Data-driven Robust Verification and Control

Luke Rickard

Worcester College

University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Michaelmas 2025

To my darling Emily
and all that the future shall hold

Acknowledgements

Before anything gets too technical, I would like to take a moment to thank the many people who have assisted me in completing this thesis. It has not always been the easiest journey, but it has been an enjoyable start to all that is to come, and I am very grateful to have reached this stage. I am certain that in writing these thanks there will be many people I have missed, but I am grateful for all who have been there during my journey.

First and foremost, I would not be here without the help and support of my supervisors Prof. Kostas Margellos and Prof. Alessandro Abate. Prof. Abate, thank you for introducing me to a new area of research and for the support in deciding where to take my research. Prof. Margellos, thank you for all the support and time you dedicated to helping me produce high quality research, and thank you especially for the guidance in my future career. I am sincerely grateful to have been able to work with both of you, and hope to collaborate again in the future!

I would like to thank Prof. Dave Parker and Prof. Antonis Papachristodoulou for serving as assessors for my transfer and confirmation of status. Your feedback at both of these important milestones has helped to shape the direction of this thesis, and my skills as a young academic.

Thank you also to Prof. Luca Furieri and Prof. Abolfazl Lavaei for acting as examiners for my DPhil viva, and for the feedback that has improved the final version of this document.

I am grateful to all the friends who have supported me along this journey. Many thanks to Boris Andrews, Chris Wright and Joseph Dendle who have been there since before I began this doctorate, and thanks to Joe and Chris for joining me in our hike along the ridgeway. To Patrick Benjamin, Kelsey Doerksen, Ben Gutteridge, Seb Towers, and Matthew Jackson for the DnD sessions, and to all the AIMS students for the support and good times together, both in the first year of this journey and throughout. Many of these get togethers would not have been possible without the support of Wendy Poole, who has also been there whenever I had travel to book or needed equipment, thank you for all your efforts in taking care of these.

Outside of research, I have enjoyed the opportunity to volunteer at my local Scout group. Thank you to Avril Williams for the many hours of hard work needed to allow the group to run, and for the warm welcome to the Scout group; to Sam Hutchins-Fry for joining the group as I prepare to leave; and to all the Scouts who have made every week interesting and fun. Thanks also to the efforts of all others who allowed the group to run.

Thank you to my parents, Heidi and Jonathan, for the support you have both given me throughout my life. Without you I could not be the person I am today. Many thanks also to my brother James, and sister Abi.

Finally, I would like to thank my partner Emily Long for being by my side throughout this long journey. We may have faced many difficulties throughout these past few years, but I know that together we will make it through. Your support has been invaluable and I know you will always be there to cheer me on. I dedicate this thesis to you.

Abstract

In recent years, the availability of data has exploded, leading to the so-called data era. This has led to a shift in the way we think about controlling systems, from time consuming model-based approaches, to data-driven techniques such as reinforcement learning, which can learn to control a system based only on data from that system. However, such techniques come at a cost, in particular, a lack of guarantees on system behaviour. In this thesis we explore how to make use of data to guarantee systems behave in a specified way, as well as how we can exploit data to control systems, whilst maintaining guarantees on future behaviour.

In order to verify and control our systems we require data from that system, obtaining this data may be, in general, difficult or expensive. As such, we seek to utilise all available data in order to optimise for a controller and provide guarantees, that is, we do not wish to withhold any data for testing. We therefore turn to the so-called scenario approach, which provides a framework for obtaining probably approximately correct bounds on the generalisability of a solution to a scenario program. In particular, given an optimisation program with data affecting the outcome, the scenario approach provides a guarantee on the probability of a newly sampled data point changing the result.

This thesis considers applying the scenario approach theory to increasingly complex control and verification problems. We begin by considering the synthesis of a controller for a class of finite-state models with uncertain probabilistic transitions between states. Then, we investigate a certificate synthesis problem for general continuous-state systems, introducing a novel algorithm with favourable properties for non-convex scenario programs. Finally, we turn to controller synthesis for continuous-state systems with uncertain state transitions.

Contents

List of Figures

List of Tables

Notation

$\mathbb{R}$	Denotes the set of real numbers
$\mathbb{R}_{\geq 0}$	Denotes the non-negative reals $\{n \in \mathbb{R} \colon n \geq 0\}$
$\mathbb{R}^n$	Denotes the cartesian product $\prod_{i=1}^{n} \mathbb{R}$
$\mathbb{N}$	Denotes the natural numbers $\{0, 1, 2, \dots\}$
$\{\xi_k\}_{k=0}^{K}$	Denotes a sequence indexed by $k \in \{0, 1, \dots, K\}$
$A \subseteq B$	Denotes that every element of A is also an element of B
$A \subset B$	Denotes that every element of A is also an element of B, but that the converse is not true
$\lvert A \rvert$	Denotes the number of elements of A
A^B	Denotes the set of all functions from B to A
$A \cap B$	Denotes the set of elements contained in both A and B
$A \cup B$	Denotes the set of elements contained in either A or B
$A \setminus B$	Denotes the set of elements contained in A that are not also in B
$a \wedge b$	Denotes logical conjunction (i.e. it evaluates to true only if a and b both evaluate to true
$a \vee b$	Denotes logical disjunction (i.e. it evaluates to true if either a or b both evaluate to true
$V \models \psi$	Denotes satisfaction, that is, it evaluates to true if the quantity V on the left satisfies the condition ψ on the right, e.g., $x = 1 \models x > 0$ evaluates to true and $x = -1 \models x > 0$ evaluates to false
$V \not\models \psi$	Denotes the logical inverse of satisfaction (i.e., condition dissatisfaction)
$a \models \psi, \forall a \in A$	Evaluates to true only if ψ is satisfied for all elements of A
$\exists a \in A \colon a \models \psi$	Evaluates to true if ψ is satisfied for any element of A
$(\forall \xi \in \Xi) V \models \psi(\xi)$	Denotes that some quantity V satisfies a condition ψ which, in turn, depends on some parameter ξ, for all $\xi \in \Xi$

$(\Delta, \mathcal{F}, \mathbb{P})$	Denotes a probability measure space (precisely defined per chapter)
$\theta^\star$	Denotes the optimiser of some optimisation program

1 Introduction

1.1 Motivation

In a world where autonomous systems are increasingly being deployed in safety-critical settings, it is an important challenge to prove that such systems satisfy certain properties, for example, stability, safety or reachability [46, 65, 85, 96]. In many systems it is necessary to verify not just one of these properties but some combination or product of properties. Verifying the satisfaction of these properties, also termed as specifications, is a challenging, but important, problem. Further, it is often necessary to develop a controller which guides the system to satisfy these specifications.

One technique for verification involves translating the specification to a formal logic (e.g. probabilistic computation tree logic (PCTL, [58]) or linear temporal logic (LTL, [92])). Then, there are many techniques which allow one to verify if a dynamical system meets those logic specifications, for example for LTL specifications in [17, 123], and for PCTL specifications in [15, 39].

Such techniques typically consider systems with known dynamics [16]. Provided this model accurately reflects the real-world behaviour of a dynamical system, one can provide guarantees on the underlying dynamics satisfying the specification. However, obtaining a model of the system is in general difficult, since formulating a model requires domain-specific knowledge, and the inherent uncertainty present in the system may be hard to quantify. Typically, this may be achieved by modelling

the system under study based on first principles, and accounting for unknown parts of the system or externalities as uncertainty (often assumed to be a Gaussian distribution [90]) [57]. This is a common approach in robust control, with a number of techniques being developed to explicitly deal with the presence of uncertainty [33], imposing certain assumptions on the geometry of the uncertainty set or on the underlying probability distribution of the uncertainty. Despite some recent techniques which do not assume uncertainty is normally distributed [21], this assumption is often still employed to ensure that the problem is tractable.

