

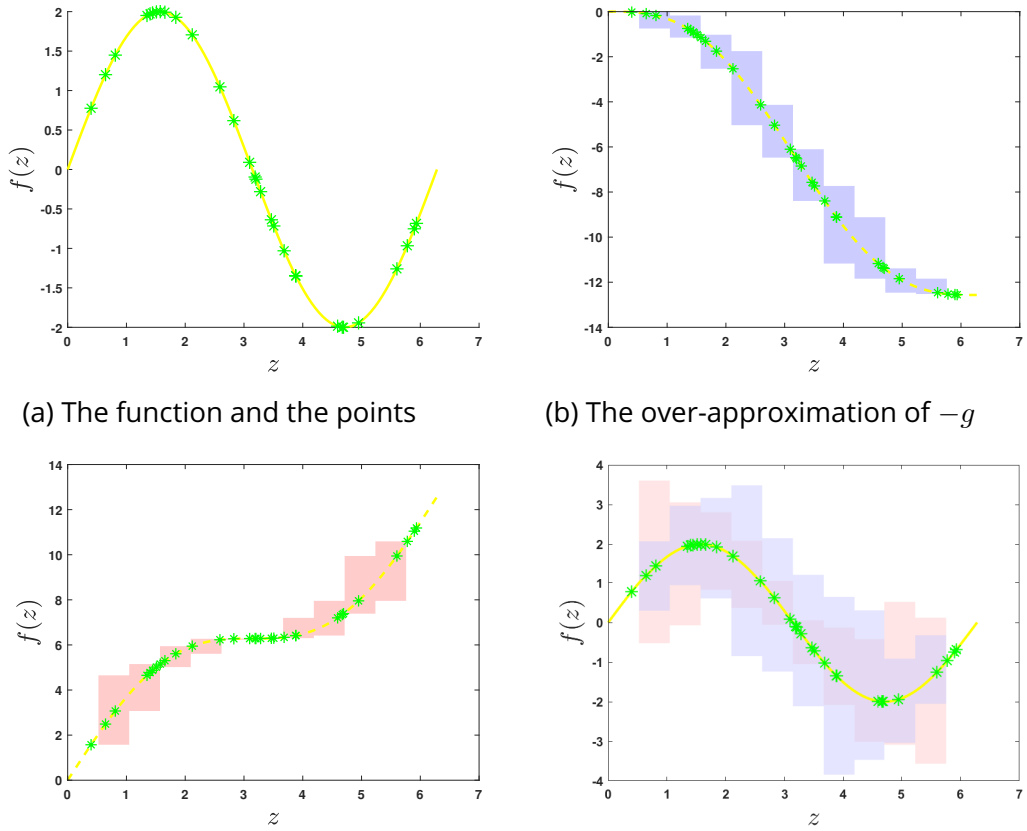
The following example illustrates the proposed approach.

Example 4

Let us consider the sinusoidal function $f : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ defined for all $z \in \overline{\mathbb{R}}$ by:

$$f(z) = \sin(z).$$

We have that $\underline{\gamma} = -1 \leq \frac{\partial f}{\partial z} \leq 1 = \overline{\gamma}$. The two functions $h, g : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ defined



(a) The function and the points

(b) The over-approximation of $-g$

(c) The over-approximation of h

(d) The over-approximation of f

Figure 3.3: The four figures show the over-approximation of the function $f(z) = sin(z)$ on the interval $[0, 2\pi]$. The function f is sampled at 30 points $z_1, \ldots, z_{30}$. Using the upper and lower bounds on the derivative of f, auxiliary data sets $\mathcal{D}_h$ and $\mathcal{D}_g$ are generated. The two auxiliary data sets $\mathcal{D}_h$ and $\mathcal{D}_g$ can be seen as generated by the monotone functions h and g, respectively. After finding the over-approximation of h and g, we can compute the over-approximation of f on the interval $[0, 2\pi]$.

for all $z \in \overline{\mathbb{R}}$ by:

$$h(z) = f(z) - \underline{\gamma} \cdot z = \sin(z) + z,$$
$$g(z) = \overline{\gamma} \cdot z - f(z) = z - \sin(z),$$

are monotonically increasing.
Now let us consider that a data set $\mathcal{D}$ is generated by sampling f at 30 points $z_1, \ldots, z_{30}$ in the domain $Z = [0, 2\pi]$. If we computed the two auxiliary data sets $\mathcal{D}_h$ and $\mathcal{D}_g$ as follows:

$$\mathcal{D}_h = \{(z_k, y_k^-) \mid y_k^- = y_k + z_k,\ k \in \{1, \ldots, 30\}\},$$
$$\mathcal{D}_g = \{(z_k, y_k^+) \mid y_k^+ = z_k - y_k,\ k \in \{1, \ldots, 30\}\}.$$

The two auxiliary data sets $\mathcal{D}_h$ and $\mathcal{D}_g$ can be seen as generated by the functions h and g, respectively. Because h and g are monotone, we can use the approach presented in chapter 2 to compute the two simulating maps S_g and S_h of the data $\mathcal{D}_g$ and $\mathcal{D}_h$, respectively. This is shown in Figures 3.3b and 3.3c. In this example, the interval $[0, 2\pi]$ is partitioned into 13 cells $R_1, \ldots, R_{13}$. Then, we can compute the over-approximation of f on a given cell $R_i = [\underline{z}_{R_i}, \overline{z}_{R_i}], i \in \{2, \ldots, 12\}$ as follows: For all $z \in R_i$, we have:

$$f(z) \in [\underline{\gamma}\ \overline{z}_{R_i} + \underline{h}_{R_i}, \underline{\gamma}\ \underline{z}_{R_i} + \overline{h}_{R_i}],$$
$$f(z) \in [\overline{\gamma}\ \underline{z}_{R_i} - \overline{g}_{R_i}, \overline{\gamma}\ \overline{z}_{R_i} - \underline{g}_{R_i}].$$

Where $\underline{h}_{R_i}, \overline{h}_{R_i}, \underline{g}_{R_i}, \overline{g}_{R_i}$ are the lower and upper bounds of the functions h and g on the cell R_i, respectively. Therefore, we can compute the over-approximation of f on the cell R_i as follows:

$$\begin{aligned} f(z) &\in [\underline{\gamma}\ \overline{z}_{R_i} + \underline{h}_{R_i}, \underline{\gamma}\ \underline{z}_{R_i} + \overline{h}_{R_i}] \cap [\overline{\gamma}\ \underline{z}_{R_i} - \overline{g}_{R_i}, \overline{\gamma}\ \overline{z}_{R_i} - \underline{g}_{R_i}] \\ &= [\max(\underline{\gamma}\ \overline{z}_{R_i} + \underline{h}_{R_i}, \overline{\gamma}\ \underline{z}_{R_i} - \overline{g}_{R_i}), \min(\underline{\gamma}\ \underline{z}_{R_i} + \overline{h}_{R_i}, \overline{\gamma}\ \overline{z}_{R_i} - \underline{g}_{R_i})]. \end{aligned}$$

This is shown in Figure 3.3d.

Now, let us show how we find the over-approximation in the more general n-dimensional case. We start by introducing the auxiliary matrices $A^-, A^+ \in \mathbb{R}^{m \times n}$, where for all $i \in \{1, \ldots, m\}$, for all $j \in \{1, \ldots, n\}$

$$A_{ij}^- = \begin{cases} \underline{\gamma}_{ij} & \text{if } \underline{\gamma}_{ij} < 0, \\ 0 & \text{otherwise,} \end{cases} \qquad A_{ij}^+ = \begin{cases} \overline{\gamma}_{ij} & \text{if } \overline{\gamma}_{ij} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Then, let the functions $f^-, f^+ : Z \to \overline{\mathbb{R}}^m$ be defined for all $\mathbf{z} \in Z$ by:

$$f^-(\mathbf{z}) = f(\mathbf{z}) - A^-\mathbf{z},$$
$$f^+(\mathbf{z}) = A^+\mathbf{z} - f(\mathbf{z}).$$

Let us remark that f^-, f^+ are unknown but monotone since it can be readily checked that all their partial derivatives are nonnegative. Given the data set $\mathcal{D}$ introduced in (3.2), we can define two auxiliary data sets:

$$\mathcal{D}^- = \{(\mathbf{z}_k, \mathbf{y}_k^-) |\ \mathbf{y}_k^- = \mathbf{y}_k - A^-\mathbf{z}_k,\ k \in \mathbb{K}\},$$
$$\mathcal{D}^+ = \{(\mathbf{z}_k, \mathbf{y}_k^+) |\ \mathbf{y}_k^+ = A^+\mathbf{z}_k - \mathbf{y}_k,\ k \in \mathbb{K}\}.$$

We then use the approach presented in Chapter 2 to compute simulating maps $\sigma_-, \sigma_+ : Q \rightrightarrows \overline{\mathbb{R}}^m$ of the data $\mathcal{D}^-$ and $\mathcal{D}^+$, respectively.

Proposition 9. *Let $\sigma_- \circ \phi \in \mathbb{S}_{\mathcal{D}^-}$ and $\sigma_+ \circ \phi \in \mathbb{S}_{\mathcal{D}^+}$, then let $S : Z \rightrightarrows \overline{\mathbb{R}}^m$ be given for all $\mathbf{z} \in Z$ by:*

$$S(\mathbf{z}) = \left(A^-\mathbf{z} + \sigma_- \circ \phi(\mathbf{z})\right) \cap \left(A^+\mathbf{z} - \sigma_+ \circ \phi(\mathbf{z})\right).$$

Then, it holds:

$$\forall \mathbf{z} \in Z,\ f(\mathbf{z}) + D \subseteq S(\mathbf{z}).$$

Proof. Let $\mathbf{z} \in Z$, from $\sigma_- \circ \phi \in \mathbb{S}_{\mathcal{D}^-}$, we have

$$f^-(\mathbf{z}) + D \subseteq \sigma_- \circ \phi(\mathbf{z}).$$

Since $f(\mathbf{z}) = f^-(\mathbf{z}) + A^-\mathbf{z}$, we get

$$f(\mathbf{z}) + D \subseteq A^-\mathbf{z} + \sigma_- \circ \phi(\mathbf{z}).$$

Similarly, we can show that

$$f(\mathbf{z}) + D \subseteq A^+\mathbf{z} - \sigma_+ \circ \phi(\mathbf{z}),$$

which leads to the result of the proposition. □

To find the interval hall of S, we can solve the following optimization problem for each cell $R_\mathbf{q}$ of the partition: to reach the upper bound of S on $R_\mathbf{q}$,

$$\overline{\sigma}^i(\mathbf{q}) = \begin{cases} \max\limits_{\mathbf{z} \in R_\mathbf{q}, c \in \mathbb{R}} & c \\ \text{s.t.} & c \leq A_i^-\mathbf{z} + \overline{\sigma}_-^i \circ \phi(\mathbf{z}) \\ & c \leq A_i^+\mathbf{z} - \underline{\sigma}_+^i \circ \phi(\mathbf{z}) \end{cases} \tag{3.6}$$

where A_i^- and A_i^+ are the i-th rows of A^- and A^+, respectively. Finding the lower bound of S on R_i can be done similarly.

$$\underline{\sigma}^i(\mathbf{q}) = \begin{cases} \min\limits_{\mathbf{z}\in R_{\mathbf{q}}, c\in\mathbb{R}} & c \\ \text{s.t.} & c \geq A_i^- \mathbf{z} + \underline{\sigma}^i_- \circ \phi(\mathbf{z}) \\ & c \geq A_i^+ \mathbf{z} - \overline{\sigma}^i_+ \circ \phi(\mathbf{z}) \end{cases} \tag{3.7}$$

Program (3.7) and (3.6) are linear programs and can be solved efficiently. Other option can be to compute the following interval-valued map instead:

$$\underline{\sigma}^i(\mathbf{q}) = \min(A^- \ \overline{\mathbf{z}}_{\mathbf{q}} + \underline{\sigma}_-(\mathbf{q}), A^+ \ \underline{\mathbf{z}}_{\mathbf{q}} - \overline{\sigma}_+(\mathbf{q})) \tag{3.8}$$

$$\overline{\sigma}^i(\mathbf{q}) = \max(A^- \ \underline{\mathbf{z}}_{\mathbf{q}} + \overline{\sigma}_-(\mathbf{q}), A^+ \ \overline{\mathbf{z}}_{\mathbf{q}} - \underline{\sigma}_+(\mathbf{q})). \tag{3.9}$$

This is what was used in Example 3.1.1.

Now, let us give some remarks on the calculated over-approximation. Proposition 9 shows that the map S is an over-approximation of the data $\mathcal{D}$. This is done using only two auxiliary matrices A^- and A^+, where only the lower bounds are used to build A^- and only the upper bounds are used to build A^+. But that is not only two options to build A; there are $2^{m\times n}$ options. For example, in the case of a function $f : \overline{\mathbb{R}}^2 \to \overline{\mathbb{R}}$, we can define the following matrices:

$$A_1 = \begin{pmatrix} \min(\underline{\gamma}_1, 0) & \min(\underline{\gamma}_2, 0) \end{pmatrix}, \ A_2 = \begin{pmatrix} \min(\underline{\gamma}_1, 0) & \max(\overline{\gamma}_2, 0) \end{pmatrix},$$
$$A_3 = \begin{pmatrix} \max(\overline{\gamma}_2, 0) & \min(\underline{\gamma}_1, 0) \end{pmatrix}, \ A_4 = \begin{pmatrix} \max(\overline{\gamma}_2, 0) & \max(\overline{\gamma}_2, 0) \end{pmatrix}.$$

All can be used to find auxiliary data sets that can be seen as generated from a monotone function. Let us denote by $\mathcal{A}$ the set of all the matrices $A^l, l \in \{1, \ldots, n \times m\}$ that can be built using the bounds on the derivatives of f.

$$\mathcal{A} = \left\{ A^l \in \mathbb{R}^{m\times n} \;\middle|\; \begin{array}{c} \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}, \\ A^l_{ij} \in \{\min(\underline{\gamma}_{ij}, 0), \max(\overline{\gamma}_{ij}, 0)\} \end{array} \right\}$$

Then, we can define the set of all the simulating maps S^l of the data $\mathcal{D}^l$ for all $A^l \in \mathcal{A}$, to find a less conservative over-approximation than the one calculated in Proposition 9.

Remark 5. *When using matrices $A^l \in \mathcal{A}$, other than A^- and A^+ (the ones built using only the lower and upper bounds on the derivatives of f, respectively), A transformation of the data $\mathcal{D}$ is needed to find the auxiliary data sets $\mathcal{D}^l$. The transformation is similar to the transformation used in Section 2.4, for all $i \in \{1, \ldots, m\}$:*

$$z'^j = \begin{cases} z^j & \text{if } A^l_{i,j} \geq 0 \\ -z^j & \text{if } A^l_{i,j} < 0 \end{cases}$$

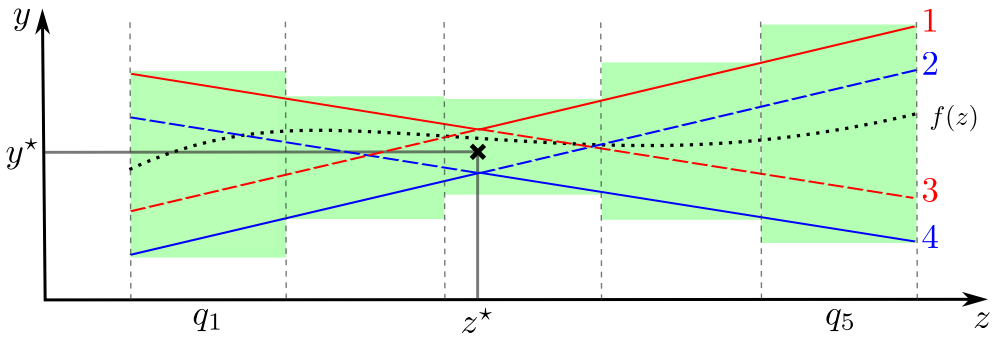


Figure 3.4: The figure shows the over-approximation of the map F on the cells $R_{\mathbf{q}}$. The over-approximation is done by finding the value of the growth cone at the vertices of the cells. Line 1 has the equation $y = \overline{\gamma} z + y^\star - \overline{\gamma} z^\star + \overline{d}$. Line 2 has the equation $y = \overline{\gamma} z + y^\star - \overline{\gamma} z^\star + \underline{d}$. Line 3 has the equation $y = \underline{\gamma} z + y^\star - \underline{\gamma} z^\star + \overline{d}$. Line 4 has the equation $y = \underline{\gamma} z + y^\star - \underline{\gamma} z^\star + \underline{d}$.

3.1.2 . Over-approximation by finding the value of growth cones at the vertices of the cells

In this section, we present the second approach to find an over-approximation of the map (3.1). The approach depends on the fact that for any $k \in \mathbb{K}$, one of the values of the growth cone $\mathcal{C}_{\mathbf{x}_k, \mathbf{y}_k}$ at the vertices of the cells in the partition will always be greater than or equal to the value of the growth cone at any other point in the cell. Also, there will always be a value of the growth cone at the vertices of the cells in the partition that is less than or equal to the value of the growth cone at any other point in the cell.

Let us formally define the over-approximation. We denote the set of vertices of the cell $R_{\mathbf{q}} = [\underline{\mathbf{z}}_{\mathbf{q}}, \overline{\mathbf{z}}_{\mathbf{q}})$ by $V_{\mathbf{q}}$.

$$V_{\mathbf{q}} = \prod_{i=1}^{n} \{\underline{z}^i_{\mathbf{q}}, \overline{z}^i_{\mathbf{q}}\}.$$

Let $\sigma : Q \rightrightarrows \overline{\mathbb{R}}^m$ be an interval-valued map $\sigma(\mathbf{q}) = [\underline{\sigma}(\mathbf{q}), \overline{\sigma}(\mathbf{q})]$. The upper and lower values are defined by their components $\underline{\sigma}^i, \overline{\sigma}^i$, $i \in \{1, \ldots, m\}$ as follows:

$$\overline{\sigma}^i(\mathbf{q}) = \min_{k \in \mathbb{K}} \left(\max_{\mathbf{v} \in V_{\mathbf{q}}} \left(\overline{\mathcal{C}}^i_{\mathbf{z}_k, \mathbf{y}_k}(\mathbf{v}) \right) \right) \tag{3.10}$$

$$\underline{\sigma}^i(\mathbf{q}) = \max_{k \in \mathbb{K}} \left(\min_{\mathbf{v} \in V_{\mathbf{q}}} \left(\underline{\mathcal{C}}^i_{\mathbf{z}_k, \mathbf{y}_k}(\mathbf{v}) \right) \right) \tag{3.11}$$

The following proposition shows that the piecewise map σ over-approximates the unknown map F.

Proposition 10. *The map $S = \sigma \circ \phi : Z \rightrightarrows \mathbb{R}^m$ over-approximates the the map F.*

Proof. To prove Proposition 10, let us assume it is wrong and try to find a contradiction. Let $\mathbf{z}^\dagger \in Z$, be a point, such that, there exists $\mathbf{y}^\dagger \in f(\mathbf{z}^\dagger) + D$ but $\mathbf{y}^\dagger \notin \sigma \circ \phi(\mathbf{z}^\dagger)$. Because σ is interval-valued we have, $\mathbf{y}^\dagger \not\preceq \overline{\sigma}(\phi(\mathbf{z}^\dagger))$ or $\mathbf{y}^\dagger \not\succeq \underline{\sigma}(\phi(\mathbf{z}^\dagger))$. We will study the case of $\mathbf{y}^\dagger \not\preceq \overline{\sigma}(\phi(\mathbf{z}^\dagger))$, the second case can be dealt with similarly. Let $\mathbf{q} = \phi(\mathbf{z}^\dagger)$, and

$$\begin{aligned}\overline{\sigma}^i(\mathbf{q}) &= \min_{k\in\mathbb{K}} \left(\max_{\mathbf{v}\in V_\mathbf{q}} \left(y^i_k + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_k, \mathbf{v}) \cdot (\mathbf{z}_k - \mathbf{v}) \right) \right) \\ &= y^i_{k^\star} + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{v}^\star) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}^\star)\end{aligned}$$

where, $\mathbf{v}^\star \in V_{\phi(\mathbf{z}^\dagger)}$ is the maximizer of the expression (3.10) and $k^\star \in \mathbb{K}$ is the minimizer. If $\mathbf{y}^\dagger \not\preceq \overline{\sigma}(\phi(\mathbf{z}^\dagger))$ then, there exists $i \in \{1, \ldots, m\}$ such that

$$y^{\dagger i} > y^i_{k^\star} + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{v}^\star) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}^\star) \tag{3.12}$$

On the other hand, we have $\mathbf{y}^\dagger \in f(\mathbf{z}^\dagger) + D$, which implies,

$$y^{\dagger i} \leq y^i_{k^\star} + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) \cdot (\mathbf{z}_{k^\star} - \mathbf{z}^\dagger) \tag{3.13}$$

Due to the conditions on the derivatives of f, from (3.12) and (3.13) we have

$$\overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) \cdot (\mathbf{z}_{k^\star} - \mathbf{z}^\dagger) > \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{v}^\star) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}^\star) \tag{3.14}$$

Let $\mathbf{v}^\circ \in V_{\phi(\mathbf{z}^\dagger)}$ be such that, for all $j \in \{1, \ldots, n\}$,

$$v^{\circ j} = \begin{cases} \underline{z}^j_\mathbf{q} & \text{if } \overline{m}^j_i(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) < 0, \\ \overline{z}^j_\mathbf{q} & \text{if } \overline{m}^j_i(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) \geq 0. \end{cases}$$

By construction, we have that

$$\overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}^\circ) \geq \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) \cdot (\mathbf{z}_{k^\star} - \mathbf{z}^\dagger) \tag{3.15}$$

But, we also have that

$$\overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}^\circ) \leq \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{v}^\circ) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}^\circ) \tag{3.16}$$

To see why the inequality 3.16 is true, let us recall the componentwise definition of $\overline{\mathbf{m}}_i$:

$$\overline{m}^j_i(\mathbf{z}_k, \mathbf{z}) = \begin{cases} \underline{\gamma}_{ij} & \text{if } z^j_k < z^j, \\ \overline{\gamma}_{ij} & \text{if } z^j_k > z^j. \end{cases}$$

Which implies, for all $j \in \{1, \ldots, n\}$, we have

$$\overline{m}^j_i(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) \cdot (z_{k^\star,j} - v^{\circ,j}) \leq \overline{m}^j_i(\mathbf{z}_{k^\star}, \mathbf{v}^\star) \cdot (z_{k^\star,j} - v^{\circ,j})$$

Hence we have (3.16). By comparing (3.15) and (3.16) we have

$$\overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{v}^\circ) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}^\circ) \geq \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{z}^\dagger) \cdot (\mathbf{z}_{k^\star} - \mathbf{z}^\dagger) \tag{3.17}$$

From (3.14) and (3.17) we reach a contradiction because $\mathbf{v}^\star$ should be the maximizer of the expression (3.10). Therefore, the map $S \circ \phi$ over-approximate the unknown map. □

Remark 6. *Given the partition $(D_{\mathbf{q}})_{\mathbf{q}\in Q}$ of Z. The straightforward algorithm to compute σ is as follows: For each data points in $\mathcal{D}$, we go through all the $\mathbf{q} \in Q$ and determine $\underline{\sigma}$ and $\overline{\sigma}$. Being able to update the map for each data point separately comes from the fact that min (max) behaves like*

$$\min_{k\in\mathbb{K}} \mathcal{S}(k) = \min(\min_{k\in\mathbb{K}\setminus\{k'\}} \mathcal{S}(k), \mathcal{S}(k'))$$

That makes the overall complexity $\mathcal{O}(|\mathbb{K}| \times |Q|)$, i.e. bilinear with respect to the number of data points and to the number of the partition elements.