However, such assumptions on the uncertainty distribution risk being overly conservative, or else not sufficiently broad to cover the true uncertainty underlying the system [90]. Instead, we may wish to investigate modelling the uncertainty based on data/samples from its distribution, thus avoiding restrictive assumptions on the shape of this distribution, and allowing for tractability even for complex distributions. At the same time, we exploit the so-called data era, and the large availability of often easily accessible data [1]. These techniques are known as data-driven or sampling-based techniques [12, 39, 104].

There are numerous data-driven techniques available in the literature. We are particularly interested in techniques which come with some form of guarantee. One widely employed technique is based on Willems' fundamental Lemma [119], which provides a technique for checking if a trajectory is a valid system trajectory, using only a previously collected data, under suitable persistency-of-excitation conditions. Willems' fundamental lemma primarily considers linear time invariant systems, however techniques to extend to non-linear systems through the use of Koopman operators [108]. Unfortunately, Willems' fundamental lemma does not generally handle noisy data, without some knowledge of, or limits on, the noise distribution. Techniques which consider robustness (e.g. by extending to model predictive control [40]) often result in conservative controllers.

Alternatively, indirect system identification methods [76] seek to identify a model

from data, and then design a controller for this model through classical control techniques. Once again, these techniques often make some assumptions on the structure of the noise, and are generally focussed on linear systems. Guarantees on system behaviour are then achieved through guarantees classical control methods, paired with a guarantee on the identified model matching the true system.

Finally, a method which is not restricted to linear systems is that of safe/robust reinforcement learning [7, 78, 117]. These techniques, however, are generally limited to finite state models, and extending to continuous state models is not straightforward.

We therefore seek to develop techniques which are able to provide guarantees for general non-linear models, with no assumptions on the noise distribution. Further, in order to avoid overly conservative results we seek a probabilistic guarantee, that allows for a trade-off between optimality and guarantees, to achieve improved controller performance. Finally, where possible, we seek to avoid learning a model from data, and instead act directly upon that data. We therefore turn to the so-called *scenario approach* [26, 27, 29, 32, 53].

We are thus interested in solving a scenario program, that is, an optimisation program where data (or "scenarios") act as constraints. In particular, we wish to optimise the probability of property satisfaction, so that the property is satisfied with at least the calculated probability on all sampled data. The scenario approach methodology allows one to obtain *probably approximately correct* (PAC) guarantees on the generalisability of the outcome of a scenario program. Such PAC guarantees are established through bounds on the change of a quantity termed *compression set* (namely, a subset of the data which would return the same result as the entire set) [30, 50, 79]. Of particular importance is the benefit that such a procedure does not require using a separate data-set for validation, allowing one to use all available data for learning.

Techniques to exploit these data may be split into two approaches; abstraction-

based and abstraction-free. Abstraction-based techniques consider synthesising an *abstraction*, a simpler model of the system which is formally shown to be related to the concrete model, and thus allow to transfer the obtained results (safety guarantees, and/or synthesised policies) back to the original model [19]. In this thesis, we focus on finite-state abstractions, that is, abstractions which only have a finite number of discrete states. Various approaches exist for creating abstractions of different forms, such as the celebrated counterexample-guided abstraction/refinement approach [37], which iteratively improves an abstract model by analysing erroneous counterexamples. There are also a number of abstraction-based techniques which involve abstractions as Markov models [14, 15, 104, 111]. These Markov models are of particular interest since there are many sophisticated tools for verifying PCTL properties on Markov models [73, 45]

Unfortunately, the creation of such an abstraction comes with a number of drawbacks. To name a few, choosing the structure of the abstraction requires domain-specific knowledge [75, 81], and synthesising the abstraction may require a large number of samples to accurately reflect a complex system [15, 104]. Hence, we may wish to consider abstraction-free techniques, which provide guarantees directly, without building an abstraction, at the cost of a potentially more complex verification. We consider abstraction-free techniques for continuous-state systems (hence necessarily requiring an infinite number of states, in contrast to the finite-state abstractions we consider).

One abstraction-free tool that is of particular interest are *certificates* [5, 8, 94]. The goal is to determine a function over the system's state space that exhibits certain properties. A well-investigated example of such certificate is that of a Lyapunov function, used to verify that dynamics satisfy some stability property [77]. But one can consider extensions to reachability, safety and reach-while-avoid (RWA) certificates. Overviews of techniques for certificate learning can be found in [5, 44].

This thesis deals with verification and control for both finite-state models (likely

abstractions of more complex systems, although we do not consider how such an abstraction is generated), as well as abstraction-free techniques for more complex systems. In order to tackle this problem, we present novel results for scenario optimisation, and accompany these with PAC guarantees.

1.2 Thesis outline and contributions

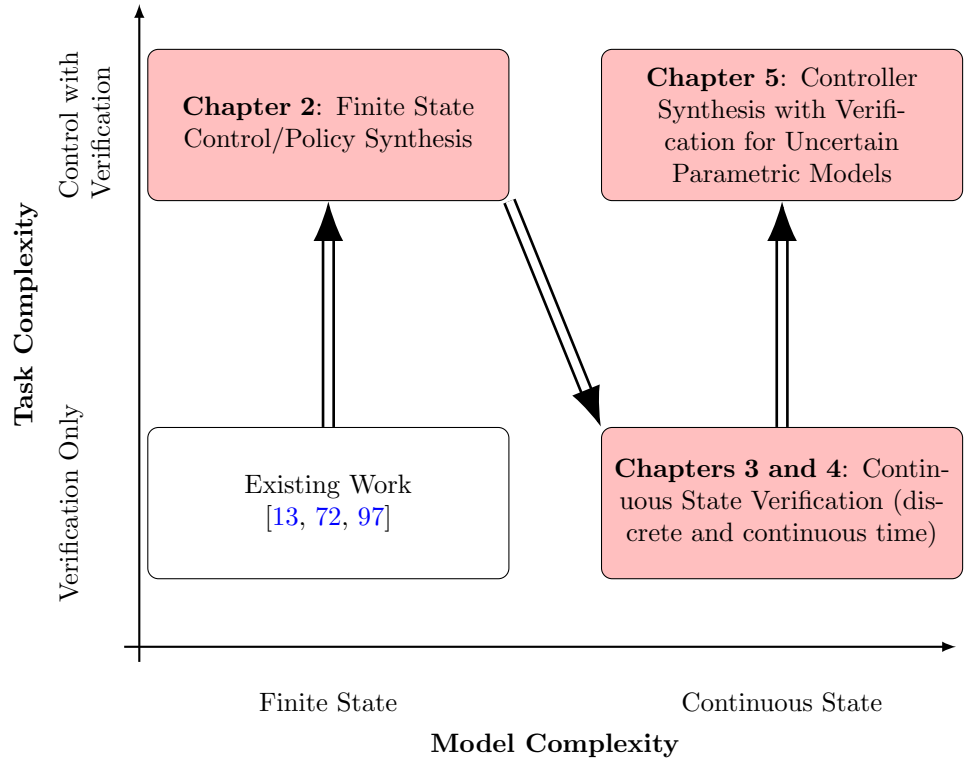


Figure 1.1: Overview of contributions within this thesis with respect to model complexity and task complexity.

The main contributions in this thesis are the novel techniques for finite-state controller synthesis, and continuous-state certificate synthesis, as well as a novel optimisation algorithm for non-convex scenario optimisation. We provide a high-level overview of the structure of this thesis in Figure 1.1. Below we include a detailed overview of each chapter, including its main contributions.

- In Chapter 2, we consider an uncertain parametric Markov decision process

as a given finite-state abstraction, and consider the problem of policy (or controller) synthesis. In particular, we develop techniques for the synthesis of policies across a range of policy classes, which are robust to the uncertainty in the parameterisation. These policies are synthesised to maximise the probability of satisfying some PCTL specification and we provide guarantees on the probability of our synthesised policies satisfying this specification for a newly sampled parameter. This chapter is based on the work in

  - L. Rickard, A. Abate, and K. Margellos. Learning robust policies for uncertain parametric Markov decision processes. In *L4DC*, volume 242 of *Proceedings of Machine Learning Research*, pages 876–889. PMLR, 2024.