Efficent calculation of the over-approximation

Similar to the way we calculated the over-approximation in Section 2.3 efficiently, this section is concerned with the efficient calculation of the over-approximation in the case of functions with bounded derivatives. In particular, for the third introduced approach in Section 3.1.2. In the case of monotone maps, it was sufficient to use each data point to update the lower bound of the over-approximation of one cell, and the upper bound of the over-approximation of another cell. Then, we can spread the information of this point to the other cells by only comparing it with neighboring cells. In an increasing fashion for the lower bound, and in a decreasing fashion for the upper bound. Using the same concept of spreading the information, we can envision an efficient algorithm to calculate the over-approximation in the case of functions with bounded derivatives. First, we go through all the data points and for each point, we find the cell that contains the data point. Then, we update the values of over-approximation on the vertices of only this cell, namely, the maximum and minimum allowed values. After updating the values of the over-approximation on the vertices using all the data points, We start a fixed-point iteration to update the values of the over-approximation on the vertices of the cells using the values of the over-approximation on the adjacent vertices. At each iteration, when the value of the over-approximation on a vertex is updated, the information about the maximum and minimum allowed values is spread to the adjacent vertices. The fixed-point iteration stops when the values of the over-approximation on the vertices of the cells do not change. In the worst case, the number of iterations is the time (number of iterations) for the information to spread from a vertex on the boundary of the partition to the opposite vertex on the other side of the partition.

As we are concerned with calculating the over-approximation on the vertices of the cells, and to simplify the description, let us give a distinct notation to those vertices. We will use the notation V to denote the set of indices of vertices of the cells in the partition.

$$V = \{\mathbf{v} \in V_{\mathbf{q}} \mid \mathbf{q} \in Q\},$$

Let us denote the set of cells containing the vertex $\mathbf{v} \in V$ by $Q_{\mathbf{v}}$, and let us denote the set of vertices of the set of cells $Q_{\mathbf{v}}$ by $V_{Q_{\mathbf{v}}}$, which resembles the set of neighboring vertices of the vertex $\mathbf{v}$.

$$Q_{\mathbf{v}} = \{\mathbf{q} \in Q \mid \mathbf{v} \in V_{\mathbf{q}}\},$$
$$V_{Q_{\mathbf{v}}} = \{\mathbf{v} \in V_{\mathbf{q}} \mid \mathbf{q} \in Q_{\mathbf{v}}\},$$

Now, let us first define the functions $\overline{\sigma}_0 : V \to \overline{\mathbb{R}}^m$ and $\underline{\sigma}_0 : V \to \overline{\mathbb{R}}^m$, for all $i \in \{1, \dots, m\}$ as follows:

$$\overline{\sigma}_0^i(\mathbf{v}) = \min\left\{\overline{C}_{\mathbf{z}_k,\mathbf{y}_k}(\mathbf{v}) \mid \mathbf{z}_k \in \mathrm{cl}\ R_{\mathbf{q}}, \mathbf{q} \in Q_{\mathbf{v}}\right\}, \tag{3.18}$$
$$\underline{\sigma}_0^i(\mathbf{v}) = \max\left\{\underline{C}_{\mathbf{z}_k,\mathbf{y}_k}(\mathbf{v}) \mid \mathbf{z}_k \in \mathrm{cl}\ R_{\mathbf{q}}, \mathbf{q} \in Q_{\mathbf{v}}\right\}. \tag{3.19}$$

With an abuse of notation, we write $\mathbf{z}_k \in R_{Q_{\mathbf{V}}}$ to denote that $\mathbf{z}_k \in \mathrm{cl}\ R_{\mathbf{q}}$ for $\mathbf{q} \in Q_{\mathbf{v}}$. Using a similar approach to the one used in Section 2.3, we can compute the values of $\overline{\sigma}_0$ and $\underline{\sigma}_0$ with a complexity $\mathcal{O}(|\mathbb{K}| \times \log(|V|))$.

Let us define $\overline{\sigma}_1 : V \to \overline{\mathbb{R}}^m$ and $\underline{\sigma}_1 : V \to \overline{\mathbb{R}}^m$, that offer the optimal over-approximation of the unknown map F on the vertices of the cells.

$$\forall \mathbf{v} \in V,\ \overline{\sigma}_1(\mathbf{v}) = \overline{\mathcal{F}}(\mathbf{v}), \tag{3.20}$$
$$\forall \mathbf{v} \in V,\ \underline{\sigma}_1(\mathbf{v}) = \underline{\mathcal{F}}(\mathbf{v}). \tag{3.21}$$

Where $\overline{\mathcal{F}}$ and $\underline{\mathcal{F}}$ are defined in (3.3).

Proposition 11. *The map σ_1 defined in* (3.20) *and* (3.21) *satisfies*

$$\overline{\sigma}_1^i(\mathbf{v}) = \min\left\{\overline{\sigma}_0^i(\mathbf{v}), \min_{\mathbf{v}^\dagger \in V_{Q_{\mathbf{v}}}} (\overline{\sigma}_1^i(\mathbf{v}^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v}))\right\}, \tag{3.22}$$
$$\underline{\sigma}_1^i(\mathbf{v}) = \max\left\{\underline{\sigma}_0^i(\mathbf{v}), \max_{\mathbf{v}^\dagger \in V_{Q_{\mathbf{v}}}} (\underline{\sigma}_1^i(\mathbf{v}^\dagger) + \underline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v}))\right\}, \tag{3.23}$$

and it can be computed according to Algorithm 1.

Proof. Let us prove (3.22). The proof for (3.23) is similar.

1. It is straightforward to see that $\overline{\sigma}_1^i(\mathbf{v}) \leq \overline{\sigma}_0^i(\mathbf{v})$. Also, we have for all $\mathbf{v}^\dagger \in V_{Q_{\mathbf{v}}}$,

$$y_k^i + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_k, \mathbf{v}) \cdot (\mathbf{z}_k - \mathbf{v}) \leq y_k^i + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_k, \mathbf{v}^\dagger) \cdot (\mathbf{z}_k - \mathbf{v}^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v})$$

Which implies,

$$\overline{\sigma}_1^i(\mathbf{v}) \leq \overline{\sigma}_1^i(\mathbf{v}^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v}).$$

Hence,

$$\overline{\sigma}_1^i(\mathbf{v}) \leq \min_{\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}} \left(\overline{\sigma}_1^i(\mathbf{v}^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v}) \right).$$

Therefore,

$$\overline{\sigma}_1^i(\mathbf{v}) \leq \min \left\{ \overline{\sigma}_0^i(\mathbf{v}), \min_{\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}} (\overline{\sigma}_1^i(\mathbf{v}^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v})) \right\}.$$

2. Now we want to that the right-hand side of the inequality is less than or equal to $\overline{\sigma}_1^i(\mathbf{v})$. Let $k^\star \in \mathbb{K}$ be such that

$$\overline{\sigma}_1^i(\mathbf{v}) = y_{k^\star}^i + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{v}) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}).$$

Let us consider two cases:

   (a) If $\mathbf{z}_{k^\star} \in R_{Q_\mathbf{v}}$, then $\overline{\sigma}_1^i(\mathbf{v}) \geq \overline{\sigma}_0^i(\mathbf{v})$.

   (b) If $\mathbf{z}_{k^\star} \notin R_{Q_\mathbf{v}}$, then there exists $\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}$ such that $\overline{\mathbf{m}}_i(\mathbf{z}_k, \mathbf{v}) = \overline{\mathbf{m}}_i(\mathbf{v}_k^\dagger, \mathbf{v})$. This is possible because for all $\mathbf{z}_k \in Z \setminus R_{Q_\mathbf{v}}$, there exists $\mathbf{v}_k^\dagger \in V_{Q_\mathbf{v}}$ such that sign$(z_k^j - v^j) =$ sign$(v_k^{\dagger j} - v^j)$ for all $j \in \{1, \ldots, n\}$. Then,

$$\begin{aligned} \overline{\mathbf{m}}_i{}^T(\mathbf{z}_k, \mathbf{v}) \cdot (\mathbf{z}_k - \mathbf{v}) =& \overline{\mathbf{m}}_i{}^T(\mathbf{z}_k, \mathbf{v}_k^\dagger) \cdot (\mathbf{z}_k - \mathbf{v}_k^\dagger) + \\ & \overline{\mathbf{m}}_i{}^T(\mathbf{v}_k^\dagger, \mathbf{v}) \cdot (\mathbf{v}_k^\dagger - \mathbf{v}). \end{aligned}$$

   Which implies,

$$\begin{aligned} \sigma_1^i(\mathbf{v}) &= y_{k^\star}^i + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{v}) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}) \\ &= y_{k^\star}^i + \overline{d}^i - \underline{d}^i + \overline{\mathbf{m}}_i{}^T(\mathbf{z}_{k^\star}, \mathbf{v}_k^\dagger) \cdot (\mathbf{z}_{k^\star} - \mathbf{v}_k^\dagger) + \\ &\quad \overline{\mathbf{m}}_i{}^T(\mathbf{v}_k^\dagger, \mathbf{v}) \cdot (\mathbf{v}_k^\dagger - \mathbf{v}) \\ &\geq \overline{\sigma}_1^i(\mathbf{v}_k^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}_k^\dagger, \mathbf{v}) \cdot (\mathbf{v}_k^\dagger - \mathbf{v}) \end{aligned}$$

   Therefore,

$$\overline{\sigma}_1^i(\mathbf{v}) \geq \min \left\{ \overline{\sigma}_0^i(\mathbf{v}), \min_{\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}} (\overline{\sigma}_1^i(\mathbf{v}^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v})) \right\}.$$

From the above two inequalities, we have that

$$\overline{\sigma}_1^i(\mathbf{v}) = \min \left\{ \overline{\sigma}_0^i(\mathbf{v}), \min_{\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}} (\overline{\sigma}_1^i(\mathbf{v}^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v})) \right\}.$$

□

Algorithm 1 Compute σ_1

$\overline{\sigma}_1 \leftarrow \overline{\sigma}_0$
$\underline{\sigma}_1 \leftarrow \underline{\sigma}_0$
change $\leftarrow$ **true**
while change do
    change $\leftarrow$ **false**
    for $\mathbf{v} \in V$ **do**
        $\overline{\sigma}^i_{old}(\mathbf{v}) \leftarrow \overline{\sigma}^i_1(\mathbf{v})$
        $\underline{\sigma}^i_{old}(\mathbf{v}) \leftarrow \underline{\sigma}^i_1(\mathbf{v})$
        $\overline{\sigma}^i_1(\mathbf{v}) \leftarrow \max_{\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}} (\underline{\sigma}^i_1(\mathbf{v}^\dagger) + \underline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v}))$
        $\underline{\sigma}^i_1(\mathbf{q}^\star) \leftarrow \max_{\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}} (\underline{\sigma}^i_1(\mathbf{v}^\dagger) + \underline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v}))$
        if $\overline{\sigma}^i_{old}(\mathbf{q}^\star) \neq \overline{\sigma}^i_1(\mathbf{q}^\star)$ or $\underline{\sigma}^i_{old}(\mathbf{q}^\star) \neq \underline{\sigma}^i_1(\mathbf{q}^\star)$ **then**
            change $\leftarrow$ **true**
        end if
    end for
end while

The next proposition addresses the complexity of the algorithm to compute $\overline{\sigma}_1$ and $\underline{\sigma}_1$.

Proposition 12. *The functions $\overline{\sigma}_1$ and $\underline{\sigma}_1$ can be computed using a fixed-point algorithm with a complexity $\mathcal{O}(|V| \times (\max_i K^i + 1))$. We recall that K^i is the number of cells of the partition on the i^{th} dimension.*

Proof. Let $\mathbf{v} \in V$. We have

$$\overline{\sigma}^i_1(\mathbf{v}) = \min \left\{ \overline{\sigma}^i_0(\mathbf{v}), \min_{\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}} (\overline{\sigma}^i_1(\mathbf{v}^\dagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\dagger, \mathbf{v}) \cdot (\mathbf{v}^\dagger - \mathbf{v})) \right\}$$

For all $\mathbf{v}^\dagger \in V_{Q_\mathbf{v}}$, we have the same expression for $\overline{\sigma}^i_1(\mathbf{v}^\dagger)$.

$$\overline{\sigma}^i_1(\mathbf{v}^\dagger) = \min \left\{ \overline{\sigma}^i_0(\mathbf{v}^\dagger), \min_{\mathbf{v}^\ddagger \in V_{Q_{\mathbf{v}^\dagger}}} (\overline{\sigma}^i_1(\mathbf{v}^\ddagger) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}^\ddagger, \mathbf{v}^\dagger) \cdot (\mathbf{v}^\ddagger - \mathbf{v}^\dagger)) \right\}$$

We can see that the recursion stops when we reach a vertex on the boundary of the partition. The longest path from a vertex on the boundary of the partition to the opposite vertex on the other side of the partition is $\max_i K^i + 1$. Thus, the complexity of the algorithm is $\mathcal{O}(|V| \times (\max_i K^i + 1))$ because we do the update for all the vertices in V. The proof for $\underline{\sigma}_1$ is similar. □

The map σ_1 defined in (3.20) and (3.21) offers the optimal over-approximation of the unknown map F on the vertices of the cells, but we are interested in the over-approximation of the unknown map F on the entire points of the cells. Let

us recall that for all the vertices $\mathbf{v} \in V$, we have that $F(\mathbf{v}) \subseteq \sigma_1(\mathbf{v})$, meaning that for each cell $\mathbf{q} \in Q$, for all $\mathbf{z} \in R_{\mathbf{q}}$ we have that

$$F(\mathbf{z}) \subseteq \bigcap_{\mathbf{v} \in V_{\mathbf{q}}} \left(\bigcup_{\mathbf{y} \in \sigma_1(\mathbf{v})} C_{\mathbf{v},\mathbf{y}}(\mathbf{z}) \right)$$

. To find the upper bound on the right-hand side of the previous inclusion, we can use the following linear optimization problems:

$$\overline{\sigma}^i(\mathbf{q}) = \begin{cases} \max\limits_{\mathbf{z} \in R_{\mathbf{q}}, c \in \mathbb{R}} & c \\ \text{s.t.} & c \leq \overline{\sigma}_1^i(\mathbf{v}) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}, \mathbf{z}) \cdot (\mathbf{v} - \mathbf{z}) \quad \forall \mathbf{v} \in V_{\mathbf{q}} \end{cases} \tag{3.24}$$

Similarly, the lower bound can be found using the following linear optimization problems:

$$\underline{\sigma}^i(\mathbf{q}) = \begin{cases} \min\limits_{\mathbf{z} \in R_{\mathbf{q}}, c \in \mathbb{R}} & c \\ \text{s.t.} & c \geq \underline{\sigma}_1^i(\mathbf{v}) + \underline{\mathbf{m}}_i{}^T(\mathbf{v}, \mathbf{z}) \cdot (\mathbf{v} - \mathbf{z}) \quad \forall \mathbf{v} \in V_{\mathbf{q}} \end{cases} \tag{3.25}$$

Another way to find the over-approximation of the unknown map F on the entire points of the cells is to find the bounds on the value of those growth cones at the vertices of the cells.

$$\overline{\sigma}^i(\mathbf{q}) = \min_{\mathbf{v} \in V_{\mathbf{q}}} \max_{\mathbf{v}^\star \in V_{\mathbf{q}}} \left(\overline{\sigma}_1^i(\mathbf{v}) + \overline{\mathbf{m}}_i{}^T(\mathbf{v}, \mathbf{v}^\star) \cdot (\mathbf{v} - \mathbf{v}^\star) \right) \tag{3.26}$$

for the upper bound, and

$$\underline{\sigma}^i(\mathbf{q}) = \max_{\mathbf{v} \in V_{\mathbf{q}}} \min_{\mathbf{v}^\star \in V_{\mathbf{q}}} \left(\underline{\sigma}_1^i(\mathbf{v}) + \underline{\mathbf{m}}_i{}^T(\mathbf{v}, \mathbf{v}^\star) \cdot (\mathbf{v} - \mathbf{v}^\star) \right) \tag{3.27}$$

for the lower bound. In both cases of finding the over-approximation, we should visite all the cells in the partition, Making the overall complexity of the algorithm $\mathcal{O}(|\mathbb{K}| \times \log(|V|) + |V| \times (\max_i K^i + 1))$. Calculating the over-approximation according to (3.26) and (3.27) does not require implementing a linear optimization problem, hence, it is faster than calculating the over-approximation according to (3.24) and (3.25). However, the over-approximation calculated according to (3.24) and (3.25) is tighter than the one calculated according to (3.26) and (3.27).

3.2 . Updating the over-approximation locally

In the previous section, we introduced algorithms to build an over-approximation of an unknown function. The algorithm described in Section 3.1.2 can be computed online (in light of remark 6), i.e., we can build the over-approximation point by point. But for each data point, we need to update the over-approximation on all the cells of the partition. Making the algorithm computationally prohibitive

if we want to run the system online and collect new data to update the over-approximation, especially if the partition contains a large number of cells. In this section, we propose another interval-valued over-approximation, which we update locally. When dealing with a data point, we find to which cell the point belongs, and then we update only the neighboring cells inside a predefined window the same way we would have done when calculating σ in the second approach. The new over-approximation will introduce more conservatism, but it will be faster to compute.

Let $Q_w(\mathbf{q}) = \{\mathbf{q}^\dagger \in Q \mid max(\mathbf{q} - \underline{\mathbf{q}}_w, \mathbf{0}_n) \preceq \mathbf{q}^\dagger \preceq \min(\mathbf{q} + \overline{\mathbf{q}}_w, K),\ \underline{\mathbf{q}}_w, \overline{\mathbf{q}}_w \in Q\}$ define the window where we want to update the over-approximation.

We define the locally-computed over-approximation $F_w : Z \rightrightarrows \mathbb{R}^{n_x}$ as follows: For all $i \in \{1, \dots, n_x\}$

$$\overline{\sigma}_w^i(\mathbf{q}) = \min_{k \in \mathcal{K}(\mathbf{r})} \left(\max_{\mathbf{v}_l \in V_\mathbf{r}} \left(y_k^j + \overline{w}^j - \underline{w}^j + \overline{\mathbf{m}}_i \cdot (\mathbf{z}_k - \mathbf{v}_l) \right) \right) \tag{3.28}$$

$$\underline{\sigma}_w^i(\mathbf{q}) = \max_{k \in \mathcal{K}(\mathbf{r})} \left(\max_{\mathbf{v}_l \in V_\mathbf{r}} \left(y_k^j + \underline{w}^j - \overline{w}^j + \underline{\mathbf{m}}_i \cdot (\mathbf{z}_k - \mathbf{v}_l) \right) \right) \tag{3.29}$$

$$\mathcal{K}(\mathbf{q}) = \{k \in \mathbb{K} \mid \mathbf{z}_k \in Z_{\mathbf{q}^\star}, \mathbf{q}^\star \in Q_w(\mathbf{q})\}$$

Proposition 13. *The map $S_w = \sigma_w \circ \phi : Z \rightrightarrows \mathbb{R}^{n_x}$ over-approximate the map F.*

Proof. We have for all $\mathbf{q} \in Q$, $\mathcal{K}(\mathbf{q}) \subseteq \mathbb{K}$. Hence, for all $\mathbf{q} \in Q$ the map σ defined in (3.10) and (3.11) is $\sigma(\mathbf{q}) \subseteq \sigma_w(\mathbf{q})$.
Map S is an over-approximation of the map F according to Proposition 10. Therefore, S_w is also an over-approximation □

The map σ_w can be computed online. For each data point, we find the cell that contains the data point. Then, we update the values of the over-approximation on the vertices of all the cells in the window. The complexity of updating the map σ_w given a new data point is $\mathcal{O}(|Q_w(\mathbf{q})|)$.

3.3 . Finding the derivative bounds from data

In the previous sections, we made several assumptions on the unknown map F: bounds on the disturbances and bounds on the partial derivatives of the function f. It is sometimes possible to derive such bounds a priori. For instance, in some cases, the monotonicity of a function can be inferred from principles of physics. However, there are many situations where these bounds also need to be inferred from data. In this section, we briefly present an approach to compute bounds on the disturbances and on the partial derivatives of the function f, directly from data, with probabilistic guarantees. Our approach is based on the scenario approach [18], a data-driven approach to robust convex optimization.

Let us consider a map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$ where $Z \subseteq \overline{\mathbb{R}}^n$ and such that for all $\mathbf{z} \in Z$, $F(\mathbf{z}) = f(\mathbf{z}) + D$ where $f : Z \to \overline{\mathbb{R}}^m$ is a differentiable function with unknown lower and upper bounds $\underline{a}_{ij}, \overline{a}_{ij}$ on its partial derivatives $\frac{\partial f^i}{\partial z^j}$ and D, $D \subseteq \mathbb{R}^m$ is a bounded interval of disturbances with unknown lower and upper bounds $\underline{\mathbf{d}}, \overline{\mathbf{d}}$. Let us remark that without loss of generality, it is always possible to choose $\underline{\mathbf{d}} = -\overline{\mathbf{d}}$.

For all $i \in \{1, \ldots, m\}$, let $\underline{\mathbf{a}}_i = (\underline{a}_{i1}, \ldots, \underline{a}_{in})$ and $\overline{\mathbf{a}}_i = (\overline{a}_{i1}, \ldots, \overline{a}_{in})$. Some bounds are consistent with our assumptions if and only if they satisfy:

$$\begin{aligned} &\forall \mathbf{z}, \mathbf{z}' \in Z,\ \forall \mathbf{y} \in F(\mathbf{z}), \forall \mathbf{y}' \in F(\mathbf{z}'), \\ &\quad y^i - y'^i \leq \overline{\mathbf{a}}_i \cdot [\mathbf{z} - \mathbf{z}']^+ + \underline{\mathbf{a}}_i \cdot [\mathbf{z} - \mathbf{z}']^- + 2\overline{d}^i \end{aligned} \tag{3.30}$$

where $[\mathbf{z}]^+ = \max(\mathbf{z}, 0)$ and $[\mathbf{z}]^- = \min(\mathbf{z}, 0)$.

Let us assume that we are given a set of random data generated by the map F:

$$\tilde{\mathcal{D}} = \{(\mathbf{z}_k, \mathbf{y}_k, \mathbf{z}'_k, \mathbf{y}'_k) \mid \mathbf{y}_k \in F(\mathbf{z}_k), \mathbf{y}'_k \in F(\mathbf{z}'_k), k \in \tilde{\mathbb{K}}\}$$

where $\tilde{\mathbb{K}}$ is a finite set of indices. We assume that the samples in $\tilde{\mathcal{D}}$ are independent and identically distributed. We aim at computing bounds such that (3.30) holds with probabilistic guarantees. This can be done using the scenario approach [18], which essentially consists of computing bounds such that the inequality in (3.30) holds at all points in $\tilde{\mathcal{D}}$. However, to obtain high confidence bounds, one needs to consider large data sets, resulting in large linear programs that can be complicated to solve in practice. For that reason, we present, in the following, a two-step approach that allows us to deal with very large data sets in practice. Let us partition the set of $\tilde{\mathcal{D}}$ in two subsets $\tilde{\mathcal{D}}'$ and $\tilde{\mathcal{D}}''$ indexed by $\tilde{\mathbb{K}}'$ and $\tilde{\mathbb{K}}''$ such that $\tilde{\mathbb{K}} = \tilde{\mathbb{K}}' \cup \tilde{\mathbb{K}}''$. In the first step, $\tilde{\mathcal{D}}'$ will be used to estimate the bounds $\underline{a}_{ij}, \overline{a}_{ij}$. In the second step, $\tilde{\mathcal{D}}''$ will be used to estimate the bound $\overline{d}^i$. Typically, the number of samples in $\tilde{\mathcal{D}}''$ will be much larger than that in $\tilde{\mathcal{D}}'$.