- Chapter 3 explores techniques for certificate synthesis. In particular, we consider a continuous-state system in discrete time, and develop a methodology for the data-driven synthesis of reachability, safety and reach-while-avoid certificates. At the core of this methodology is a non-convex scenario program, and we introduce a novel algorithm to solve this program and provide strong PAC guarantees. This chapter is based on

  - L. Rickard, A. Abate, and K. Margellos. Data-driven certificate synthesis. *Autom.*, 185:112798, 2026

- In Chapter 4, we continue investigating certificate synthesis, but now consider continuous time models. This introduces an additional layer of complexity, since we are unable to (in general) store continuous time trajectories, since they are now functions from the chosen time interval to states. Instead, we consider utilising time-discretised approximations consisting of a finite number of sample times within the time interval. Then, we alter our algorithm to allow us to provide PAC guarantees on property satisfaction for complete continuous time trajectories. This chapter is based on the work in

  - L. Rickard, A. Abate, and K. Margellos. Continuous-time data-driven

barrier certificate synthesis. In *CDC*, pages 5794–5799. IEEE, 2025

- Chapter 5 expands upon the certificate synthesis problem explored thus far to include controller synthesis, as well as extending to uncertain parametric models. This allows us to synthesise a controller to maximise the probability of property satisfaction, and provide a certificate validating that controller, with a PAC guarantee on the probability of satisfying the specification for a newly sampled parameter. The extension to uncertain parametric models also allows us to incorporate some model knowledge (where this is readily available), whilst still allowing for the uncertainty in the model. This chapter contains material which is currently being prepared for publication.
- Finally, Chapter 6 summarises the contributions contained in this thesis and provides some discussion on future research directions.

2 Finite State Control/Policy Synthesis

2.1 Introduction

Verifying the safety of complex dynamical systems is an important challenge [70], with applications including unmanned aerial vehicles (UAVs) [123] and robotics [23]. Ensuring a safety property may require verifying satisfaction of complex and rich formal specifications in the process of uncertainty arising from, for example, inaccurate modelling or process noise. A vital aspect of verification lies in finding abstractions that encompass this uncertainty, whilst accurately modelling system dynamics to aid optimal control of such systems.

There are a number of techniques for abstracting dynamical systems, such as the widely celebrated counter-example guided abstraction/refinement approach [36]. Alternatively, techniques make use of the so-called scenario approach in order to develop their models [15, 104]. We discuss these in further detail in Section 2.2.

One useful abstract model is that of parametric Markov decision processes (pMDPs) [43, 56, 67], and particularly their probabilistic counterpart uncertain pMDPs (up-MDPs) [13, 107]. Parametric MDPs extend MDPs by parameterising their transition function; any choice of parameters induces a standard MDP. Hence, a pMDP represents a family of MDPs, differing only in their transition function. By drawing these parameters from a (possibly unknown) distribution, we define an uncertain

parametric MDP. The parameterisation of the model allows for the introduction of some structure, thus avoiding overly conservative models. Here, we make no assumptions on the structure of the parameterisation, and could equivalently have a direct distribution over MDPs within an uncertain set, hence arriving at a problem closely related to that of robust MDPs [64, 86, 118]

A useful tool for expressing specifications on (up)MDPs lies in temporal logic [91], a rich language for specifying behaviours of a system [19]. One particular language of interest is that of Probabilistic Computation Tree Logic (PCTL [58]), an extension of Computation Tree Logic which allows for probabilistic quantification of properties. This language can be used to describe probabilistic specifications on system behaviour, with these probabilities allowing for uncertainty. Often, it is of interest to learn a policy which ensures satisfaction of the specification [55], even under different realisations of the uncertainty.

One common approach for verifying pMDPs is to solve the so-called *parameter synthesis* problem, finding parameter sets that satisfy the specification. Typically, only a single set of parameters is of interest [41, 43, 80]. Instead, we wish to synthesise policies that are probabilistically robust to uncertainty over the entire parameter space.

In this chapter, we investigate the following problem. Given a upMDP, with a possibly unknown distribution over parameters, we aim at synthesising a policy, accompanied by a probabilistic certificate, such that, for an MDP defined with a randomly drawn parameter set, the probability that the policy satisfies a given specification on that MDP is guaranteed by the computed certificate. To solve this problem, we capitalise on advancements from the so-called *scenario approach* literature [24, 31, 29, 53]. We sample a finite number of parameter sets, and compute a policy that is robust to any sample within that set. Then, by leveraging results from [53] we provide probably approximately correct (PAC) guarantees that the policy will be robust to a newly sampled parameter set. Importantly, these techniques al-

low us to provide guarantees based only on finite samples, without knowledge of the underlying distribution, and with no assumptions on the shape of this distribution or the geometry of its support set.

Since we are optimising a policy for multiple MDPs, our work is similar to developments on multiple environment MDPs (MEMDPs [99, 115]). These techniques consider finding policies which are optimal for a finite set of MDPs. In contrast to our approach, these methods assume to have complete knowledge of possible environments, whereas we wish to be robust to new, unseen, environments. Previous work in this area typically required overly conservative assumptions on the parameterisation, restricting to models where the uncertainty is independent across different states [97], or across different state-action pairs [72]. In the robust MDP literature, such assumptions are referred to as s-rectangular or (s, a)-rectangular ambiguity sets [118]. Some recent approaches [13] have relaxed this assumption, but required full knowledge of the parameter sets at runtime in order to apply a policy.

Our work makes no assumption on the parameterisation of the upMDP, and learns a single policy which may be applied at runtime with no explicit parameter knowledge, whilst still offering PAC-type guarantees. Our main contributions can be summarised as follows

1. We formally discuss different policy classes for an MDP, and investigate their associated probabilistic guarantees. This policy classification is interesting per se as it clarifies subtle differences among them.

2. We propose a new gradient-based algorithm to learn an optimal *robust* policy for a given specification in an upMDP; we demonstrate the benefits of this scheme over more traditional solution paradigms via extensive numerical investigation and set the basis for a theoretical convergence analysis in future work.

3. We use recent advancements in scenario approach theory to accompany our

learned policy with non-trivial guarantees on the probability that newly sampled MDPs meet certain specifications.

2.2 Background

2.2.1 Markov Decision Processes

A Markov Decision Process (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \rho, \gamma)$. Where $\mathcal{S} = \{s_0, \ldots, s_N\}$ is a finite set of states, with initial distribution $\rho : \mathcal{S} \to (0, 1)$, $\mathcal{A} = \{a_0, \ldots, a_M\}$ a finite set of actions, $T \colon \mathcal{S} \times \mathcal{A} \to \text{Dist}(\mathcal{S})$ is a probabilistic transition function, with $\text{Dist}(\mathcal{S})$ the set of all probability distributions over $\mathcal{S}$ [16], and $\gamma \in (0, 1)$ is some discount factor, that may be interpreted as there existing some probability equal to $1 - \gamma$ of entering a "failure" state.

We call a tuple (s, a, s') with probability $T(s, a)(s') > 0$ a *transition*. By absorbing state, we refer to a state $s \in \mathcal{S}$ in which all transitions return to that state with probability 1 so that $T(s, a)(s) = 1$, for all $a \in \mathcal{A}$, these typically being some goal or critical states. An infinite trajectory of an MDP is a sampled sequence of states and actions $\zeta = s_I a_0 s_1 a_1 \ldots$, where actions are chosen according to some *policy*, which we define in the sequel.

2.2.2 Policy Classes

We consider three distinct classes of policy to determine actions in this MDP. Namely, deterministic (also called pure), mixed (also called randomised), and behavioural policies. These policies are denoted by $\pi^D \in \Pi^D, \pi^M \in \Pi^M, \pi^B \in \Pi^B$, referring to a deterministic, mixed and behavioural policy respectively. Specifically,

$$\pi^D \colon \mathcal{S} \to \mathcal{A}, \qquad \pi^M \in \text{Dist}(\Pi^D), \qquad \pi^B \colon \mathcal{S} \to \text{Dist}(\mathcal{A}). \tag{2.1}$$

In other words, a deterministic policy shows which action to pick at each state; a mixed policy provides directly a distribution over deterministic policies, so that we sample one policy at the start of a trajectory and follow it throughout; a behavioural policy gives us a distribution over actions at each state, so that we sample one action at each state of a trajectory. For MDPs, it can be shown that deterministic policies suffice for optimality, but this does not hold for more complex models.

We denote by $\pi^M(\pi^D)$ the probability of choosing π^D under the distribution defined for the mixed policy, and by $\pi^B(s)(a)$ the probability of choosing action a in a state s, under the policy defined by π^B. Mixed and behavioural policies offer two semantically different options to resolve the non-determinism in a probabilistic manner.

2.2.3 Uncertain Parametric MDP

An Uncertain Parametric MDP (upMDP) is a tuple $\mathcal{M}_v^{\mathbb{P}} = (\mathcal{S}, \mathcal{A}, \mathcal{V}, \mathcal{T}, \mathbb{P}, \rho, \gamma)$. Similar to an MDP, but the probabilistic transition function is now $\mathcal{T} \colon \mathcal{S} \times \mathcal{A} \times \mathcal{V} \rightharpoonup \text{Dist}(\mathcal{S})$, so that we denote by $\mathcal{T}_v(s,a)(s')$ the probability of transitioning to state s', given action a in s, with parameters $v \in \mathcal{V}$ (i.e. the transition function is *parameterised* by $v \in \mathcal{V}$). We denote by $(\mathcal{V}, \mathcal{F}, \mathbb{P})$ a probability space, where $\mathcal{F}$ is a σ-algebra and $\mathbb{P} \colon \mathcal{V} \to [0,1]$ is a probability measure on the parameter set $\mathcal{V}$. Thus, when a set of parameters v is sampled, we extract a concrete MDP $\mathcal{M}[v]$. We do not impose any structure for this parameterisation, and only require that $\mathcal{M}[v]$ be a well-defined MDP. To uncover a trajectory in this upMDP we first sample a set of parameters v from $\mathbb{P}$, and then follow a trajectory in the resulting MDP. We denote by $\mathbb{P}^N$ the product measure associated with N sampled parameter sets.

2.2.4 Abstracting Continuous State Systems

There are a number of techniques for synthesising a finite model (such as a upMDP) from a continuous (discrete time) dynamical system. In the case where the dynamics

and noise distribution are known, this process is relatively straightforward as the state space can be gridded, and transitions calculated based on the (parametric) noise distribution.

One alternative technique explored in [15, 104, 84] considers forming an interval MDP (where transitions probabilities are shown to lie within some interval). This is achieved through the use of scenario approach techniques to guarantee state transitions, using samples of these transitions. Finally, a PAC guarantee is provided that the interval MDP captures the dynamics of the true system. The primary limitation in applying our techniques to models synthesised with this approach is the presence of these intervals, in the case of infinite data per parameter these intervals collapse to a single value, providing an uncertain parametric MDP.

In practice, this is not achievable, and techniques in [74] may be of benefit, where a stochastic bisimulation function is employed, alongside maximum likelihood estimation, to form a finite MDP abstraction. This technique, however, requires knowledge of some Lipschitz constants of this stochastic bisimulation function and of the stochasticity so that maximum likelihood estimation may be employed. This approach provides a single-level probabilistic guarantee on the validity of the abstraction, reducing the layers of nested probabilistic statements needed to interpret the guarantees compared to interval MDP-based techniques.

In this chapter, we do not discuss further the specific abstraction technique, and instead assume that this is selected appropriately depending on the availability of data and of model knowledge.

2.2.5 Probabilistic Computation Tree Logic

We consider having a set of atomic propositions AP (e.g. denoting a goal or safety), and a labelling function $L\colon \mathcal{S} \to 2^{AP}$ which assigns atomic propositions to states.

Then, probabilistic computation tree logic (PCTL) depends on the following syntax:

$$\Phi ::= \text{true} \,|\, p \,|\, \neg\Phi \,|\, \Phi \wedge \Phi \,|\, \mathsf{P}_{\sim\lambda}(\psi) \qquad \psi ::= \Phi \,\mathsf{U}\, \Phi \,|\, \Phi \,\mathsf{U}_k\, \Phi \tag{2.2}$$

Here, $\sim \in \{<, \leq, \geq, >\}$ is a comparison operator and $\lambda \in [0, 1]$ a probability threshold; PCTL formulae Φ are state formulae, which can in particular depend on path formulae ψ. For a state $s \in \mathcal{S}$, the satisfaction relation $s \models \phi$ may be defined inductively as follows

$$s \models \text{true} \qquad \text{always} \tag{2.3}$$

$$s \models p \quad \Leftrightarrow \quad p \in L(s) \tag{2.4}$$

$$s \models \neg\Phi \quad \Leftrightarrow \quad s \not\models \Phi \tag{2.5}$$

$$s \models \Phi_1 \wedge \Phi_2 \quad \Leftrightarrow \quad s \models \Phi_1 \text{ and } s \models \Phi_2. \tag{2.6}$$

Whilst for a path $(s_0, s_1, \dots)$, the satisfaction relation is defined as

$$(s_0, s_1, \dots) \models \Phi_1 \,\mathsf{U}\, \Phi_2 \quad \Leftrightarrow \exists T \geq 0 \quad : (s_T \models \Phi_2) \;\wedge\; (s_t \models \Phi_1, \forall t < T) \tag{2.7}$$

$$(s_0, s_1, \dots) \models \Phi_1 \,\mathsf{U}_k\, \Phi_2 \;\Leftrightarrow \exists T \in [0, k] : (s_T \models \Phi_2) \;\wedge\; (s_t \models \Phi_1, \forall t < T). \tag{2.8}$$

The probabilistic operator $\mathsf{P}_{\sim\lambda}(\psi)$ requires that paths generated from the initial distribution satisfy a path formula ψ with total probability exceeding (or below, depending on $\sim$) some given threshold λ. For our analysis, we consider only infinite horizon until $\Phi \,\mathsf{U}\, \Phi$, but note that extensions to include the finite time bounded-until operator can be achieved as discussed in [87].

The satisfaction relation $\mathcal{M}[v] \models_\pi \Phi$ defines whether a PCTL formula Φ holds true, when following policy π in the concrete MDP. Formal definitions for semantics and model checking are provided in [58, 16].

2.2.6 PCTL Satisfaction

This satisfaction relation defines slightly different semantics depending on the class of policy. We now explore these semantics for the policy classes identified in section 2.2.2, namely, deterministic, mixed and behavioural, respectively.

$$\mathcal{M} \underset{\pi^D}{\models} \mathsf{P}_{\geq\lambda}(\psi) \iff \sum_{\{\zeta\colon \zeta \models \psi\}} \rho(s_0)\gamma T(s_1 \mid s_0, \pi^D(s_0))\gamma T(s_2 \mid s_1, \pi^D(s_1)) \cdots \geq \lambda \tag{2.9}$$

$$= \sum_{\{\zeta\colon \zeta \models \psi\}} P_{\pi^D}(\zeta) \geq \lambda,$$

$$\mathcal{M} \underset{\pi^M}{\models} \mathsf{P}_{\geq\lambda}(\psi) \iff \sum_{\pi^D \in \Pi^D} \pi^M(\pi^D) \cdot \left(\sum_{\{\zeta\colon \zeta \models \psi\}} P_{\pi^D}(\zeta) \right) \geq \lambda, \tag{2.10}$$

$$\mathcal{M} \underset{\pi^B}{\models} \mathsf{P}_{\geq\lambda}(\psi) \iff \sum_{\{\zeta\colon \zeta \models \psi\}} \rho(s_0)\gamma T(s_1 \mid s_0, a_0)\pi^B(a_0 \mid s_0)\gamma T(s_2 \mid s_1, a_1) \cdots \geq \lambda \tag{2.11}$$

$$= \sum_{\{\zeta\colon \zeta \models \psi\}} P_{\pi^B}(\zeta) \geq \lambda.$$

We use the $P_\pi(\zeta)$ to refer to the probability of uncovering a trajectory when playing a given policy, and trajectories are truncated at the first state in which ψ is satisfied. Note that both eqs. (2.10) and (2.11) contain terms relating to the probability distributions introduced in the policy classes. Instead, eq. (2.9) only considers uncertainty in the model itself.