We first consider the following linear program:

$$\begin{cases} \min\limits_{\underline{\mathbf{a}}_i, \overline{\mathbf{a}}_i, \overline{d}^i} & \overline{\mathbf{a}}_i \cdot \sum_{k \in \tilde{\mathbb{K}}'} [\mathbf{z}_k - \mathbf{z}'_k]^+ + \underline{\mathbf{a}}_i \cdot \sum_{k \in \tilde{\mathbb{K}}'} [\mathbf{z}_k - \mathbf{z}'_k]^- + 2|\tilde{\mathbb{K}}'|\overline{d}^i \\ \text{s.t.} & y^i_k - y'^i_k \leq \overline{\mathbf{a}}_i \cdot [\mathbf{z}_k - \mathbf{z}'_k]^+ \\ & \qquad + \underline{\mathbf{a}}_i \cdot [\mathbf{z}_k - \mathbf{z}'_k]^- + 2\overline{d}^i,\ \forall k \in \tilde{\mathbb{K}}' \\ & \underline{\mathbf{a}}_i \preceq \overline{\mathbf{a}}_i. \end{cases} \tag{3.31}$$

The constraints of (3.31) are chosen such that (3.30) holds at all points in $\tilde{\mathcal{D}}'$, while the cost is chosen so as to minimize the average value of the right-hand side of (3.30). Let $\underline{\mathbf{a}}^\star_i, \overline{\mathbf{a}}^\star_i$ denote the optimal values of $\underline{\mathbf{a}}_i, \overline{\mathbf{a}}_i$ in (3.31). The optimal value of $\overline{d}^i$ is disregarded and estimated again in the second step.

Indeed, let us define

$$\overline{d}^{\star i} = \frac{1}{2} \max \left\{ \begin{array}{r} y^i_k - y'^i_k - \overline{\mathbf{a}}^\star_i \cdot [\mathbf{z}_k - \mathbf{z}'_k]^+ \\ -\underline{\mathbf{a}}^\star_i \cdot [\mathbf{z}_k - \mathbf{z}'_k]^- \end{array} \;\middle|\; k \in \tilde{\mathbb{K}}'' \right\}. \tag{3.32}$$

Let us remark that while (3.31) can be solved using a moderately large data set $\tilde{\mathcal{D}}'$, (3.32) can easily be estimated on very large data sets. Moreover, we can provide probabilistic guarantees on the estimated bounds:

Proposition 14. *Let $\beta \in (0,1)$ be a confidence parameter and let $i \in \{1,\dots,m\}$, let $\underline{\mathbf{a}}_i^\star, \overline{\mathbf{a}}_i^\star$ and $\overline{d}^{\star i}$ be estimated bounds using* (3.31) *and* (3.32)*, respectively. Then, with probability at least $1-\beta$, it holds that*

$$\mathbb{P}(y^i - y'^i \leq \overline{\mathbf{a}}_i^\star \cdot [\mathbf{z}-\mathbf{z}']^+ + \underline{\mathbf{a}}_i^\star \cdot [\mathbf{z}-\mathbf{z}']^- + 2\overline{d}^{\star i}) \geq \beta^{\frac{1}{|\tilde{\mathbb{K}}''|}}. \tag{3.33}$$

Proof. Consider the following robust linear program

$$\begin{cases} \min\limits_{\overline{d}^i} & \overline{d}^i \\ \text{s.t.} & y^i - y'^i \leq \overline{\mathbf{a}}_i^\star \cdot [\mathbf{z}-\mathbf{z}']^+ + \underline{\mathbf{a}}_i^\star \cdot [\mathbf{z}-\mathbf{z}']^- + 2\overline{d}^i, \\ & \qquad\qquad \forall \mathbf{z},\mathbf{z}' \in Z,\ \forall \mathbf{y} \in F(\mathbf{z}), \forall \mathbf{y}' \in F(\mathbf{z}'). \end{cases} \tag{3.34}$$

Following the scenario approach, we consider the following associated scenario linear program:

$$\begin{cases} \min\limits_{\overline{d}^i} & \overline{d}^i \\ \text{s.t.} & y_k^i - y_k'^i \leq \overline{\mathbf{a}}_i^\star \cdot [\mathbf{z}_k - \mathbf{z}_k']^+ \\ & \qquad + \underline{\mathbf{a}}_i^\star \cdot [\mathbf{z}_k - \mathbf{z}_k']^- + 2\overline{d}^i,\ \forall k \in \tilde{\mathbb{K}}''. \end{cases} \tag{3.35}$$

It is easy to see that (3.32) provides the unique solution to (3.35). Moreover, from the results in [18], we get that (3.33) holds for the solution of (3.35). □

Hence, we have provided a method to estimate bounds on the disturbances and bounds on the partial derivatives. By using a two-step approach, our method can be used with very large data sets and can thus estimate the bounds with very high confidence. Let us remark that the method can be used to check if the system is monotone by checking if $\underline{\mathbf{a}}_i^\star \succeq 0$, for all $i \in \{1,\dots,m\}$.

3.4 . Numerical examples

Similarly to the previous chapters, we will test the performance of the proposed approaches. We will partition the input space similarly to the previous chapters's example defined in (2.20). To test the performance of the proposed approaches, we will consider the CPU time of executing the algorithms and the tightness of the over-approximation measured using the performance criterion defined in (2.21). In this example, we will consider the following set-valued map $F : [-\pi,\pi]\times[-\pi,\pi] \rightrightarrows \overline{\mathbb{R}}$ defined as follows:

$$F(\mathbf{z}) = \{\cos(z^1) + 0.5 \cdot \cos(z^2) + d \mid d \in [-0.1, 0.1]\} \tag{3.36}$$

We use F to generate the sets of data $\mathcal{D}$ used in the subsequent experiments. We will compare the performance of the two approaches introduced in Section 3.1.1 and Section 3.1.2.

We implemented the following approaches:

1. The approach of transforming the data set to the monotone case using only the matrix A^+ and A^-, and then using Equations 3.8 and 3.9 for the final step.

2. The approach of transforming the data set to the monotone case using all the possible matrices, and then using Equations 3.8 and 3.9 for the final step.

3. The approach of transforming the data set to the monotone case using all the possible matrices, and then using Equations 3.6 and 3.7 for the final step (The linear programming step.)

4. The growth cones approach with 3.27 and 3.26 as final step.

5. The growth cones approach with 3.25 and 3.24 as final step (The linear programming step.)

Approaches 1, 2, and 4 are carried out using compiled C++ code, while approaches 3 and 5 are carried out using the script language Matlab. Thus, for execution time we compare them in two groups. The first group contains approaches 1, 2, and 4. The second group contains approaches 3 and 5.

First, we visualized the over-approximation. We sampled $|\mathbb{K}| = 10^7$ data points. The parameters of the partition are chosen as follows $K^1 = 50$, $K^2 = 50$, $c = 0.01$. Figure 3.5 shows the undisturbed function in solid and the over-approximation calculated from data. We see how the undisturbed function is included in the over-approximation.

3.4.1 . The effect of changing the number of data points

To study the effect of changing the number of data points, we chose and fixed a partition, $K^1 = 50$, $K^2 = 50$. Then, for an increasing number of data points, we calculated the over-approximation of the map F and measured the execution time and the performance criterion. For each number of data points, we redo the experiment five times using different randomly generated data sets. The results of this statistical study of changing the number of data points to the conservatism measure μ are shown in Figure 3.6. We can see from the figure that the performance criterion decreases with the increase in the number of data points for all the approaches. We observe that approaches 1, 2, and 4 converge to a value of μ higher than approaches 3 and 5. This is due to the fact that approaches 3 and 5 use a linear programming step to tighten the over-approximation. Approach 5 is the best in terms of the performance criterion. The execution time of the

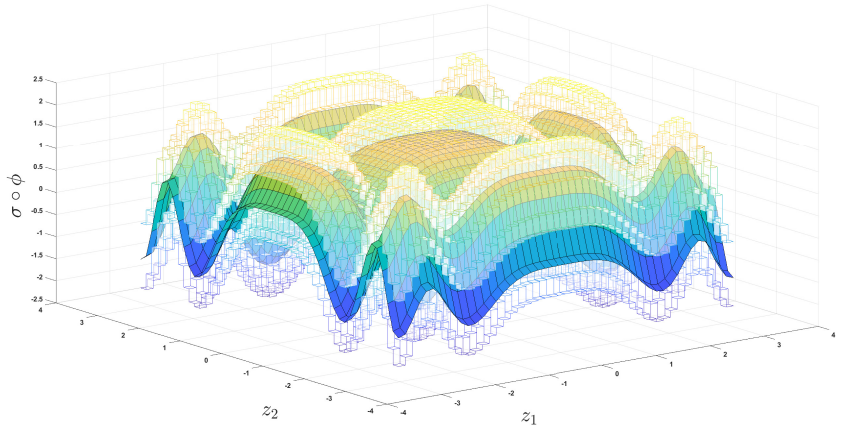


Figure 3.5: Map $F(\mathbf{z})$ with $d = 0$ everywhere is represented in solid. The upper and lower bounds of the over-approximation are represented in transparency

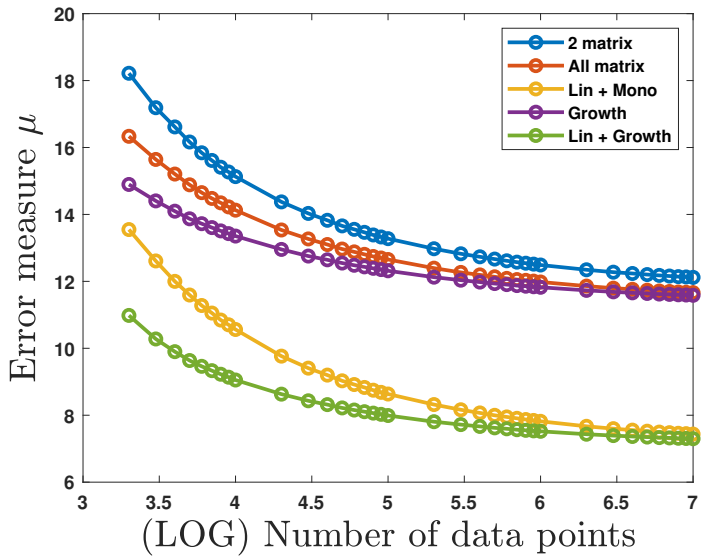


Figure 3.6: The relation between the (log) number of data points and the conservatism measure μ for different methods of over-approximation. The line in blue represents approach 1. The line in red represents approach 2. The line in yellow represents approach 3. The line in purple represents approach 4. The line in green represents approach 5.

approaches is shown in Figure 3.7 and Figure 3.8. We see that approaches 1, 2, and 4 scale linearly with the number of data points, which is expected and this is

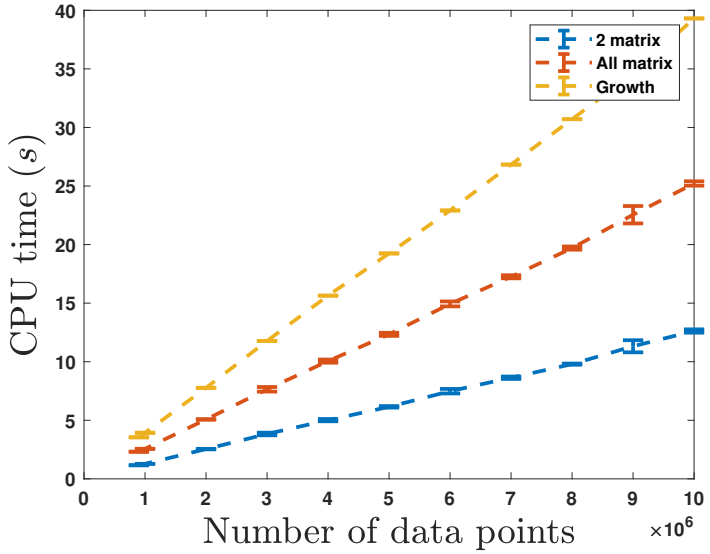


Figure 3.7: The figure shows the average time of execution with respect to the number of data points, with bars representing the standard deviation. The line in yellow represents approach 1. The line in red represents approach 2. The line in bleu represents approach 4.

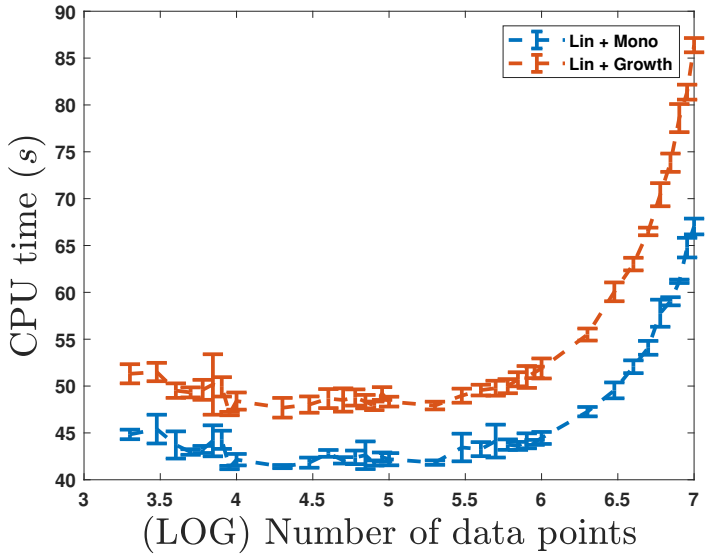


Figure 3.8: The figure shows the average time of execution with respect to the number of data points, with bars representing the standard deviation. The line in red represents approach 3. The line in bleu represents approach 5.

why we only show the relation using only the last ten possible values of the number of data points. For approaches 3 and 5, we observe that the execution time when the number of data points is smaller than 10^6 is predominantly due to the linear programming step. Thus, it is almost constant. Then, the execution time starts to increase when the number of data points increases more than 10^6.

3.4.2 . The effect of changing the size of the partition

In the second experiment, we fixed the number of data points $|\mathbb{K}| = 10^4$, and changed the size of the partition. For each size considered, the partition is chosen such that $K^1 = K^2$,. We also redo the calculation for each size five times using different randomly sampled data sets. Although we are using a uniform partition, we do not make use of this fact in the calculation of the over-approximation, and we still use binary search to find the partition element that contains the data point. Figure 3.9 shows the results of the experiment. All the approaches show the same behavior. The conservatism measure μ decreases with the increase in the number of partition elements. We observe that the conservatism measure μ of the approaches converges to the same value.

The time of execution for approaches 1, 2, and 4 was less than 70 milliseconds for all the sizes of the partition. For approaches 3 and 5, the execution time was less than 30 seconds for all the sizes of the partition. There were big fluctuations in the execution time of all the approaches to make the experiment meaningful.

3.4.3 . Comparison with the state-of-the-art robust models

We have already shown both theoretically in Proposition 6 and experimentally, the computational complexity of calculating the introduced data-driven model. Let us compare with the set membership approaches in [60] and [19] where an optimization problem is used to find the model. The complexity of this approach grows polynomically with the number of data points. In contrast, our approach scales linearly. The maximum reported number of data points used in [60] is in the tens of thousands. The number of points used is less in [19]. As can be seen, the introduced approach in this paper can handle orders of magnitude more data points. The same can be said about [66], where a Mixed Integer Linear Programming (MILP) problem is used to find the model, and the reported number of data points used in the case study is 400.

3.5 . Conclusion

In this chapter, we studied the over-approximation of maps defined by functions with bounded derivatives and added bounded disturbances. We proposed several approaches to compute such over-approximations. Although the calculated over-approximations are generally not as tight as the ones computed in previous chapters for monotone maps, they can be computed efficiently. We make the trade-off between the tightness of the over-approximation and the computational

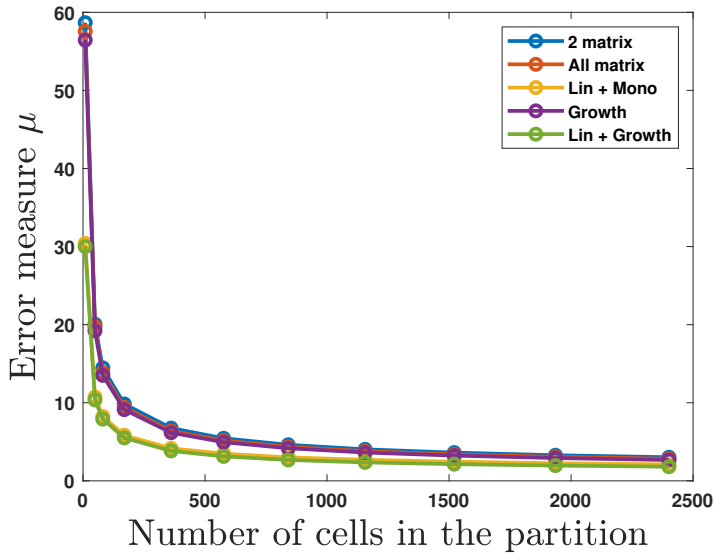


Figure 3.9: The relation between the number of cells in the partition and the conservatism measure μ for different methods of over-approximation. The line in blue represents approach 1. The line in red represents approach 2. The line in yellow represents approach 3. The line in purple represents approach 4. The line in green represents approach 5.

complexity. The introduced approaches rely on one of two ideas: The first one is to transfer the problem to the case we studied in the previous chapters by performing a transformation on the data. The second one depends on computing the value of growth cones on the vertices of the cells of the partition. Those growth cones, with apexes at the data points, are defined using bounds on the partial derivatives of the function. Finally, we presented an approach to compute bounds on the partial derivatives of the function and on the disturbances directly from data. The proposed approach is based on the scenario approach and allows us to deal with very large data sets in practice.

4 - Data-Driven Abstraction

In previous chapters, we studied the problem of over-approximating unknown dynamical systems using data generated from the system. In particular, we consider the case of unknown but monotone systems and the case of unknown but with bounded derivatives systems. In this chapter, we will use the results of the previous chapters to find finite state representations of the unknown systems. The concept of finding a finite state representation of a given system is called *abstraction*. In general, the idea behind abstraction is to find a simpler model of the system that can be used to reason about the original system. The simpler model can be used, for instance, to find a controller for the original system. For this scheme to work, we need to ensure that the simpler model is a good representation of the original system. In other words, we need to ensure that the behavior of the original system can be reproduced by the simpler model. This is done by finding a relation between the states of the original system and the states of the simpler model. [71] In a special case, studied in symbolic control, the simpler model is a finite state model, meaning that the infinite number of states of the original system are mapped to a finite set of states. Being able to work with finite state representations opens the door to the possibility of using formal verification and discrete controller synthesis techniques, implementing automated procedures to find correct-by-design controllers satisfying a variety of complex specifications [9]. Those controllers are capable of ensuring complex specifications (safety and reachability [29], behaviors described by automata [62], or temporal logic formulas [9]). Previously, we studied interval-valued over-approximation from data on interval partitions. In this chapter, we will consider the problem of computing abstractions for those learned interval-valued models. In the case of a purely unknown system, we show that working with the finite-state symbolic models does not add any conservatism compared to working with the data-driven over-approximation models.

Although symbolic control is usually presented as a model-driven technique, some data-driven approaches have emerged recently. For instance, approaches that only require the ability to sample the system dynamics on a given grid of states and inputs are introduced in [56, 32], or [77]. In [25], data-driven abstractions are found within the PAC (probably approximately correct) statistical framework. PAC guarantees are also used in [22] to build data-driven abstraction through the use of the scenario approach. The scenario approach is also used in [46] and [41] to offer probabilistic guarantees on the data-driven built abstractions. The main difference between all the previously mentioned work and the approach introduced here is that, in contrast to our approach, they all give probabilistic guarantees. Data-driven control design with regular language specifications is studied in [63] for plants described as abstract systems. In comparison to [63], our approach can handle bounded disturbances and is able to satisfy the specifications robustly

not to a predefined desired accuracy. Unknown parts of nonlinear systems are modeled using Gaussian processes; then, those models are used in building symbolic abstractions in [33].

This chapter is organized as follows. In Section 4.1, we introduce the transition system representation that will be used throughout the chapter. In Section 4.2, we introduce the notion of systems relations and abstractions. In Section 4.4, we introduce the notion of data-driven abstraction and show how to build it using the results of the previous chapters. Finally, we illustrate the results of this chapter on numerical examples in Section 4.5.

4.1 . System definition

In this section, we introduce the transition system representation that will be used throughout the chapter. transition systems representation is a powerful tool to represent dynamical systems. It is capable of representing both continuous and discrete state systems. It is also capable of representing systems combining both continuous and discrete state systems, like hybrid systems.

Definition 9. *A transition system T is a tuple $T = (X, U, \Delta, Y, H)$, where:*

- *X is a set of states,*
- *U is a set of inputs,*
- *$\Delta : X \times U \rightrightarrows X$ is a transition relation,*
- *Y is a set of outputs, and*
- *$H : X \to Y$ is an output map.*

An input $\mathbf{u} \in U$ is called *enabled* at $\mathbf{x} \in X$ if $\Delta(\mathbf{x}, \mathbf{u}) \neq \emptyset$. The set of all inputs enabled at $\mathbf{x}$ is denoted $\text{enab}_\Delta(\mathbf{x})$. A transition system T is said to be *deterministic* if for all $\mathbf{x} \in X$, for all $\mathbf{u} \in U$ $\Delta(\mathbf{x}, \mathbf{u})$ contains at most one element. A system is said to be *non-blocking* if for all $\mathbf{x} \in X$, $\text{enab}_\Delta(\mathbf{x}) \neq \emptyset$. If X and U are finite sets, then T is called a *finite transition system*. Otherwise, T is called an *infinite transition system*. Given that $\mathbf{x}' \in \Delta(\mathbf{x}, \mathbf{u})$, we say that $\mathbf{x}'$ is a *successor* of $\mathbf{x}$ under input $\mathbf{u}$, or a $\mathbf{u}$-successor. The next example illustrates the notion of transition Systems and shows how it can be used to represent continuous and discrete state systems.

Example 5

Consider the following dynamical system:

$$x(\tau + 1) = x(\tau) + u(\tau). \tag{4.1}$$

where $x(\tau) \in [\underline{x}, \overline{x}] \subseteq \mathbb{R}$ and $u(\tau) \in [\underline{u}, \overline{u}] \subseteq \mathbb{R}$ denote the state and the control

input, respectively. Then the transition system $T_1 = (X, U, \Delta, Y, H)$ associated to (4.1) is defined as follows:

- $X = [\underline{x}, \overline{x}]$,
- $U = [\underline{u}, \overline{u}]$,
- $\Delta(x, u) = \{x + u\}$,
- $Y = [\underline{x}, \overline{x}]$,
- $H(x) = x$.

Example 6

Let us consider the discrete system represented in Figure 4.1. The associated transition system $T_2 = (X, U, \Delta, Y, H)$ is defined as follows:

- $X = \{q_1, q_2, q_3, q_4\}$,
- $U = \{p_1, p_2\}$,
- $\Delta(q_1, p_1) = \{q_3\}$, $\Delta(q_1, p_2) = \{q_3, q_4\}$, $\Delta(q_2, p_1) = \{q_2\}$, $\Delta(q_3, p_2) = \{q_2\}$,
- $Y = \{q_1, q_2, q_3, q_4\}$,
- $H(q_1) = q_1$, $H(q_2) = q_2$, $H(q_3) = q_3$, $H(q_4) = q_4$.

System T_2 is nondeterministic because $\Delta(q_1, p_2) = \{q_3, q_4\}$ contains more than one element. The system is blocking because $\text{enab}_\Delta(q_4) = \emptyset$.

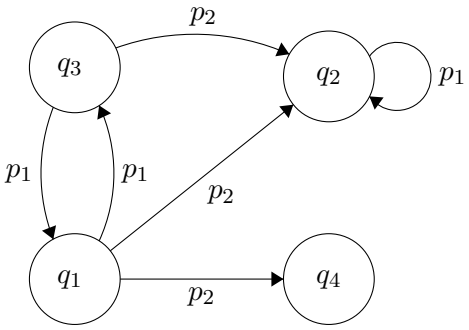


Figure 4.1: Example of a transition system.

4.2 . Systems relations and Abstractions

When modeling any system in nature, there is always the assumption or at least the hope that the output of the model will be similar to the output of the

real system. How to define this similarity is the question that we will try to answer in this section. Moreover, in the case of feedback control, if I find a controller that works for the model, will it work for the real system?

The first tool that we will use to define this similarity is *simulation*.

Definition 10. *Consider two transition systems $T_1 = (X_1, U_1, \Delta_1, Y_1, H_1)$ and $T_2 = (X_2, U_2, \Delta_2, Y_2, H_2)$. A relation $R \subseteq X_1 \times X_2$ is a simulation relation from T_1 to T_2 if the following conditions are satisfied:*

1. *for all $\mathbf{x}_1 \in X_1$, there exists $\mathbf{x}_2 \in X_2$ such that $(\mathbf{x}_1, \mathbf{x}_2) \in R$,*
2. *for all $(\mathbf{x}_1, \mathbf{x}_2) \in R$, $H_1(\mathbf{x}_1) = H_2(\mathbf{x}_2)$,*
3. *for all $(\mathbf{x}_1, \mathbf{x}_2) \in R$, for all $\mathbf{u}_1 \in enab_{\Delta_1}(\mathbf{x}_1)$, for all $\mathbf{x}_1' \in \Delta_1(\mathbf{x}_1, \mathbf{u}_1)$, there exists $\mathbf{u}_2 \in enab_{\Delta_2}(\mathbf{x}_2)$ such that there exists $\mathbf{x}_2' \in \Delta_2(\mathbf{x}_2, \mathbf{u}_2)$ satisfying $(\mathbf{x}_1', \mathbf{x}_2') \in R$.*

Then, we say that:

- *T_1 is simulated by T_2, denoted $T_1 \preceq_{\mathcal{S}} T_2$, if there exists a simulation relation R from T_1 to T_2,*
- *T_1 is bisimilar to T_2, denoted $T_1 \cong_{\mathcal{S}} T_2$, if there exists a relation R such that R is a simulation relation from T_1 to T_2 and R^{-1} is a simulation relation from T_2 to T_1.*

We only mention simulation here to describe its limitations and motivate the need for alternating simulation. Intuitively, a simulation relation R from T_1 to T_2 means that for every state $\mathbf{x}_1$ in T_1, there exists a state $\mathbf{x}_2$ in T_2 such that $\mathbf{x}_1$ and $\mathbf{x}_2$ are related by R. Moreover, for every input $\mathbf{u}_1$ enabled at $\mathbf{x}_1$, there exists an input $\mathbf{u}_2$ enabled at $\mathbf{x}_2$ such that for every successor $\mathbf{x}_1'$ of $\mathbf{x}_1$ under $\mathbf{u}_1$, there exists a successor $\mathbf{x}_2'$ of $\mathbf{x}_2$ under $\mathbf{u}_2$ such that $\mathbf{x}_1'$ and $\mathbf{x}_2'$ are related by R. In other words, T_1 is simulated by T_2 if the behavior of T_1 can be reproduced by T_2.

This relation allows us to model a system with a simpler one, and then use the simpler model to reason about the original system. In control theory, we aim to design a controller for a simpler model and then use it to control the original system. But the simulating model can be non-deterministic, and allowed to have reacher behavior than the real system; in some cases, we can synthesize a feedback controller for the simulation model and choose an input for the simulation model that has no corresponding input in the real system. The following definition is a solution to this problem.

Definition 11. *Let us consider two transition systems $T_i = (X_i, U_i, \Delta_i, Y_i, H_i)$ $i = 1, 2$, sharing the same sets of outputs ($Y_1 = Y_2 = Y$). A relation $R \subseteq X_1 \times X_2$ is an alternating simulation relation from T_1 to T_2 if the following conditions are satisfied:*

1. *for all* $\mathbf{x}_1 \in X_1$*, there exists* $\mathbf{x}_2 \in X_2$ *such that* $(\mathbf{x}_1, \mathbf{x}_2) \in R$*;*

2. *for all* $(\mathbf{x}_1, \mathbf{x}_2) \in R$*,* $H_1(\mathbf{x}_1) = H_2(\mathbf{x}_2)$*;*

3. *for all* $(\mathbf{x}_1, \mathbf{x}_2) \in R$*, for all* $\mathbf{u}_1 \in enab_{\Delta_1}(\mathbf{x}_1)$*, there exists* $\mathbf{u}_2 \in enab_{\Delta_2}(\mathbf{x}_2)$ *such that for all* $\mathbf{x}_2' \in \Delta_2(\mathbf{x}_2, \mathbf{u}_2)$*, there exists* $\mathbf{x}_1' \in \Delta_1(\mathbf{x}_1, \mathbf{u}_1)$ *satisfying* $(\mathbf{x}_1', \mathbf{x}_2') \in R$*.*

Then, we say that:

- T_1 *is* alternatingly simulated *by* T_2*, denoted* $T_1 \preceq_{\mathcal{AS}} T_2$*, if there exists an alternating simulation relation* R *from* T_1 *to* T_2*;*

- T_1 *is* alternatingly bisimilar *to* T_2*, denoted* $T_1 \cong_{\mathcal{AS}} T_2$*, if there exists a relation* R *such that* R *is an alternating simulation relation from* T_1 *to* T_2 *and* R^{-1} *is an alternating simulation relation from* T_2 *to* T_1*.*

An alternating simulation relation R from T_1 to T_2 means that for every state $\mathbf{x}_1$ in T_1, there exists a state $\mathbf{x}_2$ in T_2 such that $\mathbf{x}_1$ and $\mathbf{x}_2$ are related by R. Moreover, if I choose an input $\mathbf{u}_1$ enabled at $\mathbf{x}_1$ (the model), there exists an input $\mathbf{u}_2$ enabled at $\mathbf{x}_2$ (the original system) such that for every successor $\mathbf{x}_2'$ of $\mathbf{x}_2$ under $\mathbf{u}_2$ there exists a successor $\mathbf{x}_1'$ of $\mathbf{x}_1$ under $\mathbf{u}_1$ such that $\mathbf{x}_1'$ and $\mathbf{x}_2'$ are related by R. Meaning that the behavior of the original system can be reproduced by the model. There will be no input in the model that has no corresponding input in the original system.

In the case of bisimulation, working with the original system or the model is equivalent. There is no added conservatism in the model. This will imply that if I cannot find a controller for the model, then I cannot find a controller for the original system.

4.2.1 . Abstraction

We saw previously that we can use (alternating)simulation to model a system with a simpler one, and then use the simpler model to reason about the original system. If the model alternatingly simulates the original system, then we can build a feedback controller for the model and use it to control the original system. We call this process *abstraction-based control*, and the simpler model is called an *abstraction* of the original system.

In symbolic control, we use finite state abstractions, which are finite transition systems that simulate the original system. To find a finite state abstraction, we can discretize the state space of the original system. Each state of the abstraction represents (aggregate) a set of states of the original system. Similarly, we can discretize the input space of the original system. Then, we can use the computed sets of successors of each state under each input to define a finite transition system that simulates the original system. This is called a *discrete abstraction* of the original system. The successor of each state under each input is the set of states

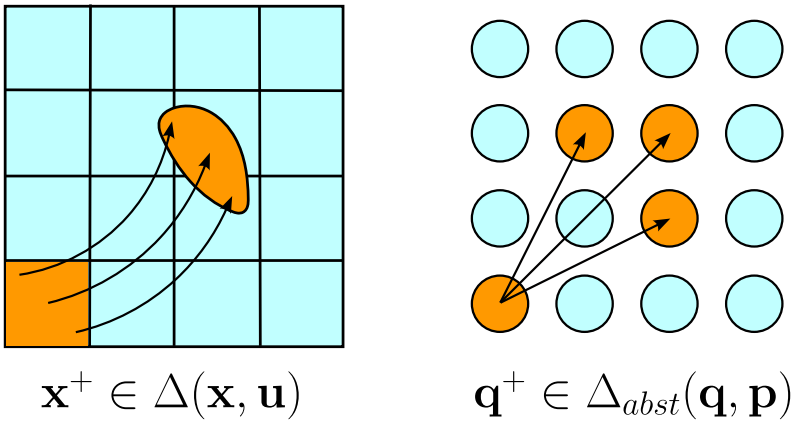


Figure 4.2: A representation of the abstraction process.

that can be reached from this state under this input. Calculating the successors, depicted in Figure 4.2, requires knowledge of the dynamics of the original system to find the one-step reachable set (the over-approximation of the dynamics) of each state under each input. To solidify the idea of abstraction, we will use the following example.

Example 7

Consider the system described in Example 4.1. Let us assume that we want to find a finite state abstraction of this system $T_{abst} = (X_{abst}, U_{abst}, \Delta_{abst}, Y_{abst}, H_{abst})$. First, we discretize the state space $[\underline{x}, \overline{x}]$ into n states $Q = \{q_1, \ldots, q_n\}$, where for all $i \in \{1, \ldots, n\}$, $q_i = [\underline{x} + \frac{i-1}{n}(\overline{x} - \underline{x}), \underline{x} + \frac{i}{n}(\overline{x} - \underline{x})] = [\underline{q}_i, \overline{q}_i]$. We also discretize the input space $[\underline{u}, \overline{u}]$ into m inputs $P = \{p_1, \ldots, p_m\}$. Similarly, for all $i \in \{1, \ldots, m\}$, $p_i = [\underline{u} + \frac{i-1}{m}(\overline{u} - \underline{u}), \underline{u} + \frac{i}{m}(\overline{u} - \underline{u})] = [\underline{p}_i, \overline{p}_i]$. The set of states of the abstraction is $X_{abst} = Q$ and the set of inputs is $U_{abst} = P$. Then, we can compute the set of successors of each state under each input. For example, for all $i \in \{1, \ldots, n\}$, for all $j \in \{1, \ldots, m\}$,

$$\Delta_{abst}(q_i, p_j) = \{q_i^\star \mid q_i^\star \in [\underline{q}_i + \underline{p}_j, \overline{q}_i + \overline{p}_j]\}.$$

If we chose the same set of outputs for the original system and the abstract one $Y = Y_{abst} = X_{abst}$ and the same output map $H_{abst}(q) = q$, $H(x) = q$ such that $x \in q$, then the abstraction T_{abst} is a discrete abstraction of the original system and it is straightforward to check that $T_{abst} \preceq_{\mathcal{AS}} T$.

In previous chapters, we saw that we can use data to learn this one-step reachable set, which opens the door to the possibility of finding a finite state abstraction of a system without knowing its dynamics. This is the backbone of the introduced data-driven control scheme.

The final piece of the puzzle is to choose the control action to be applied to

the original system based on the control action applied to the abstraction, which is called *refinement*. Usually, many control actions in the original system can be mapped to the same control action in the abstraction. This is done using an *interface* $\mathcal{I} : X_1 \times X_2 \times U_1 \to U_2$ that maps the control actions of the abstraction to the control actions of the original system. An example of such an interface for the system in Example 4.2.1 is given by

$$\mathcal{I}(q_i, x, p_j) = \frac{\underline{p}_j + \overline{p}_j}{2}$$

for all $i \in \{1, \ldots, n\}$, for all $j \in \{1, \ldots, m\}$, and for all $x \in q_i$. This chosen interface does not depend on the state of the original system or the state of the abstraction.

4.3 . Discrete controller synthesis

We talked previously about discrete abstraction and how we can use it to find a finite state representation of continuous (infinite) state systems. Dealing with finite state systems allows us to use the tools of formal verification to reason about the system, and to synthesize controllers that satisfy certain specifications. In this section, we describe some of the famous specifications used in formal verification and control synthesis, and how to synthesize controllers that satisfy these specifications.

4.3.1 . Safety

Safety is one of the most common specifications used in formal verification and control synthesis. It is used to ensure that the system will never enter an unsafe state.

Definition 12. *Consider a transition system $T = (X, U, \Delta, Y, H)$. A safety controller C_{safe} for the safe set X_s and is a set-valued map $C_{safe} : X \rightrightarrows U$ satisfying*

- $dom(C_{safe}) \subseteq X_s$,
- $\forall \mathbf{x} \in dom(C_{safe}), \forall \mathbf{u} \in C_{safe}(\mathbf{x}), \Delta(\mathbf{x}, \mathbf{u}) \subseteq dom(C_{safe})$,

where $dom(C_{safe}) = \{\mathbf{x} \in X \mid C_{safe}(\mathbf{x}) \neq \emptyset\}$ is the domain of C_F.

In other words, a safety controller is a controller that ensures that the system will never enter an unsafe state. In the case where the system T is finite-state, to find the set that contains the maximum number of safe states what we call the maximum control invariant set X_{inv}, we can use the fixed point algorithm presented in Algorithm 2.

Finding a controller for the safety game can be done by removing, from the set of enabled actions, all the actions that lead to a state outside of the (maximum) control invariant set.

Algorithm 2 Fixed point algorithm for safety

Input The unsafe states $X_{\text{unsafe}} \subseteq X$
Output The maximum control invariant set $X_{\text{inv}} \subseteq X$

1: $X_{\text{safe}} \leftarrow X \setminus X_{\text{unsafe}}$
2: **while** X_{safe} changes **do**
3: $\quad X_{\text{safe}} \leftarrow X_{\text{safe}} \setminus \{\mathbf{x} \in X_{\text{safe}} \mid \forall\, \mathbf{u} \in \text{enab}_{\Delta}(\mathbf{x}), \Delta(\mathbf{x}, \mathbf{u}) \cap X_{\text{unsafe}} \neq \emptyset\}$
4: $\quad X_{\text{unsafe}} \leftarrow X \setminus X_{\text{safe}}$
5: **end while**
6: **return** $X_{\text{inv}} = X_{\text{safe}}$

Example 8

Consider the transition system T_1 in Example 4.1. Let $X_{\text{unsafe}} = \{q_4\}$. For the system to be safe, we need to remove the actions that lead to q_4 from the set of enabled actions at q_1. Namely, we need to remove p_2 from $\text{enab}_{\Delta}(q_1)$. The resulting transition system is safe.

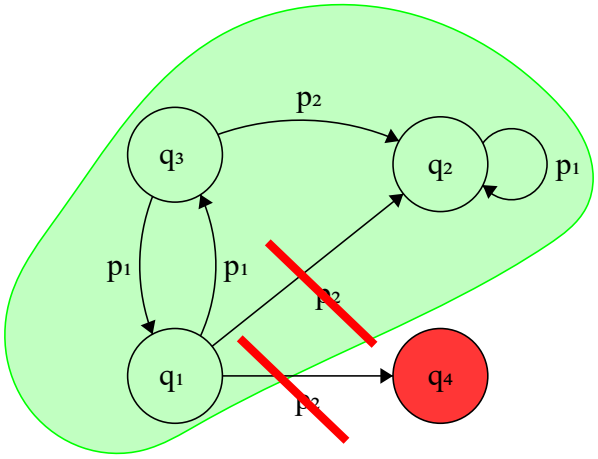


Figure 4.3: Unsafe state in T_2.

4.3.2 . Reachability

Reachability is another common specification used in formal verification and control synthesis. It is used to ensure that the system will eventually reach a set of desirable states.

Definition 13. *Consider a transition system $T = (X, U, \Delta, Y, H)$ and a set of desirable states $X_{desirable} \subseteq X$. Then, the controller $C_{reach} : X \to U$ is a reachability controller if all the trajectories of the closed-loop system $T_{C_{reach}}$ reach a state in $X_{desirable}$.*

In other words, a reachability controller is a controller that ensures that the system will eventually reach a state in $X_{\text{desirable}}$.

In the case where the system T is finite-state, to find the set that contains the maximum number of states from which the desirable states are reachable, we can use the fixed point algorithm presented in Algorithm 3. Let us define

Algorithm 3 Fixed point algorithm for reachability

Input The desirable states $X_{\text{desirable}} \subseteq X$

Output The sets of states from which the desirable states are reachable $X_{\text{reach}}(i)$ in at most i steps for all $i \in \{0, \dots, k\}$.

1: $X_{\text{reach}}(0) \leftarrow X_{\text{desirable}}$
2: **while** X_{reach} changes **do**
3: $\quad X_{\text{reach}}(k+1) \leftarrow X_{\text{reach}}(k) \cup \{\mathbf{x} \in X \mid \exists\, \mathbf{u} \in \text{enab}_\Delta(\mathbf{x}), \Delta(\mathbf{x}, \mathbf{u}) \subseteq X_{\text{reach}}\}$
4: $\quad k \leftarrow k+1$
5: **end while**
6: **return** $X_{\text{reach}}(0), \dots, X_{\text{reach}}(k)$

the controller $C_{\text{reach}} : X \rightrightarrows U$ as follows. For all $i \in \{0, \dots, k\}$, for all $\mathbf{x} \in X_{\text{reach}}(i) \setminus X_{\text{reach}}(i-1)$

$$C_{\text{reach}}(\mathbf{x}) = \{\mathbf{u} \mid \mathbf{u} \in \text{enab}_\Delta(\mathbf{x}), \Delta(\mathbf{x}, \mathbf{u}) \subseteq X_{\text{reach}}(i-1)\}. \tag{4.2}$$

where $X_{\text{reach}}(i)$ is the i-th output of Algorithm 3.

Example 9

Consider the transition system T_2 in Example 4.1. Let $X_{\text{desirable}} = \{q_4\}$. The desirable states are reachable from q_1 and q_3. The reachability controller is defined as follows:

$$C_{\text{reach}}(q_1) = \{p_1\}, \quad C_{\text{reach}}(q_2) = \emptyset, \quad C_{\text{reach}}(q_3) = \{p_2\}, \quad C_{\text{reach}}(q_4) = \emptyset.$$

4.4 . Data-Driven Abstraction

In this section, we show that our approach presented can be used for data-driven modeling of discrete-time dynamical systems. We show that simulating maps computed from data can be used to define sound finite state abstractions for unknown dynamical systems. The unknown dynamics can be either monotone, making them suitable for the approach presented in Section 2, or with bounded derivatives, making them suitable for the approach presented in Section 3.

We consider a discrete-time dynamical system:

$$\mathbf{x}(\tau+1) \in f(\mathbf{x}(\tau), \mathbf{u}(\tau)) + g(\mathbf{x}(\tau), \mathbf{u}(\tau)) + D \tag{4.3}$$

where $\mathbf{x}(\tau) \in X \subseteq \overline{\mathbb{R}}^{n_x}$ and $\mathbf{u}(\tau) \in U \subseteq \overline{\mathbb{R}}^{n_u}$ denote the state and the control input, $g : X \times U \to \overline{\mathbb{R}}^{n_x}$ is a known function whereas $f : X \times U \to \overline{\mathbb{R}}^{n_x}$ is an unknown function which is either monotone or has bounded derivatives. $D = [\underline{\mathbf{d}}, \overline{\mathbf{d}}] \subseteq \mathbb{R}^{n_x}$ is a bounded interval of disturbances with known bounds.

Let be given a set of data $\mathcal{D} \subseteq X \times U \times \overline{\mathbb{R}}^{n_x}$ consisting of transitions sampled from the system (4.3):

$$\mathcal{D} = \left\{ (\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}'_k) \;\middle|\; \begin{array}{r} \mathbf{x}'_k \in f(\mathbf{x}_k, \mathbf{u}_k) + g(\mathbf{x}_k, \mathbf{u}_k) + D, \\ k \in \mathbb{K} \end{array} \right\} \tag{4.4}$$

where $\mathbb{K}$ is a finite set of indices. Because g is known, we can construct an auxiliary data set $\mathcal{D}'$ as follows:

$$\mathcal{D}' = \{(\mathbf{x}_k, \mathbf{u}_k, \mathbf{y}_k) |\; \mathbf{y}_k = \mathbf{x}'_k - g(\mathbf{x}_k, \mathbf{u}_k),\; k \in \mathbb{K}\}.$$

The new auxiliary data set can be seen as a set of data generated by a monotone map of the form (2.4) or by a map of the form (3.1).

Therefore, we can use the approach presented in the previous sections to compute a simulating map $S : X \times U \rightrightarrows \overline{\mathbb{R}}^{n_x}$ of data $\mathcal{D}'$. Then, a data-driven model of system (4.3) can be defined as follows

$$\mathbf{x}(\tau + 1) \in S(\mathbf{x}(\tau), \mathbf{u}(\tau)) + g(\mathbf{x}(\tau), \mathbf{u}(\tau)). \tag{4.5}$$

In the following, we formally relate the behaviors of (4.3) and (4.5). We define transition systems $T_{sys} = (X, U, \Delta_{sys}, Y, H)$ and $T_{data} = (X, U, \Delta_{sys}, Y, H)$ associated to (4.3) and (4.5) where the set of states X and inputs U are the same as in (4.3) and (4.5). The transition relation Δ_{sys} is defined as follows, for all $\mathbf{x} \in X$:

$$\mathbf{u} \in \mathsf{enab}_{\Delta_{sys}}(\mathbf{x}) \iff f(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) + D \subseteq X, \tag{4.6}$$

and

$$\forall \mathbf{u} \in \mathsf{enab}_{\Delta_{sys}}(\mathbf{x}),\; \Delta_{sys}(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) + D. \tag{4.7}$$

Similarly, the transition relation Δ_{data} is defined as follows, for all $\mathbf{x} \in X$:

$$\mathbf{u} \in \mathsf{enab}_{\Delta_{data}}(\mathbf{x}) \iff S(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) \subseteq X, \tag{4.8}$$

and

$$\forall \mathbf{u} \in \mathsf{enab}_{\Delta_{data}}(\mathbf{x}),\; \Delta_{data}(\mathbf{x}, \mathbf{u}) = S(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}). \tag{4.9}$$

Let us remark that (4.6) and (4.8) ensure that an input $\mathbf{u}$ is enabled at state $\mathbf{x}$ only if it is guaranteed that the next state of (4.3) and (4.5) belongs to the set of states X. The set of outputs Y and the output map H are left unspecified. They can be chosen arbitrarily but are assumed to be the same for T_{sys} and T_{data}. One can, for instance, choose $Y = X$ and H to be the identity map.

Proposition 15. *Let $S \in \mathbb{S}_{\mathcal{D}'}$, then for any choice of set of outputs Y and of output map $H : X \to Y$, $T_{data} \preceq_{\mathcal{AS}} T_{sys}$.*

Proof. Let us show that

$$R = \{(\mathbf{x}_{data}, \mathbf{x}_{sys}) \in X \times X \mid \mathbf{x}_{data} = \mathbf{x}_{sys}\}$$

is an alternating simulation relation from T_{data} to T_{sys}. The first two conditions of alternating simulation follow directly from the form of R and from the fact that T_{sys} and T_{data} have the same sets of states X and of outputs Y, and the same output maps H.

Since $S \in \mathbb{S}_{\mathcal{D}'}$, it follows from Definition 6 that

$$\forall \mathbf{x} \in X, \mathbf{u} \in U,\ f(\mathbf{x}, \mathbf{u}) + D \subseteq S(\mathbf{x}, \mathbf{u}).$$

Hence, $S(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) \subseteq X$ implies $f(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) + D \subseteq X$. Therefore, from (4.6) and (4.8), we have for all $\mathbf{x} \in X$, $\text{enab}_{\Delta_{data}}(\mathbf{x}) \subseteq \text{enab}_{\Delta_{sys}}(\mathbf{x})$. Moreover, from (4.7) and (4.9), for all $\mathbf{u} \in \text{enab}_{\Delta_{data}}(\mathbf{x})$, it holds $\Delta_{sys}(\mathbf{x}, \mathbf{u}) \subseteq \Delta_{data}(\mathbf{x}, \mathbf{u})$.