For MDPs, we note that there *is* a link between property satisfaction through behavioural and mixed policies by Kuhn's theorem [10]. Further, for an MDP, it is always possible to find memoryless behavioural policies that are realisation-equivalent to mixed policies, as demonstrated below.

First, consider any behavioural policy $\pi^B \in \Pi^B$. We know that for an MDP, there will exist a deterministic policy with maximum probability of satisfying a

PCTL formula, which we call $\overline{\pi^\star}$ (see [16] for a proof of this). More formally, we have that

$$\overline{\pi^\star} \in \arg\max_{\pi^B \in \Pi^B} \max_{\lambda \in [0,1]} \mathsf{P}_{\geq\lambda}\psi,$$

recognising that a deterministic policy is a behavioural policy with all probability mass assigned to a single action in each state.

We define a solution function as the maximum probability of satisfying a given PCTL path formula for a given policy, on a concrete MDP, as

$$\mathrm{sol}^{\pi}_{\mathcal{M}}(v;\psi) := \max_{\lambda \in [0,1]} \lambda \text{ s.t. } \mathcal{M}[v] \models \mathsf{P}_{\geq\lambda}\psi. \tag{2.12}$$

Hence, for an MDP induced by a given sample $v, \mathrm{sol}^{\pi^B}_{\mathcal{M}}(v;\psi) \leq \mathrm{sol}^{\overline{\pi^\star}}_{\mathcal{M}}(v;\psi)$. If we consider trying to maximise for $\neg\psi$, it is obvious that there exists another optimal deterministic policy, $\underline{\pi^\star}$, (since we are now optimising for a different, but equally valid PCTL formula).

$$\underline{\pi^\star} \in \arg\max_{\pi^B \in \Pi^B} \max_{\lambda \in [0,1]} \mathsf{P}_{\geq\lambda}\neg\psi = \arg\min_{\pi^B \in \Pi^B} \min_{\lambda \in [0,1]} \mathsf{P}_{\leq\lambda}\psi,$$

This policy will then have a minimum probability of satisfying the original formula, ψ so that $\mathrm{sol}^{\pi^B}_{\mathcal{M}}(v;\psi) \geq \mathrm{sol}^{\underline{\pi^\star}}_{\mathcal{M}}(v;\psi)$. Hence, we have

$$\mathrm{sol}^{\underline{\pi^\star}}_{\mathcal{M}}(v;\psi) \leq \mathrm{sol}^{\pi^B}_{\mathcal{M}}(v;\psi) \leq \mathrm{sol}^{\overline{\pi^\star}}_{\mathcal{M}}(v;\psi).$$

Finally, since a mixed policy defines a finite distribution over deterministic policies, it is straightforward to see that

$$\mathrm{sol}^{\pi^M}_{\mathcal{M}}(v;\psi) = \sum_{\pi^D \in \Pi^D} \pi^M(\pi^D) \cdot \mathrm{sol}^{\pi^D}_{\mathcal{M}}(v;\psi).$$

Then, since $\mathrm{sol}^{\pi^B}_{\mathcal{M}}(v;\psi)$ is bounded from above and below there must exist a convex combination of these bounds that is equal to $\mathrm{sol}^{\pi^B}_{\mathcal{M}}(v;\psi)$, and this convex combina-

tion defines the realisation equivalent mixed policy.

A similar argument can be made for finding a behavioural policy for a given mixed policy $\pi^M \in \Pi^M$. However, for behavioural policies it does not, in general, hold that a convex combination of policies will provide an equally weighted convex combination of values. Specifically, for $\gamma \in (0,1)$, $\pi_1^B, \pi_2^B \in \Pi^B$:

$$\mathrm{sol}_{\mathcal{M}}^{\gamma\pi_1^B+(1-\gamma)\pi_2^B}(v;\psi) \neq \gamma \mathrm{sol}_{\mathcal{M}}^{\pi_1^B}(v;\psi) + (1-\gamma)\mathrm{sol}_{\mathcal{M}}^{\pi_2^B}(v;\psi).$$

We note that there are limited cases where this may in fact hold with equality. For example, for very simple MDPs with a single state having more than one enabled action.

Instead, we rely on the intermediate value theorem. As with the previous subsection, for all $\pi^B \in \Pi^B$, there exist $\underline{\pi^\star}, \overline{\pi^\star}$, such that

$$\mathrm{sol}_{\mathcal{M}}^{\pi^B}(v;\psi) \in [\mathrm{sol}_{\mathcal{M}}^{\underline{\pi^\star}}(v;\psi), \mathrm{sol}_{\mathcal{M}}^{\overline{\pi^\star}}(v;\psi)],$$

and for our given mixed policy, $\mathrm{sol}_{\mathcal{M}}^{\pi^M}(v;\psi) \in [\mathrm{sol}_{\mathcal{M}}^{\underline{\pi^\star}}(v;\psi), \mathrm{sol}_{\mathcal{M}}^{\overline{\pi^\star}}(v;\psi)]$. Hence, if $\mathrm{sol}_{\mathcal{M}}^{\pi^B}(v;\psi)$ can be shown to be continuous in π^B, then it must hold that

$$\exists \pi^B \in \Pi^B \colon \mathrm{sol}_{\mathcal{M}}^{\pi^B}(v;\psi) = \mathrm{sol}_{\mathcal{M}}^{\pi^M}(v;\psi).$$

Since the discount factor is strictly less than 1, there will not be any discontinuities in the solution function[1].

[1] If this assumption is violated, discontinuities may arise in the presence of PCTL formulae with infinite horizons where loops which can be followed with certainty exist. For a concrete example, consider a simple MDP with an initial state and a goal state, and two actions in the initial state corresponding to transitioning to the goal state or staying in the initial state (both with probability 1), with a discount factor equal to 1. The PCTL formula is simply an infinite horizon reach probability. For any policy taking the action to transition to the goal state with a non-zero probability, the formula will be satisfied with probability 1. However, for the single policy which always transitions back to the initial state (i.e. taking the action to transition with probability 0), the reach probability will also be 0. Hence, a discontinuity arises. If the loop probability is less than 1, then this discontinuity is removed. This example clearly also violates our equivalence.

To see this, consider $\mathrm{sol}^{\pi^B}_{\mathcal{M}}(v;\psi)$ encoding a PCTL formula $\Phi = \mathsf{P}_{\geq\lambda}\mathsf{SUG}$, with state s_G satisfying G, this may be expanded as

$$\mathrm{sol}^{\pi^B}_{\mathcal{M}}(v;\psi) =$$
$$\sum_{s\in\mathcal{S}} \rho(s) \sum_{a\in\mathcal{A}(s)} \pi^B(a \mid s) \cdot \Bigg(\gamma T(s_G \mid s,a) +$$
$$\sum_{s'\in\mathcal{S}\setminus s_G} \gamma T(s' \mid s,a) \left[\sum_{a'\in\mathcal{A}(s')} \pi^B(a' \mid s') \cdot \left(\gamma T(s_G \mid s',a') + \sum_{s''\in\mathcal{S}\setminus s_G} \gamma T(s'' \mid s',a') \dots \right) \right] \Bigg).$$

Since this is a summation and multiplication of many terms linear in the policy, and all infinite series involved are convergent, it is continuous in the policy. Thus, the intermediate value theorem will hold, and for any mixed policy there exists an associated behavioural policy which is realisation equivalent (although the statement is not constructive). More complex properties may be considered by defining a Q-function as in section 2.3.3.

In Uncertain Parametric MDPs the link between behavioural and mixed policies may not be satisfied.

2.2.7 Robust Policies

We are interested in synthesising *robust policies* for upMDPs. That is, policies that maximise the probability of satisfying a PCTL formula ψ, within some given risk tolerance. Thus, we are interested in solving the chance-constrained optimisation program

$$\max_{\pi\in\Pi,\ \lambda\in[0,1]} \lambda \quad \text{subject to} \quad \mathbb{P}\left\{ v \in \mathcal{V} | \mathcal{M}[v] \underset{\pi}{\models} \mathsf{P}_{\geq\lambda}(\psi) \right\} \geq 1-\epsilon. \tag{2.13}$$

for some a priori chosen risk level $\epsilon \in (0,1)$. There is an inherent trade-off here between risk and satisfaction probability: namely, for a small risk we might obtain a small satisfaction probability, and vice versa. For brevity, we discuss only max-

imising λ, but note that the minimisation for $\sim \in \{<, \leq\}$ can be obtained trivially. Note that here we optimise over policies π in a generic set Π; the exact policy class from the ones of section 2.2.2 will be specified in the sequel.

2.3 Robust Policy Synthesis

Problem 1. *Given an upMDP $\mathcal{M}_v^{\mathbb{P}}$, PCTL formula ψ and risk level ϵ, find a robust policy π and maximum satisfaction probability λ, such that, with probability $1-\epsilon$ playing policy π on a newly sampled MDP will satisfy the PCTL formula with satisfaction probability at least λ.*

Due to the chance constraint on the parameter distribution $\mathbb{P}$, this problem is a semi-infinite optimization program (having a finite number of optimisation variables, but infinite constraints), and is generally intractable. Further, we may not have access to an analytical form for $\mathbb{P}$. Thus, we turn to a sample based analogue to this problem. In section 2.4, we investigate how solutions to this problem can provide guarantees to the original chance-constrained problem.

Consider an upMDP $\mathcal{M}_v^{\mathbb{P}}$ and N i.i.d. sampled parameter sets (or *scenarios*) $\mathcal{U}_N = \{u_1, \ldots, u_N\}$ sampled from $\mathbb{P}$. In this section, we investigate how to learn a robust policy which maximises the probability of satisfying a PCTL formula ψ, under the worst case realisation of the parameters, i.e. we investigate the following *scenario program* associated to eq. (2.13)

$$\max_{\pi \in \Pi,\ \lambda \geq 0} \lambda \qquad \text{subject to } \mathrm{sol}^{\pi}_{\mathcal{M}}(u; \psi) \geq \lambda,\ \forall u \in \mathcal{U}_N. \tag{2.14}$$

Where we recall that $\mathrm{sol}^{\pi}_{\mathcal{M}}(u; \psi)$ refers to the maximum probability of satisfying a formula ψ, in the concrete MDP $\mathcal{M}[u]$, under policy π.

This scenario program problem coincides closely with the concept of Multiple Environment MDPs (MEMDPs) [99], but differs in some key ways. When finding an optimal policy for MEMDPs, one wishes to find the policy that is optimal for

the given environments. Instead, our goal to find a policy that will be robust to an unseen environment. Furthermore, the concrete chance constrained problem we study is drastically different to MEMDPs since it involves continuous parameters rather than a finite set of environments.

It is shown in [99] that infinite memory policies are required for optimality in MEMDPs. In our setting, infinite memory policies will indeed lead to optimal satisfaction probabilities in the sample set, but suffer when generalising to new samples. This can be seen as a problem of overfitting, and may also lead to a deterioration in our guarantees.

In the sequel, we provide different solution methodologies for eq. (2.14), where each methodology is based on a different class of policies.

2.3.1 Solution by Interval MDPs (under deterministic policies)

A straightforward approach to solve this problem is to ignore the structure induced by the parameterisation, and to build instead an interval MDP, where each transition is only known up to an interval. Each interval can be constructed by looking at each transition in turn and finding the minimum and maximum probabilities from the sampled MDPs. Such techniques are examined further (generally when no parameterisation is available) in [72] and are also closely aligned with the concept of (s, a)-rectangularity [118]. The resulting policy is in general very conservative, since it considers the worst case probability for every single transition. In reality, it is likely to be the case that the worst case transitions are unlikely to co-occur (for example, in a UAV motion planning problem like the one considered in table 2.2 the wind may push us left or right into an obstacle, but is unlikely to do both).

2.3.2 MaxMin Game (under deterministic and mixed policies)

The problem in eq. (2.14) can be rewritten as a MaxMin problem

$$\max_{\pi \in \Pi} \min_{u \in \mathcal{U}_N} \text{sol}^{\pi}_{\mathcal{M}}(u; \psi),$$

which can be seen as a two player zero-sum game. In which we have a policy player, and the samples act as an adversary. The policy player's actions are the set of all deterministic policies Π^D for the upMDP, providing a finite, but potentially very large $\mathcal{O}(|\mathcal{A}|^{|\mathcal{S}|})$, action set. The sample adversary's actions are the set of samples. Given sample u and policy π^D, the reward to the policy player is $r_p(\pi^D, u) = \text{sol}^{\pi^D}_{\mathcal{M}}(u; \psi)$.

2.3.2.1 Deterministic Policies

We may solve this as a Stackelberg game [112, 122], with the adversary as the leader, to obtain an optimal deterministic policy. To achieve this, the sample adversary acts as the leader and finds a minimising sample for *each* possible deterministic policy, that is, for every policy π^D, the adversary solves

$$r_p(\pi^D) = \min_{u \in \mathcal{U}_N} \text{sol}^{\pi^D}_{\mathcal{M}}(u; \psi), \tag{2.15}$$

then the policy agent acts as the follower and chooses a deterministic policy which maximises the reward $r_p(\pi^D)$. In this case, we are working with finite sets of actions for both agents, and so we can consider simply enumerating over all choices. The resulting policy is the optimal deterministic policy, but may not be a Nash equilibrium policy (see theorem 2.3.1). To see this note that if the sample adversary takes a fixed strategy (as opposed to always being allowed to choose the worst case), the policy agent may be able to improve their reward.

2.3.2.2 Mixed Policies

Alternatively, a mixed strategy set (s_p, s_σ), contains finite probability distributions over possible actions of each player, respectively, and returns the reward $r_p(s_p, s_\sigma) = \sum_{\pi^D \in \Pi^D} \sum_{u \in \mathcal{U}_N} s_p(\pi^D) \cdot s_\sigma(u) \cdot \text{sol}_{\mathcal{M}}^{\pi_D}(u; \psi)$, or in matrix form $s_p R s_\sigma$, where matrix $R \in \mathbb{R}^{|\Pi_D| \times N}$ has elements $R_{\pi_D^i, u_j} = \text{sol}_{\mathcal{M}}^{\pi_D^i}(u_j; \psi)$. Since the game consists of a finite number of players, each with a finite set of pure strategies, then by allowing players to play mixed strategies, it can be shown that a Nash equilibrium of the game will exist, and further, that all Nash equilibria have the same value. Note that the mixed strategy defines a mixed policy $s_p = \pi^M$.

Theorem 2.3.1 (Nash Equilibrium [83, 52])**.** *For a two player zero-sum game, there exists at least one mixed strategy profile $s^\star = (s_p^\star, s_\sigma^\star)$, such that*

$$r_p(s_p^\star, s_\sigma^\star) \geq r_p(s_p, s_\sigma^\star), \forall s_p \in S_p, \qquad r_p(s_p^\star, s_\sigma^\star) \leq r_p(s_p^\star, s_\sigma), \forall s_\sigma \in S_\sigma. \tag{2.16}$$

If both inequalities are strict, then there is a single unique Nash equilibrium, called a strict Nash equilibrium. Otherwise, there is a set of Nash equilibria, all having equal value.

Finding Nash equilibrium strategies in this game is relatively straightforward, and there are a number of existing methods for solving the game (for example, the Porter-Nudelman-Shoham (PNS) algorithm [93, Algorithm 1] or fictitious play methods [59]). Thus, we can simply build the reward matrix R, by considering each determinsitic policy and sample in turn, and pass this to one of these algorithms. Unfortunately, this method is computationally very expensive, with the size of the reward matrix being $\mathbb{R}^{N \times |\mathcal{S}|^{|\mathcal{A}|}}$, and algorithms to solve such games being exponential in the size of this matrix.