Let us now show the third condition of alternating simulation. Let us consider $(\mathbf{x}_{sys}, \mathbf{x}_{data}) \in R$, then $\mathbf{x}_{sys} = \mathbf{x}_{data}$. Let $\mathbf{u}_{data} \in \text{enab}_{\Delta_{data}}(\mathbf{x}_{data})$, then for $\mathbf{u}_{sys} = \mathbf{u}_{data}$, we have $\mathbf{u}_{sys} \in \text{enab}_{\Delta_{sys}}(\mathbf{x}_{sys})$. Moreover, $\Delta_{sys}(\mathbf{x}_{sys}, \mathbf{u}_{sys}) \subseteq \Delta_{data}(\mathbf{x}_{data}, \mathbf{u}_{data})$. Therefore, for all $\mathbf{x}'_{sys} \in \Delta_{sys}(\mathbf{x}_{sys}, \mathbf{u}_{sys})$, there exists $\mathbf{x}'_{data} \in \Delta_{data}(\mathbf{x}_{data}, \mathbf{u}_{data})$ satisfying $\mathbf{x}'_{sys} = \mathbf{x}'_{data}$, and hence $(\mathbf{x}'_{sys}, \mathbf{x}'_{data}) \in R$. □

From the previous result, it follows that the data-driven model T_{data} can be used to synthesize controllers that can be refined into controllers for the partially unknown system T_{sys}, with formal guarantees of correctness.

4.4.1 . Symbolic abstraction

We now go one step further by computing symbolic abstractions of T_{data}. This will allow us to use discrete controller synthesis techniques to control the system with formal guarantees on the closed-loop behavior. Let us assume that the sets of states and inputs X and U are closed intervals of $\overline{\mathbb{R}}^{n_x}$ and $\overline{\mathbb{R}}^{n_u}$ and let $(X_{\mathbf{q}})_{\mathbf{q} \in Q}$, $(U_{\mathbf{p}})_{\mathbf{p} \in P}$ be given rectangular partitions of X and U as defined for Z in Section 2.2.1. We define a quantization function $\phi : X \times U \to Q \times P$ associated to these finite partitions as follows,

$$\begin{aligned} &\forall (\mathbf{x}, \mathbf{u}) \in X \times U,\ \forall (\mathbf{q}, \mathbf{p}) \in Q \times P, \\ &\qquad \phi(\mathbf{x}, \mathbf{u}) = (\mathbf{q}, \mathbf{p}) \iff \mathbf{x} \in X_{\mathbf{q}},\ \mathbf{u} \in U_{\mathbf{p}}. \end{aligned} \tag{4.10}$$

Then, we can use the approach presented in Section 2.2.1 to compute a map $\sigma : Q \times P \rightrightarrows \overline{\mathbb{R}}^{n_x}$ such that $\sigma \circ \phi$ is a simulating map of $\mathcal{D}'$.

Let us assume that for all $(\mathbf{q}, \mathbf{p}) \in Q \times P$, we can compute subsets $G(\mathbf{q}, \mathbf{p}) \subseteq \overline{\mathbb{R}}^{n_x}$ such that

$$\forall \mathbf{x} \in X_{\mathbf{q}},\ \forall\, \mathbf{u} \in U_{\mathbf{p}},\ g(\mathbf{x}, \mathbf{u}) \in G(\mathbf{q}, \mathbf{p}). \tag{4.11}$$

Such sets can be computed for instance using interval analysis [37] or using approaches based on mixed monotonicity or on growth bounds [54].

We define a symbolic transition system $T_{symb} = (Q, P, \Delta_{symb}, Y, H_{symb})$ where the set of states and inputs are given by the partitions index sets Q and P, and the transition relation Δ_{symb} is defined as follows, for all $\mathbf{q} \in Q$:

$$\mathbf{p} \in \mathsf{enab}_{\Delta_{symb}}(\mathbf{q}) \iff \sigma(\mathbf{q}, \mathbf{p}) + G(\mathbf{q}, \mathbf{p}) \subseteq X, \tag{4.12}$$

and

$$\begin{aligned} &\forall \mathbf{p} \in \mathsf{enab}_{\Delta_{symb}}(\mathbf{q}), \\ &\quad \Delta_{symb}(\mathbf{q}, \mathbf{p}) = \left\{ \mathbf{q}' \in Q \middle| \, (\sigma(\mathbf{q}, \mathbf{p}) + G(\mathbf{q}, \mathbf{p})) \cap X'_{\mathbf{q}} \neq \emptyset \right\}. \end{aligned} \tag{4.13}$$

We define the set of outputs to be $Y = Q$ and the output map H_{symb} to be the identity map.

Theorem 3. *Let $S = \sigma \circ \phi \in \mathbb{S}_{\mathcal{D}'}$, let T_{sys} and T_{data} be defined for the set of outputs $Y = Q$ and the output map $H : X \to Q$, given by*

$$\forall \mathbf{x} \in X, \ \forall \mathbf{q} \in Q, \ H(\mathbf{x}) = \mathbf{q} \iff \mathbf{x} \in X_{\mathbf{q}}. \tag{4.14}$$

Then, the following relation holds:

1. $T_{symb} \preceq_{\mathcal{AS}} T_{data} \preceq_{\mathcal{AS}} T_{sys}$.
2. *If $g = 0$, then $T_{symb} \cong_{\mathcal{AS}} T_{data}$.*

Proof. The fact that $T_{data} \preceq_{\mathcal{AS}} T_{sys}$ follows directly from Proposition 15. Then, let us show that

$$R = \{(\mathbf{q}, \mathbf{x}) \in Q \times X \mid \mathbf{x} \in X_{\mathbf{q}}\}$$

is an alternating simulation relation from T_{symb} to T_{data}. Let $\mathbf{q} \in Q$ and let $\mathbf{x} \in X_{\mathbf{q}}$, then $(\mathbf{q}, \mathbf{x}) \in R$ and the first condition of alternating simulation holds. Let $(\mathbf{q}, \mathbf{x}) \in R$, then $\mathbf{x} \in X_{\mathbf{q}}$, which by (4.14) gives $H(\mathbf{x}) = \mathbf{q}$. Since $H_{symb}(\mathbf{q}) = \mathbf{q}$, the second condition of alternating simulation holds.

Let us now show the third condition of alternating simulation. Let $(\mathbf{q}, \mathbf{x}) \in R$ and $\mathbf{p} \in \mathsf{enab}_{\Delta_{symb}}(\mathbf{q})$, then choose $\mathbf{u} \in U_{\mathbf{p}}$. Since $S = \sigma \circ \phi$, we have $S(\mathbf{x}, \mathbf{u}) = \sigma(\mathbf{q}, \mathbf{p})$, and by (4.11) we have $g(\mathbf{x}, \mathbf{u}) \in G(\mathbf{q}, \mathbf{p})$. Hence,

$$S(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) \subseteq \sigma(\mathbf{q}, \mathbf{p}) + G(\mathbf{q}, \mathbf{p}). \tag{4.15}$$

Then, it follows from (4.8) and (4.12) that $\mathbf{u} \in \mathsf{enab}_{\Delta_{data}}(\mathbf{x})$. Let $\mathbf{x}' \in \Delta_{data}(\mathbf{x}, \mathbf{u})$, from (4.9) and (4.15), we get $\mathbf{x}' \in \sigma(\mathbf{q}, \mathbf{p}) + G(\mathbf{q}, \mathbf{p})$. Let $\mathbf{q}' \in Q$ such that $\mathbf{x}' \in X_{\mathbf{q}'}$, from (4.13), we get that $\mathbf{q}' \in \Delta_{symb}(\mathbf{q}, \mathbf{p})$. Moreover, since $\mathbf{x}' \in X_{\mathbf{q}'}$, $(\mathbf{q}', \mathbf{x}') \in R$. Hence, we proved that $T_{symb} \preceq_{\mathcal{AS}} T_{data}$.

Now, let us assume that for all $\mathbf{x} \in X$, and $\mathbf{u} \in U$, $g(\mathbf{x}, \mathbf{u}) = 0$. Then, (4.11) holds with $G(\mathbf{q}, \mathbf{p}) = \{0\}$, for all $\mathbf{q} \in Q$ and $\mathbf{p} \in P$. Let us show that in that case, R^{-1} is an alternating simulation relation from T_{data} to T_{symb}. Let $\mathbf{x} \in X$,

since $(X_{\mathbf{q}})_{\mathbf{q}\in Q}$ is a partition of X, there exists $\mathbf{q} \in Q$ such that $\mathbf{x} \in X_{\mathbf{q}}$. Hence, $(\mathbf{x}, \mathbf{q}) \in R^{-1}$ and the first condition of alternating simulation holds. The proof that the second condition holds for R^{-1} is the same as for R.

Then, let us show the third condition of alternating simulation. Let $(\mathbf{x}, \mathbf{q}) \in R^{-1}$ and $\mathbf{u} \in \text{enab}_{\Delta_{data}}(\mathbf{x})$, then there exists $\mathbf{p}$ such that $\mathbf{u} \in U_{\mathbf{p}}$. Since $S = \sigma \circ \phi$, we have $S(\mathbf{x}, \mathbf{u}) = \sigma(\mathbf{q}, \mathbf{p})$. Moreover, since $g(\mathbf{x}, \mathbf{u}) = 0$ and $G(\mathbf{q}, \mathbf{p}) = \{0\}$, we get from (4.8) and (4.12) that $\mathbf{p} \in \text{enab}_{\Delta_{symb}}(\mathbf{q})$. Let $\mathbf{q}' \in \Delta_{symb}(\mathbf{q}, \mathbf{p})$, from (4.13) we get that $\sigma(\mathbf{q}, \mathbf{p}) \cap X_{\mathbf{q}'} \neq \emptyset$. Hence, $S(\mathbf{x}, \mathbf{u}) \cap X_{\mathbf{q}'} \neq \emptyset$. Then, let $\mathbf{x}' \in S(\mathbf{x}, \mathbf{u}) \cap X_{\mathbf{q}'}$. From (4.9), $\mathbf{x}' \in \Delta_{data}(\mathbf{x}, \mathbf{u})$. Moreover $(\mathbf{x}', \mathbf{q}') \in R^{-1}$. Hence, we proved that $T_{data} \cong_{\mathcal{AS}} T_{symb}$. □

Theorem 3 shows that the symbolic abstraction T_{symb} can be used to synthesize controllers for the data-driven system T_{data} and hence also for the partially unknown system T_{sys}. Note that the fact that T_{symb} has only a finite number of state and input values allows us to use algorithmic techniques to synthesize controllers for various specifications such as safety, reachability, or attractivity [30] or even more complex specifications expressed e.g. in linear temporal logic [9] or using hybrid automata [69]. Due to the fact that there is an alternating simulation relation between T_{sys} and T_{symb} the correct-by-design synthesized controller will guarantee the desired closed-loop behavior [71]. Moreover, when T_{sys} is fully unknown and monotone (i.e. when $g = 0$), then Theorem 3 shows that working with the symbolic abstraction T_{symb} does not bring any conservatism compared to working with the data-driven model T_{data}.

4.5 . Numerical examples

In this section, we present several examples to illustrate the use of the proposed approach. We first show how to use the approach to find an abstraction of a system with unknown monotone dynamics. We use the abstraction to find the maximum controllable set of the system. The second example shows how to use the approach to find an abstraction of a system with unknown dynamics with bounded derivatives. This example also shows how to use abstraction to synthesize a safety controller for a system with unknown dynamics, and it will have a continuation in the next chapter where we introduce an estimation algorithm for the unknown dynamics. In the third example, we study the chaotic Lorenz system. We use the algorithm introduced in Section 3.3 to find the bounds on the derivatives of the system. Then, we use the algorithm introduced in Section 4.4 to find an abstraction of the system.

4.5.1 . Cruise control problem

The first example showcases a system that can be seen as a sum of a known function and an unknown monotone function. We find the abstraction representing the system. Then, we use this abstraction to synthesize a safety controller.

Let us consider two vehicles moving in one lane on an infinite straight road. The leader is uncontrollable (vehicle 2) whereas the follower is controllable (vehicle 1). A discrete-time model of this setup is given by equations:

$$\begin{aligned} d(k+1) &= d(k) + (v_f(k) - v_l(l))T_0 \\ v_f(k+1) &= v_f(k) + \gamma(u(k), v_f(k))T_0 \\ v_l(k+1) &= v_l(k) + d(k)T_0 \end{aligned} \tag{4.16}$$

Here u is the control input, and d is the signed distance between the vehicles. v_l, v_f are the velocities of the leader and follower, respectively. The function γ represents the follower vehicle acceleration caused by the control input and the friction forces acting upon it. The term $d(k) \in [\underline{d}, \overline{d}]$ accounts for uncertainty in the leader velocity and is considered a disturbance. Model (4.16) can be seen as the sum of a known part which can be inferred from the physics of the system, and an unknown or hard-to-model part, namely the function $\gamma(u(k), v_f(k))T_0$. We are also able to study the monotonicity of the function $\gamma(u(k), v_f(k))T_0$ starting from the physics of the system. The acceleration of the car will increase with the increase of the input, and the friction forces will increase with the increase of the velocity. Therefore, by making the change of variable $v_f' = -v_f$ we can apply the algorithm introduced in this work to find an abstraction of the system.

Function γ is given as follows

$$\gamma(u, v) = \; u - M_C^{-1}(f_0 + f_1 v + f_2 v^2).$$

The vector of parameters $f = (f_0, f_1, f_2) \in \mathbb{R}^3_+$ describes road friction and vehicle aerodynamics whose numerical values are taken from [59]: $f_0 = 51\ N$, $f_1 = 1.2567\ Ns/m$, $f_2 = 0.4342 Ns^2/m^2$. For the car mass, we chose $M_C = 1370\ kg$. Other numerical values related to the model are $X = [-50, -5] \times [0, 30] \times [5, 25]$, $U = [-3, 3], T_0 = 0.7\ s$, $\underline{d} = -\overline{d} = 2\ m/s^2$. To build the abstraction we fixed a partition $K_x^1 = 50$, $K_x^2 = 50$, $K_x^3 = 50$, $K_u^1 = 50$, and sampled a set containing $|\mathbb{K}| = 10^6$ data points. Finding the over-approximation of the unknown monotone part was done in $0.633s$.

To show the usefulness of the calculated abstraction, we use it to synthesize a safety controller. In this example, the leader's vehicle is uncontrollable. Therefore, we considered a specification given as an assume-guarantee contract:

$$\begin{aligned} \forall k \in \mathbb{N}\ v_l(k) \in [\underline{v}_l, \overline{v}_l] \implies \\ \forall k \in \mathbb{N},\ v_f(k) \in [\underline{v}_f, \overline{v}_f] \wedge d(k) \in [\underline{d}, \overline{d}]. \end{aligned} \tag{4.17}$$

If the velocity of the leader remains within the bounds $[\underline{v}_l, \overline{v}_l]$, then the velocity of the follower and the distance between the two vehicles should remain within the bounds $[\underline{v}_f, \overline{v}_f]$ and $[\underline{d}, \overline{d}]$, respectively. The synthesis of symbolic controllers enforcing assume-guarantee contracts such as (4.17) has been considered in [67]. For this example, we chose $\underline{d} = \alpha_1^1$, $\overline{d} = \alpha_{K^1}^1$, $\underline{v}_f = \alpha_1^2$, $\overline{v}_f = \alpha_{K^2}^2$, $\underline{v}_l = \alpha_1^3$, $\overline{v}_f = \alpha_{K^3}^3$. The controllable set of the resulting symbolic controller is shown in Figure 4.4.

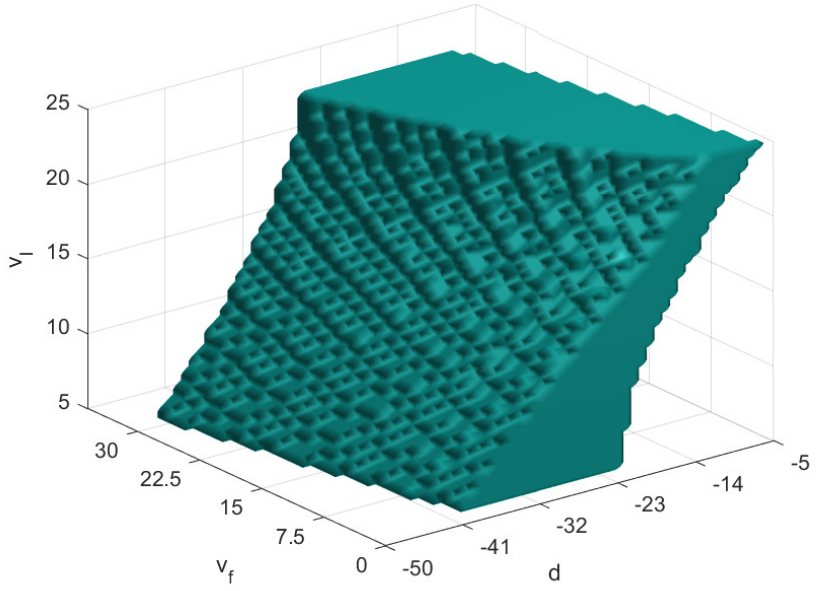


Figure 4.4: Controllable set of the symbolic controller enforcing the assume-guarantee contract (4.17) for system (4.16)

4.5.2 . Path planning

In this path-planning example, we consider a vehicle moving in a two-dimensional environment. We learn the dynamics of a unicycle model by sampling data transitions from it. Then, we use the learned model to drive the vehicle in an environment containing obstacles. To complete this task, we will use tools that will be introduced in the next chapter. Thus, this example will be continued in the next chapter where we introduce an estimation algorithm for the unknown dynamics.

For now, we use the approach introduced in this work to find an abstraction of the learned model. Then, we use the abstraction to synthesize a safety controller for the vehicle.

We consider the unicycle models defined by the following equations

$$\begin{aligned} \dot{x} &= v \cdot \cos(\theta) + d_1 \\ \dot{y} &= v \cdot \sin(\theta) + d_2 \\ \dot{\theta} &= \frac{v}{L} \tan(\delta) + d_3 \end{aligned} \tag{4.18}$$

where x, y are the coordinates of the vehicle, θ is the heading angle, as can be seen in Figure 4.5, L is the length of the vehicle, and we chose the value $L = 0.1$ m. The velocity v and the steering angle δ are considered the input. The goal of this experiment is to drive the vehicle from a starting position to a target position in an environment, as shown in Figure 4.6. The vehicle should maneuver around

an obstacle to reach its target. We consider a 2×5 m area, a 1×0.5 m obstacle positioned at 2.5 m away from the left side of the room.

To build the over-approximation of the dynamics in (4.18), we sampled $|\mathbb{K}| = 10^6$ data points using a black box simulator of (4.18). We chose for the states, inputs, and disturbance intervals the following $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}], \delta \in [-1, 1], v \in [0.025, 0.5], \mathbf{d} \in [-0.05, 0.05] \times [-0.05, 0.05] \times [-0.05, 0.05]$, and partitioned those intervals uniformly into 30 cells each. We only use θ, δ, and v to study the dynamics of the unicycle model as the function representing this model clearly does not depend on the states x and y. The resulting over-approximation can be seen in Figure 4.7. The execution time to find the over-approximation is $t_{ov} = 2.78$ s.

Then, we use the over-approximation model to build a symbolic abstraction of the behavior of the vehicle in the environment as follows: We partitioned the x-axis into 100 sections and the y-axis into 40 sections uniformly. Then, we found the reachable set of the vehicle starting from each cell in the partition using the calculated over-approximation. We use the Euler method for discretizing. The symbolic abstraction is then used to find the maximal control invariant inside the safe region and the set of safe actions at each cell. The speed is chosen to be

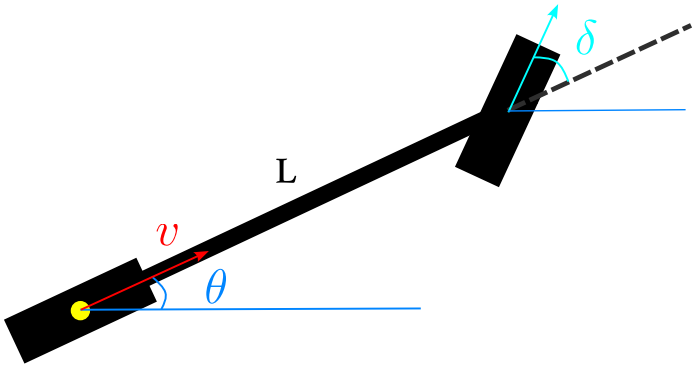


Figure 4.5: The unicycle model. The reference point is at the center of the rear axle.

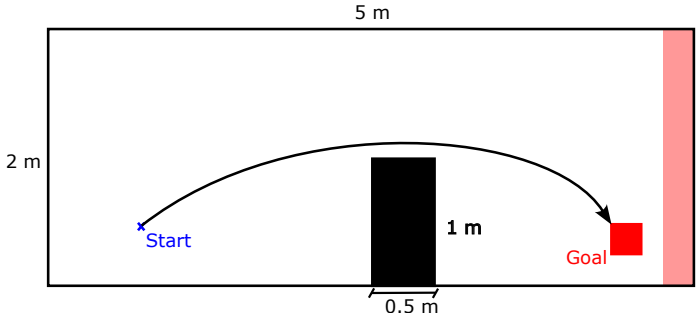
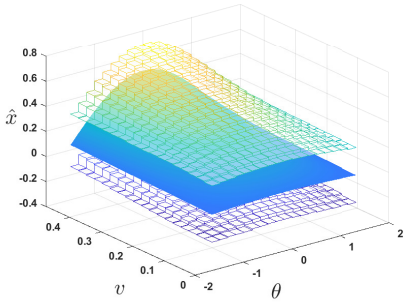
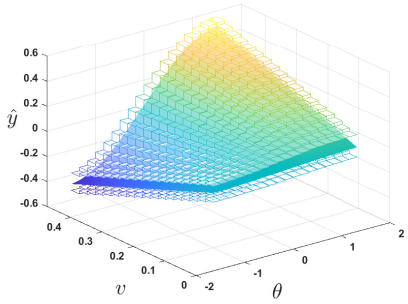


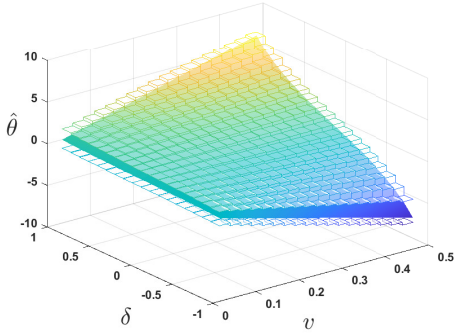
Figure 4.6: The environment where the vehicle should maneuver an obstacle to reach a target position



(a) The over-approximation of the dynamics of $\dot{x}$



(b) The over-approximation of the dynamics of $\dot{y}$



(c) The over-approximation of the dynamics of $\dot{\theta}$

Figure 4.7: Over-approximation of the bicycle dynamics

strictly positive, and hence the vehicle cannot stop. All the walls of the area are considered obstacles except the right wall because the vehicle cannot stop. Figure 4.8 shows the regions where we can find a safety controller for certain values of θ. The execution time to find the maximal controlled invariant is $t_{inv} = 871.88$ s.

4.5.3 . Lorenz system

In the last example, we study the Lorenz system, a system with bounded derivatives. We first compute the bounds on the disturbance and on the derivatives of the function representing the system from the data. Then, we find a data-driven abstraction that we use to synthesize a controller for the system. The controlled Lorenz system has three states and one input. We built the abstraction using a large number of points and a partition with a large number of elements, and we showed that we could do it efficiently.

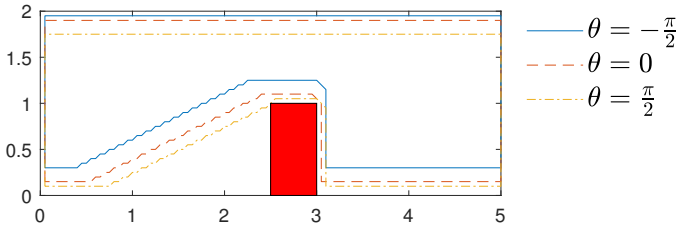


Figure 4.8: The maximal controlled invariant for three values of θ.

The controlled discrete-time Lorenz system is described by the following model

$$\begin{aligned} x(k+1) &= x(k) + \sigma(x(k) - y(k) + d_1(k))T_0 \\ y(k+1) &= y(k) + (\rho\ x(k) - x(k)z(k) - \\ &\qquad y(k) + u(k) + d_2(k))T_0 \\ z(k+1) &= z(k) + (x(k)y(k) - \beta\ z(k) + d_3(k))T_0 \end{aligned} \tag{4.19}$$

For the particular parameter values of $\sigma = 10, \beta = \frac{8}{3}, \rho = 28$ the system behaves chaotically [48]. We studied the system on the sets $X = [-10, 10] \times [-10, 10] \times [-10, 10]$, $U = [-200, 200]$, $\overline{\mathbf{d}} = [0.5, 0.5, 0.5]$. The discretization time was chosen to be $T_0 = 0.01s$. First, we used the algorithm described in Section 3.3 to calculate the bounds on the disturbance and the derivatives. In the first step, we used $5 * 10^4$ data points sampled randomly. In the second step, we sampled 10^6 data points. The results of the calculated bounds on Jacobian matrices and the disturbance are shown below. Subscript $\mathcal{D}$ refers to matrices calculated from data, while their unsubscripted counterparts are the actual matrices we are estimating. The results

are rounded to the third significant digit.

$$\overline{J}^x = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.38 & 0.99 & 0.1 \\ 0.1 & 0.1 & 0.973 \end{bmatrix}, \overline{J}^x_{\mathcal{D}} = \begin{bmatrix} 0.9 & 0.1 & 1.64{*}10^{-6} \\ 0.377 & 0.99 & 9.46{*}10^{-2} \\ 9.09{*}10^{-2} & 9.31{*}10^{-2} & 0.975 \end{bmatrix}$$

$$\underline{J}^x = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.18 & 0.99 & -0.1 \\ -0.1 & -0.1 & 0.973 \end{bmatrix},$$

$$\underline{J}^x_{\mathcal{D}} = \begin{bmatrix} 0.9 & 0.1 & -2.1{*}10^{-7} \\ 0.185 & 0.989 & -9.27{*}10^{-2} \\ -9.3{*}10^{-2} & -9.2{*}10^{-2} & 0.97 \end{bmatrix}$$

$$\overline{J}^u = \begin{bmatrix} 0 \\ 0.01 \\ 0 \end{bmatrix}, \overline{J}^u_{\mathcal{D}} = \begin{bmatrix} -1.9{*}10^{-7} \\ 9.95{*}10^{-3} \\ 2.88{*}10^{-6} \end{bmatrix}$$

$$\underline{J}^u = \begin{bmatrix} 0 \\ 0.01 \\ 0 \end{bmatrix}, \underline{J}^u_{\mathcal{D}} = \begin{bmatrix} -1.9{*}10^{-7} \\ 9.95{*}10^{-3} \\ 4.58{*}10^{-5} \end{bmatrix}$$

$$\overline{\mathbf{d}}{*}T_0 = \begin{bmatrix} 0.005 \\ 0.005 \\ 0.005 \end{bmatrix}, \overline{\mathbf{d}}_{\mathcal{D}} * T_0 = \begin{bmatrix} 0.005 \\ 5.85{*}10^{-2} \\ 5.76{*}10^{-2} \end{bmatrix}$$

The execution time of the first step is $12.4s$, whereas the execution time for the second step is $0.5s$. From the results, we can see that all values but the disturbances added to the second and third states are similar to the actual values. The second and third values of the disturbance are more conservative. If we choose $\beta = 10^{-6}$, then according to Proposition 14, with a probability at least $1 - \beta$, it holds that the probability of finding two points violating the calculated bounds is less than $2 * 10^{-5}$.

We used these values to build the data-driven abstraction. We sampled 10^8 data points. We choose the parameters of the partition as follows $K_x^1 = 100,\ K_x^2 = 100,\ K_x^3 = 100,\ K_u^1 = 100,\ c = 0.05$. The execution time to reach the abstraction was $929.2\ s$. To test the validity of the calculated abstraction, we used it to find the maximal safe controlled invariant of the system. Given a safe set $X_s \subseteq X$, a safe controlled invariant $\mathcal{I}_s$ is a set included in the safe set $\mathcal{I}_s \subseteq X_s$, and that can be rendered invariant using a suitable controller. Using the finite-state abstraction we calculated earlier; we can apply an iterative algorithm to find the maximal safe controlled invariant [71]. We chose the safe set to be $X_s = X$. The resulting maximal control invariant is represented in Figure 4.9.

4.6 . Conclusion

In this chapter, we introduced transition systems; the representation used to model the dynamics of a variety of systems. We then introduced the notion of alternating simulation, which is a relation between two transition systems. We showed that if there exists an alternating simulation relation between two transition systems, then we can use the controller of one system to control the other system. We then introduced the notion of symbolic abstraction, which is a finite-state transition system that over-approximates the behavior of a system. If the symbolic abstraction is an alternating simulation of the system, then we can use the controller of the symbolic abstraction to control the system. We described

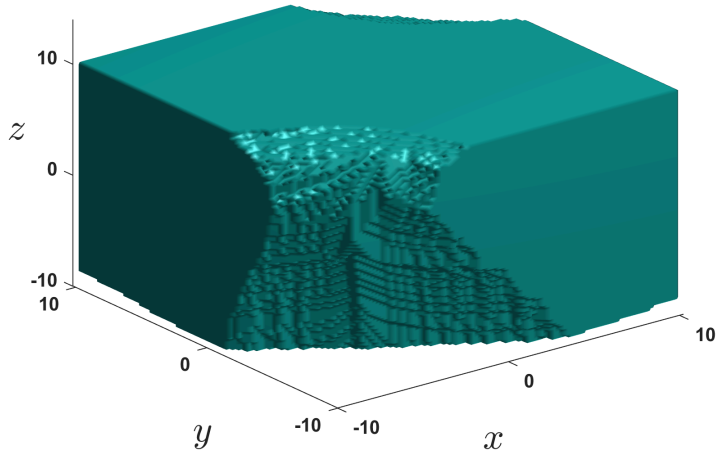


Figure 4.9: The maximal controlled invariant of (4.19) calculated using the data-driven abstraction

the type of discrete controller synthesis problems that can be solved using symbolic abstractions. Examples of such problems are safety and reachability. We then introduced the notion of data-driven abstraction, which is calculated using the data-driven interval-valued over-approximation introduced in Chapter 2 and Chapter 3. Several examples were presented to show the validity of the approach.

5 - Safe Learning

Ensuring safety is a crucial requirement for any control system, and it is still an open problem for many data-driven control approaches. One can find in the literature many approaches to ensure safety, such as using barrier functions [4], or using reachability analysis [2]. Also, in the field of learning-based control, one can find many approaches to ensure safety, such as using safe exploration [11], or using safe reinforcement learning [27].

In the previous sections, we studied how we can learn over-approximation of unknown dynamical systems. Those over-approximations are then used to find data-driven finite-state abstractions, and thus to build controllers. As each discrete input of the controller, and we focus here on safety controllers, is mapped to a set of continuous inputs, Each one of those input sets can be applied to the system to ensure safety. And thus, the question of which input to choose arises.

In this chapter, we introduce a safe learning-based MPC scheme. We use the data set sampled from a given system to build two models of this system. The first model is the finite-state abstraction of the system, and it is used to ensure strong safety requirements, whereas the second is used to find a controller that minimizes a cost function and thus fulfills some performance requirements. To find the second model, we use the piecewise multi-affine estimation of the system's dynamics. The idea of decoupling safety and performance by using two models is introduced in [7], where one model is used to ensure safety using tube MPC, and the other model, updated online, is used to minimize a cost function. By contrast, we do not impose any linear structure on the studied system.

Model Predictive Control (MPC) has established its position as a vital tool for many control problems, research to incorporate data-driven methods into MPC has been surging recently [34, and references therein], with topics such as safety and robustness still remaining largely open topics [78, 43]. While some use input-output data to find optimal control policies directly, eliminating the need for building a model [23, 10], others use the data to find data-driven models and then apply the MPC scheme [7, 38]. A comparison between these direct and indirect approaches is provided in [44].

This chapter is organized as follows. In Section 5.1, we introduce a safe learning-based MPC scheme and prove its safety and well-posedness. In Section 5.2, we introduce the piecewise multi-affine compatible estimation of the system's dynamics. This estimation is used to build the safe learning-based MPC scheme. Section 5.3 deals with the problem of updating the estimation of the system's dynamics locally. Whereas Section 5.4 deals with the problem of updating the model of the system's dynamics and the safe controller online while operating the system. In Section 5.5, we illustrate the proposed scheme on a numerical example. Finally, in Section 5.6, we conclude the chapter.

5.1 . MPC with safety guarantees

In this section, we introduce the safe learning-based MPC scheme. We first introduce the problem formulation and the assumptions we make on the system. Then, we introduce the over-approximation of the system's dynamics and the safety controller. Finally, we introduce the safe learning-based MPC scheme and prove its safety and well-posedness.

We consider a discrete-time system of the form.

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{d}(t) \tag{5.1}$$

where $\mathbf{x}(t) \in X \subseteq \mathbb{R}^{n_x}$ is the state of the system, $\mathbf{u}(t) \in U \subseteq \mathbb{R}^{n_u}$ is the input, $\mathbf{d}(t) \in D \subseteq \mathbb{R}^{n_x}$ is the disturbance, and $f : X \times U \to X$ is the unknown nonlinear function representing the dynamics of the system. Similar to previous chapters, we assume the following assumptions on the system.

Assumption 1. *The unknown function f has bounded derivatives i.e. for all $\mathbf{x} \in X$, $\mathbf{u} \in U$:*

$$\frac{\partial f^i}{\partial x^j}(\mathbf{x}, \mathbf{u}) \in [\underline{\alpha}_{ij}, \overline{\alpha}_{ij}],\ i, j \in \{1, \dots, n_x\},$$
$$\frac{\partial f^i}{\partial u^j}(\mathbf{x}, \mathbf{u}) \in [\underline{\beta}_{ij}, \overline{\beta}_{ij}],\ i \in \{1, \dots, n_x\},\ j \in \{1, \dots, n_u\},$$

where the bounds $\underline{\alpha}_{ij}, \overline{\alpha}_{ij}, \underline{\beta}_{ij}, \overline{\beta}_{ij} \in \mathbb{R}$ are assumed to be known. The set of disturbances $D = [\underline{\mathbf{d}}, \overline{\mathbf{d}}]$, is a bounded interval with known bounds $\underline{\mathbf{d}}, \overline{\mathbf{d}} \in \mathbb{R}^{n_x}$ and such that $0 \in D$.

Assumption 2. *We are given a set of data generated from the dynamic system (5.1):*

$$\mathcal{D} = \{(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}'_k) \mid \mathbf{x}'_k \in f(\mathbf{x}_k, \mathbf{u}_k) + D, k \in \mathbb{K}\}$$

where $\mathbb{K}$ is a finite set of indices.

Collecting the data set $\mathcal{D}$ can be done in various fashion. One can for instance sample the dynamics of the system randomly using independent samples. This can be done easily if one can use a black box model of (5.1) to generate independent simulations. However, our approach does not require the use of independent samples. Actually, they can be collected by recording the evolution of the true system over a given period of time. In that case, we would have $\mathbf{x}'_k = \mathbf{x}_{k+1}$.

We recall that an over-approximation map F of the true function f is a set-valued map $S : X \times U \rightrightarrows X$ satisfying

$$f(\mathbf{x}, \mathbf{u}) + D \subseteq F(\mathbf{x}, \mathbf{u}), \forall \mathbf{x} \in X, \forall \mathbf{u} \in U,$$

Let us also recall that safety controllers, calculated using the over-approximation, can attribute to each state $\mathbf{x} \in \mathrm{dom}(C_{\mathsf{safe}})$ a set of allowed inputs. To find the one input (out of several safe ones) that minimizes a receding horizon cost function, we build a single-valued estimation of the true function

Definition 14. *An estimation $\hat{f} : X \times U \to X$ of the true function f is said to be compatible with the over-approximation F if*

$$\hat{f}(\mathbf{x}, \mathbf{u}) + D \subseteq F(\mathbf{x}, \mathbf{u}), \forall \mathbf{x} \in X, \forall \mathbf{u} \in U. \tag{5.2}$$

Section 5.2 deals with finding this compatible estimation. Now, we have everything we need to introduce the safe learning-based MPC scheme. First, we find the data-driven over-approximation F. Then, we use it to find the safe controller C_{safe}. After that, we build an estimation of the dynamics $\hat{f}$ compatible with the over-approximation. The following theorem shows how to use those models to implement a learning-based MPC program to meet the strict safety requirements while enforcing a soft performance optimization by minimizing the estimation of a cost function J using an estimate $\hat{f}$.

Theorem 4. *Given a stage costs $J_k : X \times U \to \mathbb{R},\ k \in \{1, \dots, N-1\}$ and a terminal cost $J_N : X \to \mathbb{R}$, starting from $\mathbf{x}(0) \in$ dom(C_F), consider the trajectory of* (5.1) *with $\mathbf{u}(t) = \mathbf{u}(0|t)$ where $\mathbf{u}(0|t)$ is obtained by solving the optimization problem below:*

$$\begin{aligned}
&\min_{\mathbf{u}(0|t),\dots\mathbf{u}(N-1|t)} \sum_{i=0}^{N-1} J_i(\mathbf{x}(i|t), \mathbf{u}(i|t)) + J_N(\mathbf{x}(N|t)) \\
&\textit{subject to} \quad \mathbf{x}(i+1|t) = \hat{f}(\mathbf{x}(i|t), \mathbf{u}(i|t)), \\
&\qquad\qquad \forall i \in \{0, \dots, N-1\} \\
&\qquad\qquad \mathbf{x}(i|t) \in X_s,\ \forall i \in \{0, \dots, N\} \\
&\qquad\qquad \mathbf{u}(0|t) \in C_{\textit{safe}}(\mathbf{x}(t)) \\
&\qquad\qquad \mathbf{x}(0|t) = \mathbf{x}(t)
\end{aligned} \tag{5.3}$$

Then, for all $t \in \mathbb{N}$, $\mathbf{x}(t) \in X_s$ and (5.3) *admits a feasible solution, i.e. the closed-loop system is safe and well-posed.*

Proof. Let $t \in \mathbb{N}$, and let us assume that $\mathbf{x}(t) \in \text{dom}(C_{\text{safe}})$.

Let us first show that the optimization problem (5.3) has a feasible solution. At each prediction stage $i \in \{0, \dots, N-1\}$, let us choose an input $\mathbf{u}(i|t) \in C_{\text{safe}}(\mathbf{x}(i|t))$ then from (5.2),

$$\mathbf{x}(i+1|t) = \hat{f}(\mathbf{x}(i|t), \mathbf{u}(i|t)) \in F(\mathbf{x}(i|t), \mathbf{u}(i|t)).$$

Since $\mathbf{x}(0|t) = \mathbf{x}(t) \in \text{dom}(C_{\text{safe}})$, we have, according to the second item of Definition 12, that for all $i \in \{0, \dots, N-1\}$, $\mathbf{x}(i+1|t) \in \text{dom}(C_{\text{safe}})$. Then, it follows from the first item of Definition 12, that for all $i \in \{0, \dots, N\}$, $\mathbf{x}(i|t) \in X_s$ and the optimization problem is feasible.

Then, by (5.3), we have that $\mathbf{u}(t) = \mathbf{u}(0|t) \in C_{\text{safe}}(\mathbf{x}(t))$. From (4.15), we get that

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{w}(t) \in F(\mathbf{x}(t), \mathbf{u}(t)).$$

From the second item of Definition 12, it follows that $\mathbf{x}(t+1) \in \text{dom}(C_{\text{safe}})$. Then, starting from $\mathbf{x}(0) \in \text{dom}(C_F)$, we have by induction that for all $t \in \mathbb{N}$, $\mathbf{x}(t) \in \text{dom}(C_F)$ and (5.3) admits a feasible solution. Moreover, from the first item of Definition 12, we get that for all $t \in \mathbb{N}$, $\mathbf{x}(t) \in X_s$. □

5.2 . Piecewise multi-affine estimation

In this section, we demonstrate how to build a single-valued piecewise compatible estimation of the system's dynamics using the class of multi-affine functions. This class of functions was studied on n-dimensional intervals in [8]. We make use of this study to build a piecewise multi-affine estimation compatible with the over-approximation introduced in the previous section. The piecewise estimation is defined on the same partition used to build the over-approximation.

5.2.1 . Piecewise multi-affine functions

Now, we introduce the class of piecewise multi-affine functions that we will use to estimate the unknown function. In this section and to simplify the notations, we will note $Z = X \times U$ and $\mathbf{z} = (\mathbf{x}, \mathbf{u})$.

Definition 15. *A multi-affine function $g : Z \to \mathbb{R}^m$, $Z \subseteq \mathbb{R}^n$ is a function of the form*

$$g(z^1, \dots, z^n) = \sum_{i_1,\dots,i_n \in \{0,1\}} c_{i_1,\dots,i_n} (z^1)^{i_1} \dots (z^n)^{i_n} \tag{5.4}$$

where $c_{i_1,\dots,i_n} \in \mathbb{R}^m$ for all $i_1, \dots, i_n \in \{0,1\}$

In the case where Z is an interval; $Z = [\underline{\mathbf{z}}, \overline{\mathbf{z}}]$, $\underline{\mathbf{z}}, \overline{\mathbf{z}} \in \mathbb{R}^n$, we denote the set of vertices of this interval by V

Let $\xi_i : \{\underline{z}^i, \overline{z}^i\} \to \{0,1\}$ for all $i \in \{1, \dots, n\}$ denote the indicator function

$$\xi_i(\underline{z}^i) = 0 \quad \xi_i(\overline{z}^i) = 1 \quad \forall i \in \{1, \dots, n\}.$$

The following proposition states that any multi-affine function defined on an interval can be written as a linear combination of the values of the function on the vertices of this interval.

Proposition 16 (see [8, Proposition 1]). *Let Z be an n-dimensional interval, $g : Z \to \mathbb{R}^m$ a multi-affine function such that, for all $\mathbf{v} = (v^1, \dots, v^n) \in V$ we have $g(v^1, \dots, v^n) = \mathbf{y}_\mathbf{v}$. Then, for all $\mathbf{z} = (z^1, \dots, z^n) \in Z$ the function g is uniquely given by*

$$g(\mathbf{z}) = \sum_{\mathbf{v} \in V} \prod_{i=1}^{n} \left(\frac{z^i - \underline{z}^i}{\overline{z}^i - \underline{z}^i} \right)^{\xi_i(v^i)} \left(\frac{\overline{z}^i - z^i}{\overline{z}^i - \underline{z}^i} \right)^{1-\xi_i(v^i)} \mathbf{y}_\mathbf{v}. \tag{5.5}$$

As a consequence of Proposition 16, we can estimate a multi-affine function on a given interval by estimating the function's values on the vertices of the interval.

Lemma 2 (see [8, Lemma 2]). *Let $\mathbf{s} \in \mathbb{R}^m$ and $d \in \mathbb{R}$. Then, $\mathbf{s}^T g(\mathbf{z}) \bowtie d$ for all $\mathbf{z} \in Z$ if and only if $\mathbf{s}^T g(\mathbf{v}) \bowtie d$, for all $\mathbf{v} \in V$, where $\bowtie$ stands for any of $<, \leq, =, \geq, >$.*

Given a partition $(R_{\mathbf{q}})_{\mathbf{q}\in Q}$ of the interval $Z \subseteq \mathbb{R}^n$, we denote the vertices of an interval $R_{\mathbf{q}}$ by $V_{\mathbf{q}}$. A function $g : Z \to \mathbb{R}^m$ is piecewise multi-affine if for all $\mathbf{q} \in Q$ the function is multi-affine on $R_{\mathbf{q}}$.

The following proposition establishes that if a piecewise multi-affine function is continuous on the grid points of the partition, then it is continuous everywhere

Proposition 17. *If a piecewise multi-affine function $g : Z \to \mathbb{R}^m$ is continuous on the grid points of the partition $(R_{\mathbf{q}})_{\mathbf{q}\in Q}$:*

$$\lim_{\mathbf{z}\to\mathbf{v}} g(\mathbf{z}) = g(\mathbf{v}) \quad \forall \mathbf{q} \in Q, \mathbf{v} \in V_{\mathbf{q}}$$

then g is continuous for all $\mathbf{z} \in Z$.

Proof. To establish the continuity of piecewise multi-affine functions, it is sufficient to study them on the shared facets of neighboring cells in the partition because they are defined as multi-affine functions on the interior of those cells; and multi-affine functions are continuous. The facets of an interval $R_{\mathbf{q}} = [\underline{\mathbf{z}}_{\mathbf{q}}, \overline{\mathbf{z}}_{\mathbf{q}}]$, $\underline{\mathbf{z}}_{\mathbf{q}}, \overline{\mathbf{z}}_{\mathbf{q}} \in \mathbb{R}^n$ are given by

$$E_{R_{\mathbf{q}},w^j} = R_{\mathbf{q}} \cap \{\mathbf{z} \in \mathbb{R}^n \mid z^j = w^j\}$$

for all $w^j \in \{\underline{z}^j, \overline{z}^j\}, j \in \{1, \ldots, n\}$ which implies according to (5.4) that $g(z^1, \ldots, w^j, \ldots, z^n)$ is also a multi-affine function.

We also have that the facets of the n-dimensional interval are $(n-1)$-dimensional intervals, which implies that the expression of the function g on a given facet is of the form (5.5). Therefore, as the shared facets of two neighboring intervals in the partition have the same vertices, then the limit of the function g from the two neighboring intervals on the shared facet will be the same. □

Based on this result, we can see that to estimate a continuous piecewise multi-affine function on a given partition, we only need to estimate its values on the vertices of the partition.

5.2.2 . Compatible estimation

Now we will describe how to build a piecewise multi-affine estimation function $\hat{f}$ of the system's dynamics compatible with the over-approximation calculated in the previous section.

Starting from the given finite rectangular partitions $(X_{\mathbf{q}})_{\mathbf{q}\in Q}$, $(U_{\mathbf{p}})_{\mathbf{p}\in P}$ of X and U, and Assumption 2, each transition triple $(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}'_k), k \in \mathbb{K}$ allows us to write the following equation

$$\mathbf{x}'_k = \hat{f}_{\mathbf{q},\mathbf{p}}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{e}(\mathbf{x}_k, \mathbf{u}_k)$$

where $(\mathbf{q}, \mathbf{p}) = \phi(\mathbf{x}_k, \mathbf{u}_k)$. The function $\hat{f}_{\mathbf{q},\mathbf{p}} : X_{\mathbf{q}} \times U_{\mathbf{p}} \to X$ is the multi-affine estimation of the true function on the interval $X_{\mathbf{q}} \times U_{\mathbf{p}}$, and the vector $\mathbf{e}(\mathbf{x}_k, \mathbf{u}_k)$ represents the residuals of the estimation.