2.3.3 Subgradient Ascent (under behavioural policies)

Under the choice of behavioural policies, we propose an alternative method that avoids computing a full payoff matrix for every deterministic policy. Taking inspiration from [20], we rewrite the solution function as

$$\mathrm{sol}^{\pi^B}_{\mathcal{M}}(u;\psi) = \sum_{s\in\mathcal{S}} \eta^u_{\pi^B}(s) \sum_{a\in\mathcal{A}} Q^u_{\pi^B}(s,a)\pi^B(s)(a),$$

where $\eta^u_{\pi^B}(s)$ is the discounted state-occupancy measure $\eta^u_{\pi^B}(s) = (1-\gamma)\sum_{k=0}^{\infty}\gamma_k P_k^{\pi^B}(s)$, with $P_k^{\pi^B}(s)$ the probability of uncovering state s at time k, and $Q^u_{\pi^B}(s,a)$ is a Q-function, defined to model the PCTL requirement, as follows.

Note that all operators in PCTL may be derived from the until operator [58], as such we consider only formulae with this operator.

Then, our formula is of the form

$$\Phi = \mathsf{P}_{\geq\lambda}\mathrm{SUG},$$

and we consider states $\mathcal{S}_S, \mathcal{S}_G \subseteq \mathcal{S}$ to refer to the states labelled with atomic propositions S and G respectively.

For a given sampled MDP $\mathcal{M}[v]$, we can then define an iterative bellman equation $\mathcal{B}^v_{\pi_B} : \mathcal{S} \to [0,1]$ as follows (we consider only behavioural policies here, since the other classes of policy can be solved without the use of a Q-function):

$$\mathcal{B}^v_{\pi_B}(s) := \begin{cases} 1 & \text{if } s \in \mathcal{S}_G, \\ 0 & \text{if } s \notin \mathcal{S}_G \text{ and } s \in \mathcal{S}\setminus\mathcal{S}_S, \\ \sum_{a\in\mathcal{A}} \pi_B(s)(a) \sum_{s'\in\mathcal{S}} \mathcal{T}_v(s,a)(s')\mathcal{B}^v_{\pi_B}(s') & \text{otherwise.} \end{cases} \tag{2.17}$$

This naturally models the probability of a given state s satisfying the PCTL path formula SUG.

Finally, define the Q-function using this bellman equation as

$$Q^{v}_{\pi_B}(s,a) \coloneqq \sum_{s' \in \mathcal{S}} T_v(s,a)(s') \mathcal{B}^{v}_{\pi_B}(s'). \tag{2.18}$$

For nested formulae, we simply consider using a computation tree (as is done in [104]). Thus, we break a nested formula down until the leaves represent simple reach-avoid formulae $\mathsf{P}_{\geq\lambda}\mathsf{SUG}$, we optimise the probability of satisfying these formulae, and find states exceeding (or below, depending on the direction of the inequality) the specified bound λ. States that satisfy the bound can be passed up the tree as goal states (if appearing to the right of an until operator), or as safe states (if on the left of an until operator).

Intuitively, $\eta^{u}_{\pi^B}(s)$ defines the (discounted) fraction of time the system spends in a given state. We fix $\eta^{u}_{\pi^B}$ and $Q^{u}_{\pi^B}$ (as in policy iteration [51]), and find the gradient (given analytically in algorithm 1).

This provides us with a (sub)gradient for a single solution function, however, we are primarily interested in the pointwise minimum amongst a set of solution functions. Hence, we turn to subdifferentials (or *subgradients*) [69, 11]. One simple way of finding a valid subdifferential is to find the gradient of one of the current minimum functions.

Algorithm 1 Projected Subgradient Ascent

Require: upMDP $\mathcal{M}$, samples $\mathcal{U}_N$, formula ψ, step size sequence $\{\alpha_0, \alpha_1, \dots\}$

$k \leftarrow 0$
$\pi^B_0 \leftarrow$ uniform random
while not converged **do**
  $u^\star \leftarrow$ random choice$(\arg\min_{u \in \mathcal{U}_N} \mathrm{sol}^{\pi^B_k}_{\mathcal{M}}(u;\psi))$ ▷ Select worst case sample
  $\nabla_{k+1}(s,a) = \eta^{u^\star}_{\pi^B}(s) Q^{u^\star}_{\pi^B}(s,a)$ ▷ Find gradient for worst case
  $\pi^B_{k+1}(s)(a) \leftarrow \mathrm{proj}_{\sum \pi^B(\cdot \mid s)=1}[\pi^B_k + \alpha_k \nabla_{k+1}(s,a)]$ ▷ Gradient step
  $f^{k+1}_* = \max\{f^k_*, \min_{u \in \mathcal{U}_N} \mathrm{sol}^{\pi^B_{k+1}}_{\mathcal{M}}(u;\psi)\}$ ▷ Store record objective
  $k \leftarrow k+1$
end while
return Optimal policy $\pi^\star$ associated with record objective

We initialise with a uniform random policy, select a minimising (worst-case) sam-

ple from the set of minimisers (which may not be a singleton), then find the gradient for this sample, take a step in this direction, and project onto the constraint set. We use a diminishing, non-summable step size α_k, typically employed in subgradient methods. We give numerical evidence that this algorithm exhibits a convergent behaviour in section 2.5.

2.4 Guarantees

Once we have synthesised a policy using one of the methods above, we can accompany each of the policies synthesised according to the aforementioned methodologies with PAC-type guarantees on the satisfaction of the PCTL property under consideration. To achieve this, we use the following recent result in the scenario approach theory.

Given the solution to a scenario program, we are interested in quantifying the risk associated with the solution (i.e. the probability that our solution violates a new sampled constraint). The scenario approach provides us with the techniques to achieve this by considering the number of support constraints. In an optimisation problem, if the removal of a constraint leads to a changed solution, then this constraint is said to be a support constraint. Further, if a solution returned when considering only the support constraints is the same as the solution obtained when employing all samples, then the problem is a non-degenerate problem.

Theorem 2.4.1 (PAC Guarantees [53]). *Consider the optimisation problem in eq.* (2.14)*, with N sampled parameter sets. Given a confidence parameter $\beta \in (0,1)$, for any $k = 0, 1, \ldots, N$, consider the polynomial equation in the variable t*

$$\xi_k(t) = \begin{cases} 1 - \frac{\beta}{6N} \sum_{i=N+1}^{4N} \binom{i}{k} t^{i-k}, & \text{if } k = N, \\ \binom{N}{k} t^{N-K} - \frac{\beta}{2N} \sum_{i=k}^{N-1} \binom{i}{k} t^{i-k} - \frac{\beta}{6N} \sum_{i=N+1}^{4N} \binom{i}{k} t^{i-k}, & \text{otherwise.} \end{cases} \tag{2.19}$$

Solving $\xi_k(t) = 0$ for $t \in [0, +\infty)$, for $k = 0, 1, \ldots, N-1$, we find exactly two

solutions, which we denote with $\underline{t}(k), \overline{t}(k)$ *with* $\underline{t}(k) \leq \overline{t}(k)$*. For* $k = N$*, we find a single solution, which we denote by* $\overline{t}(N)$*, and define* $\underline{t}(N) = 0$*. Let* $\underline{\epsilon}(k) \coloneqq \max\{0, 1 - \overline{t}(k)\}$ *and* $\overline{\epsilon}(k) \coloneqq 1 - \underline{t}(k)$*.*

Assume the problem is non-degenerate[2]*, and has a unique solution. Let* π_N^* *be the optimal policy,* λ_N^* *the optimal satisfaction probability, and* $\tilde{s}_N^*$ *is the number of support samples. Then, for any* $\mathbb{P}$ *it holds that*

$$\mathbb{P}^N\{\underline{\epsilon}(\tilde{s}_N^*) \leq \mathbb{P}\left\{v \in \mathcal{V} \colon \mathcal{M}[v] \underset{\pi^M}{\not\models} \mathsf{P}_{\geq \lambda^\star}(\psi)\right\} \leq \overline{\epsilon}(\tilde{s}_N^*)\} \geq 1 - \beta, \tag{2.20}$$

where $\mathbb{P}\{v \in \mathcal{V} \colon \mathcal{M}[v] \not\models_{\pi^M} \mathsf{P}_{\geq \lambda^\star}(\psi)\}$ *is the risk (the probability that our solution violates the specification for another sample* v*).*

Note that in π_N^* we do not specify the problem class; this is considered in the sequel when we deploy this theorem for each solution methodology from the previous section. Moreover, to determine the number of support samples $\tilde{s}_N^*$, we exploit our problem's structure and, based on the solution methodology adopted, we discuss how an upper-bound on their number can be obtained.