According to (5.5), we can rewrite $\hat{f}$ as a linear combination of the estimated function on the vertices of the interval $\hat{\mathbf{y}}_{\mathbf{v}_1}, \ldots, \hat{\mathbf{y}}_{\mathbf{v}_{2^{n_x+n_u}}}$, where $\mathbf{v}_1, \ldots, \mathbf{v}_{2^{n_x+n_u}} \in V_{\mathbf{q},\mathbf{p}}$ are the vertices of the interval $X_{\mathbf{q}} \times U_{\mathbf{p}}$

$$\hat{f}_{\mathbf{q},\mathbf{p}}(\mathbf{x}_k, \mathbf{u}_k) = \sum_{\mathbf{v} \in V_{\mathbf{q},\mathbf{p}}} \gamma_{\mathbf{v}}(\mathbf{x}_k, \mathbf{u}_k) \hat{\mathbf{y}}_{\mathbf{v}}.$$

where $\gamma_{\mathbf{v}}$ represent the coefficients of the linear combination given by (5.5). Therefore, we can write the estimation problem to calculate the piecewise multi-affine estimation function $\hat{f}$ from the data set in a matrix form.

We first denote the set of all vertices of the partition

$$\mathcal{V} = \bigcup_{\mathbf{q} \in Q, \mathbf{p} \in P} V_{\mathbf{q},\mathbf{p}}.$$

The set $\mathcal{V}$ is finite and thus can be numbered $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_{|\mathcal{V}|}\}$. For every $j \in \{1, \ldots, n_x\}$, We aggregate all variables representing the values of $\hat{f}^j$ on the grid's points of the partition ($\hat{y}^j_{\mathbf{v}}$, for all $\mathbf{v} \in \mathcal{V}$) in a single vector $\Phi_j \in \mathbb{R}^K$, Then, the regression problem for every component $\hat{f}^j$ is

$$\chi_j = A \cdot \Phi_j + \mathbf{E}_j \tag{5.6}$$

where $\chi_j \in \mathbb{R}^{|\mathbb{K}|}$ is a vector aggregating the j components of the data points' transitions $\chi_j = ({x'}_1^j, \ldots, {x'}^j_{|\mathbb{K}|})$, $\mathbf{E}_j \in \mathbb{R}^{|\mathbb{K}|}$ the vector of residuals, and $A \in \mathbb{R}^{|\mathbb{K}| \times |\mathcal{V}|}$ is the sparse coefficients matrix. Each row of this matrix is built using a data point and contains $2^{n_x+n_u}$ entries which is the number of vertices of the interval $X_{\mathbf{q}} \times U_{\mathbf{p}}$ to which the data point belongs. The values of the entries at each row of the matrix are the coefficients of the multi-affine function defined on this interval as seen in (5.5).

We use the least squares estimator to find the values of Φ_j. The cost function, which is the sum of the squares of residuals, can be written as

$$\begin{aligned} S(\Phi_j) = \mathbf{E}_j^T \mathbf{E}_j &= (\chi_j - A \cdot \Phi_j)^T (\chi_j - A \cdot \Phi_j) \\ &= \chi_j^T \chi_j - 2\ \Phi_j^T A^T \chi_j + \Phi_j^T A^T A \Phi_j \end{aligned}$$

Hence, the estimation problem can be expressed as a sparse quadratic optimization problem.

Finally, let us define the two vectors $\underline{\Phi_j}, \overline{\Phi_j} \in \overline{\mathbb{R}}^{|\mathcal{V}|}$. For all the component $i \in \{1, \ldots, K\}$, we define $\underline{\Phi}^i_j, \overline{\Phi}^i_j$

$$\begin{aligned} \underline{\Phi}^i_j &= \max_{\mathbf{q}', \mathbf{p}'} \{\underline{\sigma}^j(\mathbf{q}', \mathbf{p}') \mid \mathbf{v}_i \in V_{\mathbf{q}', \mathbf{p}'}\} - \underline{d}^j \\ \overline{\Phi}^i_j &= \min_{\mathbf{q}', \mathbf{p}'} \{\overline{\sigma}^j(\mathbf{q}', \mathbf{p}') \mid \mathbf{v}_i \in V_{\mathbf{q}', \mathbf{p}'}\} - \overline{d}^j \end{aligned}$$

where $\underline{\sigma}^j(\mathbf{q}', \mathbf{p}')$ and $\overline{\sigma}^j(\mathbf{q}', \mathbf{p}')$ are the lower and upper bounds of the over-approximation of the true function on the interval $X_{\mathbf{q}'} \times U_{\mathbf{p}'}$ respectively. The two vectors $\underline{\Phi_j}, \overline{\Phi_j}$ resemble the minimum of over-approximation's upper bound and the maximum of over-approximation's lower bound for all the cells that $\mathbf{v}_i$ is a vertex of.

The following proposition gives a sufficient condition so that the estimated piecewise multi-affine function is compatible with the over-approximation built using the same partition and assumptions.

Proposition 18. *Given the finite rectangular partitions $(X_{\mathbf{q}})_{\mathbf{q}\in Q}$, $(U_{\mathbf{p}})_{\mathbf{p}\in P}$ of X and U, and under Assumption 2, the piecewise multi-affine estimation function $\hat{f}$ whose components are calculated using the following optimization problem, for every $j \in \{1, \ldots, n_x\}$,*

$$\begin{aligned} \min_{\Phi_j} \quad & \Phi_j^T A^T A \Phi_j - 2\, \Phi_j^T A^T \chi_j \\ \text{subject to} \quad & \underline{\Phi_j} \preceq \Phi_j \preceq \overline{\Phi_j} \end{aligned} \tag{5.7}$$

is compatible with the over-approximation F, i.e. $\hat{f}(\mathbf{x}, \mathbf{u}) + D \subseteq F(\mathbf{x}, \mathbf{u})$, for all $\mathbf{x} \in X$, for all $\mathbf{u} \in U$.

Proof. First, let us show that the optimization problem (5.7) has a feasible solution. According to the results in Chapter 3 we have that $\forall \mathbf{x} \in X, \mathbf{u} \in U,\ f(\mathbf{x}, \mathbf{u}) + D \subseteq F(\mathbf{x}, \mathbf{u})$. Therefore, for every $j \in \{1, \ldots, n_x\}$ the values of the f^j on the grid's points $y^j_{\mathbf{v}} = f^j(\mathbf{v})$, for all $\mathbf{v} \in V_{\mathbf{q},\mathbf{p}}$ for all $\mathbf{q} \in Q, \mathbf{p} \in P$ are a feasible solution for the optimization problem (5.7).

From (5.7), we have that for a given cell $X_{\mathbf{q}} \times U_{\mathbf{p}}$, all the resulting values of the estimation are included in the over-approximation map, i.e. $y^{\star j}_{\mathbf{v}} \leq \overline{\sigma}^j(\mathbf{q}, \mathbf{p}) - \overline{d}^j$ and $y^{\star j}_{\mathbf{v}} \geq \underline{\sigma}^j(\mathbf{q}, \mathbf{p}) - \underline{d}^j$ for all $\mathbf{v} \in V_{\mathbf{q},\mathbf{p}}$ for all $\mathbf{q} \in Q, \mathbf{p} \in P$. This implies according to Lemma 2 (by choosing $\mathbf{s} = \mathbf{e}_j$) that $\hat{f}^j(\mathbf{x}, \mathbf{u}) \leq \overline{\sigma}^j(\mathbf{q}, \mathbf{p}) - \overline{d}^j$ and $\hat{f}^j(\mathbf{x}, \mathbf{u}) \geq \underline{\sigma}^j(\mathbf{q}, \mathbf{p}) - \underline{d}^j$ for all $\mathbf{x} \in [\underline{\mathbf{x}}_{\mathbf{q}}, \overline{\mathbf{x}}_{\mathbf{q}}]$ for all $\mathbf{u} \in [\underline{\mathbf{u}}_{\mathbf{p}}, \overline{\mathbf{u}}_{\mathbf{p}}]$. Hence, $\hat{f}(\mathbf{x}, \mathbf{u}) + W \subseteq F(\mathbf{x}, \mathbf{u})$, for all $\mathbf{x} \in X$, for all $\mathbf{u} \in U$. □

Let us remark that $A^T A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $A^T \chi_j \in \mathbb{R}^{|\mathcal{V}|}$, which makes the size of the quadratic program (5.7) independent of the number of data points. These products of matrices can be computed in linear time with respect to the number of data points. Moreover, due to the sparsity of $A \in \mathbb{R}^{|\mathbb{K}|\times|\mathcal{V}|}$, these computations can be done efficiently. It follows that large data sets can be handled in practice.

Remark 7. *The estimated function $\hat{f}$ is continuous and differentiable almost everywhere, making the MPC optimization problem in* (5.3) *solvable using subgradient descent methods.*

5.3 . Updating the estimation locally

In section 3.2, we discussed how to update the estimation locally point by point. We introduce a stochastic gradient descent-like method to update the estimation function. Similar to the over-approximation case we define a window around the data point where we update the estimation function. Let

$$Q_w(\mathbf{q}) = \{\mathbf{q}^\dagger \in Q \mid max(\mathbf{q} - \underline{\mathbf{q}}_w, \mathbf{0}_n) \preceq \mathbf{q}^\dagger \preceq \min(\mathbf{q} + \overline{\mathbf{q}}_w, K^x),\ \underline{\mathbf{q}}_w, \overline{\mathbf{q}}_w \in Q\},$$

$$P_w(\mathbf{p}) = \{\mathbf{p}^\dagger \in P \mid max(\mathbf{p} - \underline{\mathbf{p}}_w, \mathbf{0}_n) \preceq \mathbf{p}^\dagger \preceq \min(\mathbf{p} + \overline{\mathbf{p}}_w, K^u),\ \underline{\mathbf{p}}_w, \overline{\mathbf{p}}_w \in P\},$$

define the window where we want to update the over-approximation. Given a data transition $(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}'_k) \in \mathcal{D}$, such that $(\mathbf{q}, \mathbf{p})\times = \phi(\mathbf{x}_k, \mathbf{u}_k)$. We will update the estimation function on the vertices of all the cells in the neighborhood that contain the data point, $\mathbf{q}' \times \mathbf{p}' \in Q_w(\mathbf{q}) \times P_w(\mathbf{p})$. We saw that for all $\mathbf{q}' \times \mathbf{p}' \in Q_w(\mathbf{q}) \times P_w(\mathbf{p})$ we have

$$\mathbf{x}'_k = \hat{f}_{\mathbf{q}',\mathbf{p}'}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{e}(\mathbf{x}_k, \mathbf{u}_k) \tag{5.8}$$

Let us denote the set of all vertices of all the cells $\mathbf{q}' \times \mathbf{p}' \in Q_w(\mathbf{q}) \times P_w(\mathbf{p})$ by

$$\mathcal{V}(k) = \bigcup_{\mathbf{q}'\times\mathbf{p}' \in Q_w(\mathbf{q})\times P_w(\mathbf{p})} V_{\mathbf{q}'}.$$

For every $j \in \{1, \ldots, n_x\}$, We aggregate all the components of the function's estimation on the vertices of the window in a single vector $\Psi_j(k) \in \mathbb{R}^K$, Then, the set of equations in (5.8) can be written in the matrix form as follows

$$\chi_j(k) = A(k) \cdot \Psi_j(k) + \mathbf{E}_j(k)$$

where $\chi_j(k) \in \mathbb{R}^{n_w}$ is a vector containing n_w replications of ${x_k}'^j$, n_w is the number of cells in the window $n_w = |Q_w(\mathbf{q}) \times P_w(\mathbf{p})|$ which we fix for all $k \in \mathbb{K}$, $A(k) \in \mathbb{R}^{|\mathcal{V}|\times n_w}$ is coefficients matrix, and $\mathbf{E}_j(k) \in \mathbb{R}^{n_w}$ the vector of residuals. Given the diagonal weights' matrix H. We want to minimize the sum of the weighted square of errors

$$\begin{aligned} S(\Psi_j(k)) = \mathbf{E}_j^T(k) \cdot H \cdot \mathbf{E}_j(k) &= (\chi_j(k) - A(k) \cdot \Psi_j(k))^T \cdot \\ & H \cdot (\chi_j(k) - A(k) \cdot \Psi_j(k)) \\ &= \chi_j^T(k) \cdot H \cdot \chi_j(k) - 2\ (A^T(k) \cdot H \cdot \chi_j(k))^T \cdot \Psi_j(k) + \\ & \Psi_j^T(k) \cdot A^T(k) \cdot H \cdot A(k) \cdot \Psi_j(k) \end{aligned}$$

We will minimize S by implementing a stochastic gradient descent-like method

$$\Psi_j(k+1) \coloneqq \Psi_j(k) - \eta(k) \nabla S(\Psi_j(k))$$

Where

$$\nabla S(\Phi_j(k)) = 2A^T(k) \cdot H \cdot A(k) \cdot \Psi_j(k) - 2\ A^T(k) \cdot H \cdot \chi_j(k)$$

To ensure that the estimation $\hat{f}$ is compatible with the over-approximation, a final step is carried out to project the estimation onto the over-approximation. For all $i \in \{1, \ldots, |\mathcal{V}|\}$

$$\Psi_j^i(k+1) \coloneqq P_{\sigma(\mathbf{q}^\dagger, \mathbf{p}^\dagger) \ominus D}\left(\Psi_j(k)\right), \text{ for all } \mathbf{q}^\dagger \times \mathbf{p}^\dagger \in Q_{\mathbf{v}_i} \tag{5.9}$$

Where $Q_{\mathbf{v}_i}$ is the set of cells that contain $\mathbf{v}_i$, and $\sigma(\mathbf{q}^\dagger, \mathbf{p}^\dagger)$ is the over-approximation of the true function on the interval $X_{\mathbf{q}^\dagger} \times U_{\mathbf{p}^\dagger}$. The operator $\ominus$ is the Minkowski difference, and D is the set of disturbances.

Proposition 19. *Given the partitions $(X_{\mathbf{q}})_{\mathbf{q} \in Q}$, $(U_{\mathbf{p}})_{\mathbf{p} \in P}$ of X and U, the piecewise multi-affine estimation function $\hat{f}$ updated locally as in* (5.9) *is compatible with the over-approximation σ*

Proof. Lemma 2 states that the value of multi-affine functions inside an interval is a convex combination of their values on the vertices of that interval. Therefore, according to (5.9) we have that for all $\mathbf{q} \in Q$, $\mathbf{p} \in P$, for all $(\mathbf{x}, \mathbf{u}) \in X_{\mathbf{q}} \times U_{\mathbf{p}}$, $\hat{f}_{\mathbf{q},\mathbf{p}}(\mathbf{x}, \mathbf{u}) \in \sigma(\mathbf{q}, \mathbf{p}) \ominus D$ or $\hat{f}_{\mathbf{q},\mathbf{p}}(\mathbf{x}, \mathbf{u}) + D \subseteq \sigma(\mathbf{q}, \mathbf{p})$ □

5.4 . Online update of model and safety controller

In this section, we will discuss how to update the safe set and the safety controller using data points online. let us consider that we sampled a set of data $\mathcal{D}$ from the system's dynamics. Then, we calculated, offline, the over-approximation of the system's dynamics and a compatible estimation function as discussed previously. Using the over-approximation, we built a finite-sate model and implemented a discrete controller synthesis algorithm to find a safety controller $\mathcal{C}_{safe}$ and a maximal control invariant set X_{inv}. According to Theorem 4, we can implement an MPC scheme to achieve some control objectives while ensuring safety. Now, while the system is running, we can collect data points and use them to update the over-approximation and the estimation function. The update process should be done in a way that ensures that

- The updated over-approximation is still a valid over-approximation of the true function.
- The updated over-approximation should be used to update the maximal control invariant set and the safety controller.
- The updated estimation is still compatible with the over-approximation.
- The process of updating the over-approximation, the estimation function and the safety controller should be done online. Meaning that the entire update process should be done between two consecutive sampling times, while also leaving enough time to solve the MPC optimization problem.

In previous sections, we discussed how to update the over-approximation and the estimation function locally using the collected data points. Now we will discuss how to use the updated over-approximation and estimation function to update the maximal control invariant set and the safety controller. First, one can ask if and when can we update the control invariant set and the safety controller. The answer to this question lies in the idea that knowing the bounds of the derivatives of the function allows us to calculate growth bounds for the function, and learn more about the system's dynamics in the neighborhood of the data points. Therefore, even if we are operating the system in the safe region found after the offline phase, we can still learn more about the system's dynamics and update the control invariant set and the safety controller by sampling data points on the boundary of the safe set. Therefore, when sampling data points on the boundary of the safe set, we can update the over-approximation in the neighborhood of the data point, which will result in less conservative over-approximation and the possibility of finding a larger maximal control invariant set. Algorithm 4 shows how to update the control

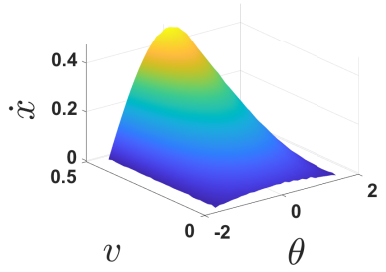
Algorithm 4 Update the control invariant set and the safety controller

Input: σ_{new}, $\hat{f}$, X_{inv}, $\mathcal{C}_{safe}$
Output: X_{inv}, $\mathcal{C}_{safe}$
for $\mathbf{q} \in Q_w$ **do**
    for $\mathbf{p} \in P_w$ **do**
        if $\sigma_{new}(\mathbf{q}, \mathbf{p}) \subseteq X_{inv}$ **then**
            $X_{inv} \leftarrow X_{inv} \cup \mathbf{q}$
            $\mathcal{C}_{safe}(\mathbf{q}) \leftarrow \mathcal{C}_{safe}(\mathbf{q}) \cup \mathbf{p}$
        end if
    end for
end for

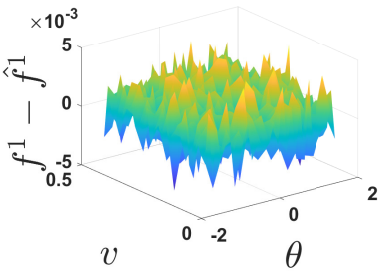
invariant set and the safety controller using the updated over-approximation σ_{new}. The resulting control invariant set is not necessarily maximal, but it is a superset of the previous control invariant set, i.e. $X_{inv} \supseteq X_{inv}^{old}$. It is not maximal because we only updated the control invariant set in the neighborhood of the data points, and to find the maximal control invariant set, we need to go through all the cells in the partition and check if any other cells became safe after the update. The reason why we only update the control invariant set in the neighborhood of the data points is that we want to update the control invariant set online, and going through all the cells in the partition is computationally expensive.

5.5 . Numerical examples

In this section, we will show some numerical examples to demonstrate the effectiveness of the proposed method. We first continue with the example introduced



(a) The single-valued estimation of the dynamics of $\dot{x}$ ($\hat{\dot{x}}$)



(b) The difference between the true function and the estimation on the grid's points

Figure 5.1: Estimation with piecewise multi-affine functions

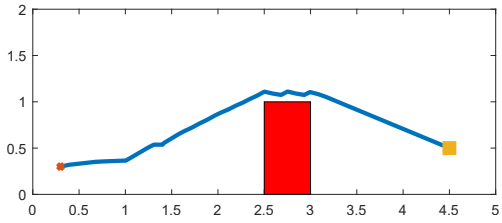


Figure 5.2: Trajectory of the vehicle

in Chapter 3 where we found the maximal control invariant set and the safety controller for a bicycle model. Now, we will show how to use the proposed method to find a feasible trajectory for a bicycle model to reach a target position while avoiding an obstacle. In the second example, we consider also a path-planning problem for a bicycle model. However, the vehicle is capable of rotating in all directions, and we want to update the safety controller online.

5.5.1 . Path planning offline - Continued

In 4.5.2 we build the over-approximation of the system's dynamics of a bicycle model and found the maximum safe control input for each state. Now we will use this information to build a path planner for the bicycle model. We will find an optimal path from a given initial state to a given goal state by applying the safe MPC scheme introduced in this chapter.

that, we calculated the piecewise multi-affine estimation of the dynamics according to the algorithm introduced in Section 5.2. Figure 5.1 shows the result of estimation for the first component of the dynamical model and the difference between the true function and the estimation on the grid's points. The execution time to find the piecewise multi-affine estimation is $t_{est} = 31.48$ s.

Finally, the calculated estimation and safety controller are used to find a feasible trajectory to reach the target position while avoiding the obstacle. We chose a

starting position $\mathbf{x}_0 = (0.3, 0.3)$ and the target $\mathbf{x}_f = (4.5, 0.5)$. The cost function

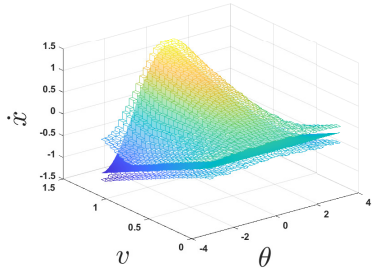
$$J = \sum_{i=1}^{N} ||\mathbf{x}(i|t) - \mathbf{x}_f|| \tag{5.10}$$

was chosen to drive the vehicle to the target. The trajectory is shown in Figure 5.2, where it can be seen that the vehicle reached the target position while avoiding the obstacle.
Although we chose a simple sum-of-distances cost function (5.10), we were able to reach the goal while avoiding the obstacle.

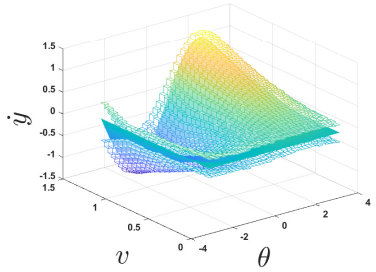
5.5.2 . Path planning online

In this example, we present a path-planning problem where a mobile robot (a vehicle) needs to navigate a given environment and reach four distinct regions while avoiding obstacles in its way. We sampled a set of data from the vehicle dynamics at low speed. We found the set of safe controllers and the estimation of dynamics. We then ran the system at those settings and collected new data points online. We chose a cost function that favors moving as fast as possible in order to learn the dynamics of the robot at high speed. We do all of that while always ensuring safety. We consider the same unicycle model defined in the previous example.

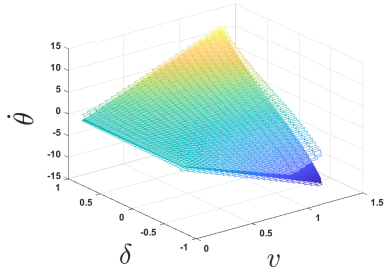
We chose for the states, inputs, and disturbance intervals the following $\theta \in [-\pi, \pi], \delta \in [-\frac{\pi}{4}, \frac{\pi}{4}], v \in (0, 2], \mathbf{w} \in [-0.05, 0.05] \times [-0.05, 0.05] \times [-0.05, 0.05]$, and partitioned those intervals uniformly into 40 cells each.
To find the over-approximation, we sampled $|\mathbb{K}| = 10^4$ data transition. Only low speeds from the interval $v \in (0, 1]$ were chosen. We chose a window of size 11×11. The average time to update the over-approximation of each component was $t = 1.7$ ms. The learned over-approximation of the dynamics can be seen in Figure 5.3. Although we sampled data points with speeds $v \leq 1$, the proposed technique allows for finding over-approximation for speeds $v > 1$, albeit more conservative, as it can be seen in Figure 5.3. The environment where the robot operates, which can be seen in Figure 5.6, is a 5×5 m area. It was partitioned into 50×50 cells. The calculated over-approximation was used to find the maximal control invariant, which can be seen in Figure 5.4, and the set of safe controllers at each cell. We use the Euler method for discretizing with a discretization time $dT = 0.2$ s. The time it took to find the finite-state model and calculate the safety controller was $t = 424$ s. Also, compatible estimations were calculated using the data collected offline, as can be seen from Figure 5.5a. The average time to update the estimation of each component was $t = 80$ ms.
Everything till now was done offline. We then ran the experiment, which consisted of making the robot visit the four colored areas in the order (red, yellow, green, blue) as fast as possible. This was done using the cost function $J = \sum_{i=1}^{N} ||\mathbf{x}(i|t) - \mathbf{x}_f|| + \frac{1}{v(i|t)}$, where $\mathbf{x}_f$ is the center of the colored areas, and it is changed to next destination the moment the robot touches a given area. We collected the data



(a) The over-approximation of the dynamics of $\dot{x}$



(b) The over-approximation of the dynamics of $\dot{y}$



(c) The over-approximation of the dynamics of $\dot{\theta}$

Figure 5.3: Over-approximation of the bicycle dynamics. The true unknown functions are shown in solid

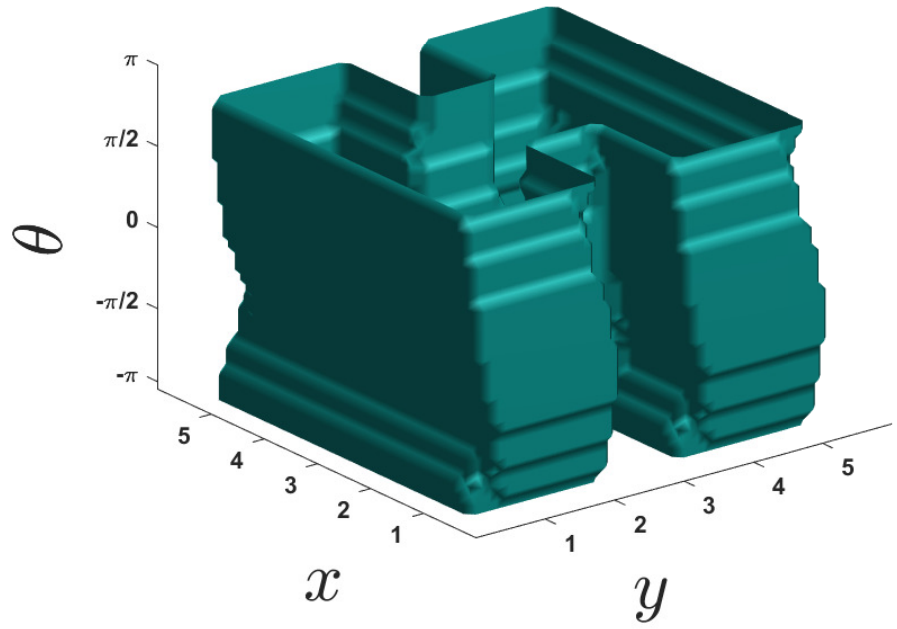
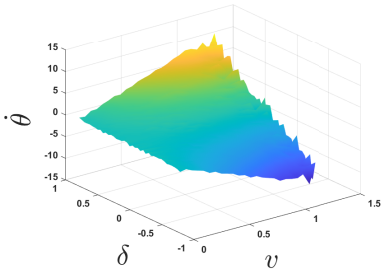


Figure 5.4: Caption

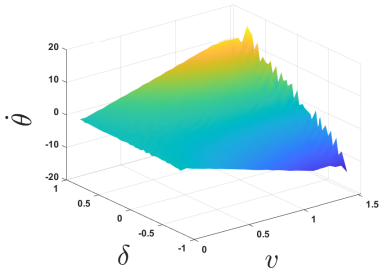
online and updated the models, as can be seen in Figure 5.5. We were able to learn the dynamics of the system for higher speed that we did not sample for the offline stage. Most importantly, we did all of that while always ensuring safety.

5.6 . Conclusion

In this chapter, we proposed an approach to learning-based MPC for safety-critical systems. The approach is based on finding two models of the system's dynamics, an over-approximation, and a compatible estimation function. The over-approximation is used to find a maximal control invariant set and a safety controller. The estimation function is used to solve the MPC optimization problem and enforce performance objectives. The proposed estimation is piecewise multi-affine. We chose this type of estimation because it is compatible with the over-approximation. Moreover, it is continuous and differentiable almost everywhere, which makes the MPC optimization problem solvable using subgradient descent methods. We also presented a method to update the estimation function locally using the collected data points. Finally, we showed how to update the control invariant set and the safety controller using the updated over-approximation and estimation function. We demonstrated the effectiveness of the proposed method using two numerical examples.



(a) The single-valued estimation of the dynamics of $\dot{\theta}$ ($\hat{\dot{\theta}}$) from the data collected offline



(b) The single-valued estimation of the dynamics of $\dot{\theta}$ ($\hat{\dot{\theta}}$) After it was updated using data points collected online

Figure 5.5: Estimation with piecewise multi-affine functions

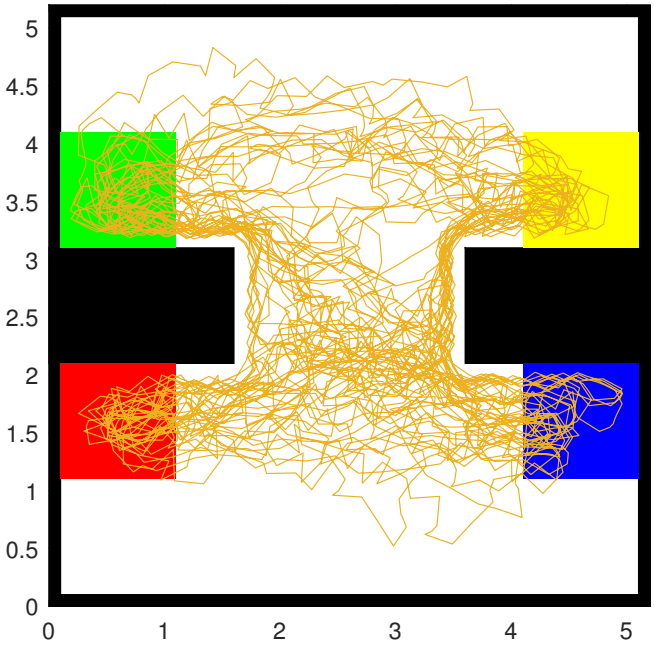


Figure 5.6: The trajectory of the mobile robot visiting all four areas ten times

6 - Conclusion and Future Work

Conclusions

In this thesis, we have addressed the problem of learning models of nonlinear dynamical systems from data that are suitable for controller synthesis and safe to implement. We have proposed novel data-driven methods for over-approximating the behavior of monotone systems and systems with bounded derivative functions. We have also shown how to use these over-approximations to construct finite-state abstractions of the systems, which can be used to synthesize controllers that satisfy safety and reachability specifications. Furthermore, we have introduced a two-model approach to safe learning, where we use a discrete model to synthesize a safety controller and a compatible estimation of the dynamics to achieve the desired performance.

A method to learn data-driven over-approximations of monotone maps from input-output data was introduced in Chapter 2. We have shown that the resulting map is the tightest map that is consistent with the data. The resulting map is piecewise interval-valued and monotone. We also introduced a piecewise interval-valued map on a predefined partition of the input space. This map is the tightest map that is consistent with the data on the predefined partition. This map offers a trade-off between tightness from one side and easiness of calculation and storage size from the other side. We have also shown how to calculate this map efficiently.

In Chapter 3 we have extended the method of Chapter 2 to learn data-driven over-approximations of maps described with bounded derivative functions plus an interval of disturbances from input-output data. We have presented several techniques to compute such an over-approximation, such as transforming the data into a monotone case or finding growth cones that over-approximate the unknown map. The calculated over-approximations are also piecewise interval-valued. We demonstrated how to obtain those over-approximations efficiently. We have also discussed how to update the over-approximation locally when new data points are added. An algorithm to calculate the bounds of the derivative of the unknown function was also discussed.

Using the over-approximations of the unknown maps, we have shown in Chapter 4 how to construct finite-state abstractions of the systems, which can be then used to synthesize controllers that can satisfy complex specifications. In this chapter, we introduced the needed background on systems relations, abstraction, symbolic models, and discrete controller synthesis. Theorem 3 was introduced to show how to construct a data-driven symbolic model out of the interval-valued over-approximation of the unknown map. Moreover, we showed that in the case where the studied system is completely unknown, working with the data-driven symbolic model is equivalent to working with the data-driven over-approximation of the

unknown map.

In Chapter 5, we introduced a two-model approach to safe learning. The first model is a discrete model that is used to synthesize a safety controller. Thus, ensuring that the learned system is safe. The second model is a compatible estimation of the dynamics, which is used to achieve desired performance. The method relies on implementing a learning-based MPC scheme where the safety controller synthesized from the discrete model is used as a constraint on the optimization problem, and the compatible estimation of the dynamics is used to predict the future behavior of the system, and subsequently to calculate and minimize the cost function. We show how to learn a compatible estimation of the dynamics of the system from data, which is a piecewise multi-affine estimation of the dynamics of the system. We have also presented a method to update the estimation locally when new data points are added. Finally, we have shown how to use the methods of updating the over-approximation and the compatible estimation locally to update the models of the system online when the system is running. The updated models can increase the set of states where we can guarantee the safety of the system, broaden the set of safe actions, and improve the performance of the Learning-based MPC controller.

Future Work

Our research opens up several directions for future work, such as:

- We have shown how to learn data-driven over-approximations of monotone systems and systems with bounded derivative functions on a predefined partition of the space. More work is needed to study how to choose the partition of the space in order to obtain the best over-approximation.
- Exploring other types of specifications and objectives for controller synthesis, such as temporal logic, robustness, or optimality. Also, we should try to apply our methods to real-world problems and scenarios, such as autonomous driving, robotics, or smart grids.
- Study more thoroughly the effects of hyperparameters on the performance of the algorithms. Like the size of the window when we update the models locally, the learning rate of the gradient descent algorithm, etc.
- Investigate how to explore the state space of the system in order to obtain the best data points to learn the models of the system. How can we visit all the state space and sample the space densely? We know that all introduced methods improve with the increase in the number of data points. Finally, we should find a way to balance between exploring the state space and exploiting the data that we already have (the trade-off between exploration and exploitation).

- Adapt the learned model to the given specifications and objectives. For example, the case of monotone systems with lower(upper)-closed specifications, or any other scenario where we may only need to learn a sparse model of the system.
- Investigate the possibility of learning data-driven over-approximations and compatible estimations of systems with more general dynamics, such as hybrid systems, or systems with more general nonlinear dynamics.

Finally, we hope that our work will inspire further research on data-driven control and safe learning, and will provide useful tools and techniques for designing reliable and efficient controllers for complex systems.

Bibliography

[1] A. Alanwar, A. Koch, F. Allgöwer, and K. H. Johansson. Data-driven reachability analysis from noisy data. *IEEE Transactions on Automatic Control*, 2023.

[2] M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

[3] M. Althoff, G. Frehse, and A. Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:369–395, 2021.

[4] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[6] D. Angeli and E. D. Sontag. Monotone control systems. *IEEE Transactions on automatic control*, 48(10):1684–1698, 2003.

[7] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[8] C. Belta and L. C. G. J. M. Habets. Controlling a class of nonlinear systems on rectangles. *IEEE Transactions on Automatic Control*, 51(11):1749–1759, Nov. 2006.

[9] C. Belta, B. Yordanov, and E. A. Gol. *Formal methods for discrete-time dynamical systems*. Springer, 2017.

[10] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer. Data-driven model predictive control with stability and robustness guarantees. *IEEE Transactions on Automatic Control*, 66(4):1702–1717, 2020.

[11] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[13] A. Borri, G. Pola, P. Pepe, M. D. Di Benedetto, and P. Palumbo. Symbolic control design of an artificial pancreas for type-2 diabetes. *IEEE Transactions on Control Systems Technology*, 30(5):2131–2146, 2021.

[14] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

[15] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[16] I. Buzhinsky, A. Pakonen, and V. Vyatkin. Explicit-state and symbolic model checking of nuclear i&c systems: A comparison. In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 5439–5446. IEEE, 2017.

[17] J.-P. Calliess. *Conservative decision-making and inference in uncertain dynamical systems*. PhD thesis, University of Oxford Oxford, 2014.

[18] M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

[19] M. Canale, L. Fagiano, and M. Signorile. Nonlinear model predictive control from data: a set membership approach. *International Journal of Robust and Nonlinear Control*, 24(1):123–139, 2014.

[20] X. Chen, E. Abraham, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *2012 IEEE 33rd Real-Time Systems Symposium*, pages 183–192. IEEE, 2012.

[21] A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification. *IEEE transactions on automatic control*, 48(1):64–75, 2003.

[22] R. Coppola, A. Peruffo, and M. Mazo Jr. Data-driven abstractions for verification of deterministic systems. *arXiv preprint arXiv:2211.01793*, 2022.

[23] J. Coulson, J. Lygeros, and F. Dörfler. Data-enabled predictive control: In the shallows of the deepc. In *2019 18th European Control Conference (ECC)*, pages 307–312, 2019.

[24] C. Dawson, S. Gao, and C. Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods. *arXiv preprint arXiv:2202.11762*, 2022.

[25] A. Devonport, A. Saoud, and M. Arcak. Symbolic abstractions from data: a PAC learning approach. In *IEEE Conference on Decision and Control*, pages 599–604, 2021.

[26] A. Devonport, F. Yang, L. El Ghaoui, and M. Arcak. Data-driven reachability and support estimation with christoffel functions. *IEEE Transactions on Automatic Control*, 2023.

[27] J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[28] A. Girard. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*, pages 291–305. Springer, 2005.

[29] A. Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012.

[30] A. Girard and A. Eqtami. Least-violating symbolic controller synthesis for safety, reachability and attractivity specifications. *Automatica*, 127, 2021.

[31] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions. *IFAC Proceedings Volumes*, 41(2):8966–8971, 2008.

[32] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, 2009.

[33] K. Hashimoto, A. Saoud, M. Kishida, T. Ushio, and D. Dimarogonas. Learning-based symbolic abstractions for nonlinear control systems. *arXiv preprint arXiv:2004.01879*, 2020.

[34] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[35] Z.-S. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[36] E. Ivanova, A. Saoud, and A. Girard. Lazy controller synthesis for monotone transition systems and directed safety specifications. *Automatica*, 135:109993, 2022.

[37] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied interval analysis, 2001*. Springer, 2001.

[38] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[39] A. Kandel and S. J. Moura. Safe learning mpc with limited model knowledge and data. *arXiv preprint arXiv:2004.00759*, 2020.

[40] Y. Kawano, B. Besselink, J. M. A. Scherpen, and M. Cao. Data-driven model reduction of monotone systems by nonlinear DC gains. *IEEE Transactions on Automatic Control*, 65(5):2094–2106, 2020.

[41] M. Kazemi, R. Majumdar, M. Salamati, S. Soudjani, and B. Wooding. Data-driven abstraction-based control synthesis. *arXiv preprint arXiv:2206.08069*, 2022.

[42] M. Khajenejad, Z. Jin, and S. Z. Yong. Interval observers for simultaneous state and model estimation of partially known nonlinear systems. In *American Control Conference*, pages 2848–2854, 2021.

[43] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control (CDC)*, pages 6059–6066. IEEE, 2018.

[44] V. Krishnan and F. Pasqualetti. On direct vs indirect data-driven predictive control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 736–741. IEEE, 2021.

[45] A. B. Kurzhanski. Dynamics and control of trajectory tubes. theory and computation. In *2014 20th International Workshop on Beam Dynamics and Optimization (BDO)*, pages 1–1. IEEE, 2014.

[46] A. Lavaei and E. Frazzoli. Data-driven synthesis of symbolic abstractions with guaranteed confidence. *IEEE Control Systems Letters*, 7:253–258, 2022.

[47] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[48] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[49] Q. Luo and H. Duan. Symbolic control approach to aircraft taking off in wind shear. *Aircraft Engineering and Aerospace Technology: An International Journal*, 87(1):45–51, 2015.

[50] W. Luo, W. Sun, and A. Kapoor. Sample-efficient safe learning for online nonlinear control with control barrier functions. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 419–435. Springer, 2022.

[51] A. Makdesi, A. Girard, and L. Fribourg. Data-driven abstraction of monotone systems. In *Learning for Dynamics and Control*, pages 803–814. PMLR, 2021.

[52] A. Makdesi, A. Girard, and L. Fribourg. Efficient data-driven abstraction of monotone systems with disturbances. *IFAC-PapersOnLine*, 54(5):49–54, 2021.

[53] A. Makdesi, A. Girard, and L. Fribourg. Data-Driven Models of Monotone Systems. 2022.

[54] P.-J. Meyer, A. Devonport, and M. Arcak. *Interval reachability analysis: Bounding trajectories of uncertain systems with boxes for control and verification*. Springer Nature, 2021.

[55] P.-J. Meyer, A. Girard, and E. Witrant. Controllability and invariance of monotone systems for robust ventilation automation in buildings. In *52nd IEEE Conference on Decision and Control*, pages 1289–1294. IEEE, 2013.

[56] P.-J. Meyer, A. Girard, and E. Witrant. Safety control with performance guarantees of cooperative systems using compositional abstractions. *IFAC-PapersOnLine*, 48(27):317–322, 2015.

[57] M. Milani Fard, K. Canini, A. Cotter, J. Pfeifer, and M. Gupta. Fast and flexible monotonic functions with ensembles of lattices. *Advances in neural information processing systems*, 29, 2016.

[58] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

[59] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *IEEE Transactions on Control Systems Technology*, 24(4):1294–1307, 2015.

[60] C. Novara and M. Milanese. Set membership identification of nonlinear systems. In *IEEE Conference on Decision and Control*, volume 3, pages 2831–2836, 2000.

[61] Y. Ou and M. Tavakoli. Towards safe and efficient reinforcement learning for surgical robots using real-time human supervision and demonstration. In *2023 International Symposium on Medical Robotics (ISMR)*, pages 1–7. IEEE, 2023.

[62] G. Pola and M. D. Di Benedetto. Control of cyber-physical-systems with logic specifications: a formal methods approach. *Annual Reviews in Control*, 47:178–192, 2019.

[63] G. Pola, T. Masciulli, E. De Santis, and M. D. Di Benedetto. Data-driven controller synthesis for abstract systems with regular language specifications. *Automatica*, 134:109903, 2021.

[64] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2016.

[65] M. Rungger, M. Mazo Jr, and P. Tabuada. Specification-guided controller synthesis for linear systems and safe linear-time temporal logic. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 333–342, 2013.

[66] S. Sadraddini and C. Belta. Formal guarantees in data-driven model identification and control synthesis. In *International Conference on Hybrid Systems: Computation and Control*, pages 147–156, 2018.

[67] A. Saoud, A. Girard, and L. Fribourg. Contract-based design of symbolic controllers for safety in distributed multiperiodic sampled-data systems. *IEEE Transactions on Automatic Control*, 66(3):1055–1070, 2020.

[68] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[69] V. Sinyakov and A. Girard. Formal controller synthesis from specifications given by discrete-time hybrid automata. *Automatica*, 131, 2021.

[70] H. L. Smith. *Monotone dynamical systems: an introduction to the theory of competitive and cooperative systems: an introduction to the theory of competitive and cooperative systems*. Number 41. American Mathematical Soc., 1995.

[71] P. Tabuada. *Verification and control of hybrid systems: A symbolic approach*. Springer Science & Business Media, 2009.

[72] P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, 2006.

[73] A. Taylor, A. Singletary, Y. Yue, and A. Ames. Learning for safety-critical control with control barrier functions. In *Learning for Dynamics and Control*, pages 708–717. PMLR, 2020.

[74] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning for dynamical systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 5997–6004. IEEE, 2009.

[75] L. Yang, O. Mickelin, and N. Ozay. On sufficient conditions for mixed monotonicity. *IEEE Transactions on Automatic Control*, 64(12):5080–5085, 2019.

[76] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta. Temporal logic control of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control*, 57(6):1491–1504, 2011.

[77] M. Zamani, G. Pola, M. Mazo, and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2011.

[78] X. Zhang, J. Liu, X. Xu, S. Yu, and H. Chen. Robust learning-based predictive control for discrete-time nonlinear systems with unknown dynamics and state constraints. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

[79] D. Zonetti, A. Saoud, A. Girard, and L. Fribourg. Decentralized monotonicity-based voltage control of dc microgrids with zip loads. *IFAC-PapersOnLine*, 52(20):139–144, 2019.

Synthèse

Dans le domaine du contrôle des systèmes, les méthodes traditionnelles de contrôle des systèmes non linéaires reposent principalement sur l'utilisation de modèles mathématiques précis. Ces modèles servent de colonne vertébrale au processus de contrôle, fournissant le cadre nécessaire pour comprendre et prédire le comportement du système. Cependant, l'obtention de ces modèles peut être un défi de taille, en particulier dans les applications réelles où la complexité et l'imprévisibilité du système peuvent rendre extrêmement difficile, voire impossible, la dérivation d'un modèle précis.

Pour contourner ce problème, l'avènement des techniques de contrôle basées sur les données est apparu comme une alternative prometteuse aux méthodologies de contrôle traditionnelles. Ces techniques innovantes exploitent la puissance des données d'entrée-sortie pour apprendre directement les politiques de contrôle à partir du comportement du système. Cette approche contourne le besoin d'un modèle mathématique précis, en tirant plutôt des informations directement des réponses du système à diverses entrées.

Cependant, bien que les techniques de contrôle basées sur les données offrent des avantages significatifs, elles introduisent également de nouveaux défis. L'une des préoccupations les plus critiques dans le contrôle basé sur les données est la sûreté. Des modèles erronés, dérivés de données défectueuses ou incomplètes, peuvent entraîner des conséquences catastrophiques, ce qui rend impératif de garantir l'exactitude et la fiabilité des modèles appris.

En réponse à ces défis, ce travail propose une nouvelle approche basée sur les données pour contrôler les systèmes non linéaires. Cette approche met un accent particulier sur la sûreté pendant le processus d'apprentissage, garantissant que les politiques de contrôle dérivées ne compromettent pas la stabilité ou l'intégrité du système.

L'approche proposée emploie des sur-approximations de la dynamique du système pour fournir des représentations conservatrices mais sûres du comportement du système. Ces sur-approximations sont apprises à partir des données d'entrée-sortie, capturant les réponses du système à diverses entrées sous une forme qui peut être facilement analysée et utilisée. Ces représentations apprises sont ensuite utilisées pour construire des abstractions à états finis, qui distillent la dynamique essentielle du système sous une forme compacte et analysable.

Cette abstraction est ensuite utilisée pour la synthèse du contrôleur. Le contrôleur synthétisé est conçu pour maintenir les propriétés et spécifications souhaitées tout au long du fonctionnement du système, garantissant que le système se comporte comme prévu dans un large éventail de conditions.

Un aspect unique de l'approche proposée est l'introduction d'une stratégie à deux modèles. Dans cette stratégie, des modèles distincts sont construits pour la

vérification de la sûreté et l'optimisation des performances. Le modèle de vérification de la sûreté est utilisé pour garantir que le contrôleur appris respecte les contraintes de sûreté, tandis que le modèle d'optimisation des performances se concentre sur l'atteinte des mesures de performance souhaitées. Cette séparation des préoccupations garantit que la sûreté est priorisée sans compromettre les performances.

L'approche proposée est rigoureusement analysée pour une large classe de systèmes non linéaires, y compris les systèmes monotones et les systèmes avec des fonctions dérivées bornées. Ces analyses fournissent des garanties théoriques pour la sûreté des contrôleurs appris et établissent la solidité de la méthodologie de conception proposée.

L'efficacité des méthodes proposées est démontrée par une validation expérimentale approfondie sur divers systèmes non linéaires réels, notamment le contrôle de vitesse, la planification de trajectoires et les systèmes chaotiques (système de Lorenz). Ces expériences démontrent systématiquement la capacité de l'approche proposée à atteindre des performances satisfaisantes tout en maintenant une stricte adhésion aux contraintes de sûreté.