To this end, we provide guarantees for the policy generated by each of the methods of Section 2.3. Consider first the case of the mixed policy determined using a Nash equilibrium solver as per the developments of Section 2.3.2. In this case, finding the number of support samples consists of finding samples which are included in the mixed strategy of the sampled adversary.

Corollary 2.4.1 (PAC Guarantees for Mixed Policies)**.** *Consider the MaxMin game of Section 2.3.2, and let* π_N^* *be the optimal mixed policy* π^M *returned by a Nash equilibrium solver. The number of support constraints may be found as* $\tilde{s}_N^* = |\{u \in \mathcal{U}_N \colon s_\sigma(u) > 0\}|$*, while the satisfaction relation* $\mathcal{M}[v] \not\models_{\pi^M} \mathsf{P}_{\geq \lambda^\star}(\psi)$ *is as defined in*

[2]Non-degeneracy implies that solving the problem using only the samples that are of support results in the same solution (policy in our case) had all the samples been employed. We say that a sample (which gives rise to a constraint in eq. (2.14)) is of support, if removing only that sample results in a different solution. Repeating this procedure, removing samples one-by-one allows identifying the support samples of a given problem.

eq. (2.10).

Consider now the case of a behavioural policy obtained using the subgradient methodology (section 2.3.3). We determine the number of support constraints by finding the active constraints (as the problem is assumed to be non-degenerate, the active constraints are also the support constraints).

Corollary 2.4.2 (PAC Guarantees for Behavioural Policies)**.** *Consider the subgradient algorithm of Section 2.3.2, and let* π^*_N *be the optimal behavioural policy* π^B *returned by Algorithm 1. The number of support samples is given by the active constraints, which are in turn the ones for which the obtained satisfaction probability* $\lambda^\star$ *is tight, i.e.,* $\tilde{s}^*_N = |\{u \in \mathcal{U}_N \colon sol^{\pi^B}_{\mathcal{M}}(u;\psi) = \lambda^\star\}|$*, while the satisfaction relation* $\mathcal{M}[v] \not\models_{\pi^B} \mathsf{P}_{\geq\lambda^\star}(\psi)$ *is as defined in eq.* (2.11).

Finally, consider the construction of an interval MDP under the class of deterministic policies. In this case, our problem is degenerate, since it may be necessary to remove multiple constraints for another policy to become optimal, thus prohibiting the use of Theorem 2. Therefore, we leverage techniques from [32] that do not require imposing a non-degeneracy assumption.

Corollary 2.4.3 (PAC Guarantees for iMDP Policies)**.** *Fix* $\beta \in (0,1)$*, and as in [32, Theorem 1], define* $\mu(N) := 1$*, and for* $\tilde{s}^*_N < N$ *let*

$$\mu(\tilde{s}^*_N) := 1 - \sqrt[N-\tilde{s}^*_N]{\frac{\beta}{N\binom{N}{\tilde{s}^*_N}}}. \tag{2.21}$$

Support samples are those which define the intervals:

$$\begin{aligned}\overline{\tilde{s}^*_N} =& |\{u \in \mathcal{U}_N \colon \\ & \exists (s,a,s'), T_u(s,a)(s') \leq T_v(s,a)(s') \vee T_u(s,a)(s') \geq T_v(s,a)(s'), \\ & \qquad\qquad \forall v \in \mathcal{U}_N\}|,\end{aligned} \tag{2.22}$$

i.e., those with at least one transition probability at an extremum of its interval.

Then (with satisfaction relation defined in eq. (2.9)*) we have*

$$\mathbb{P}^N \left\{ \mathbb{P} \left\{ v \in \mathcal{V} \colon \mathcal{M}[v] \underset{\pi^D}{\not\models} \mathsf{P}_{\geq \lambda^\star}(\psi) \right\} \leq \mu(\overline{\tilde{s}_N^*}) \right\} \geq 1 - \beta. \tag{2.23}$$

2.4.1 Probabilistic Guarantees under a MaxMin Deterministic Policy

Under a deterministic MaxMin policy, our problem is degenerate, since it may be necessary to remove multiple constraints in order for another deterministic policy to become optimal, we therefore leverage techniques from [32] for non-convex problems. This approach requires solving a different equation for the risk, $\mu(\tilde{s}_N^*)$, which must satisfy [32, Theorem 1]. One equation which satisfies these requirements is that in eq. (2.21).

The easiest method for obtaining an upper bound on the size of the support set is to consider the satisfaction probabilities for each deterministic policy in the worst-case MDP $\mathcal{M}[u^\star]$ for the optimal policy. If the satisfaction probability for a different policy is greater than the worst-case, then there must exist at least one sample that prevents us switching to that policy. As an upper bound, we may consider that every such policy is "blocked" by a single unique sample:

Alternatively, to improve this bound, we first find the set of "blocked" policies as above. Then, for each such policy, we find the samples which have an associated satisfaction probability less than our optimal $\lambda^\star$. To calculate the smallest support set, we then must find the smallest set such that at least one blocking sample for every policy is contained within this set. This problem is known as the hitting set problem and can be shown to be NP-hard. Alternatively, we may take a union of all sets to again find an upper bound.

Note that both methods may be computationally expensive if there are many possible deterministic policies.

We then have the following result that bounds the risk:

Corollary 2.4.4 (PAC Guarantees for MaxMin Deterministic Policies)**.** *We make use of eq.* (2.21) *to bound the risk. Bounds on the support constraints may be found as*

$$\overline{\tilde{s}_N^*} = |\{\pi^D \in \Pi^D \colon Pr(\psi \mid \pi^D, \mathcal{M}[u^\star]) \geq \lambda^\star\}|, \tag{2.24}$$

or

$$\Sigma^*(\pi^D) = \{u \in \mathcal{U}_N \colon sol_{\mathcal{M}}^{\pi^D}(u;\psi) \leq \lambda^\star \wedge sol_{\mathcal{M}}^{\pi^D}(u^\star;\psi) \geq \lambda^\star\} \tag{2.25}$$

$$\tilde{s}_N^* = \min_{\mathcal{V} \subseteq \mathcal{U}_N} |\mathcal{V}| \text{ subject to: } \mathcal{V} \bigcap \Sigma^*(\pi^D) \neq \varnothing, \forall \pi^D \in \Pi^D, \tag{2.26}$$

the latter being less conservative, but significantly more computationally expensive. Approximate solutions to eq. (2.26) *exist which may allow for a tradeoff between computation and optimality.*

Then we have that

$$\mathbb{P}^N \left\{ \mathbb{P} \left\{ v \in \mathcal{V} \colon \mathcal{M}[v] \underset{\pi^D}{\not\models} \mathsf{P}_{\geq \lambda^\star}(\psi) \right\} \leq \mu(\tilde{s}_N^*) \leq \mu(\overline{\tilde{s}_N^*}) \right\} \geq 1 - \beta, \tag{2.27}$$

where the only non-determinism arises due to the uncertainty in the transition function.

2.5 Numerical Experiments

We implemented our techniques in Python and made use of the probabilistic model checker Storm [60] to verify PCTL formulae on MDPS, and PRISM [73] for iMDPs. Experiments were run on a server with 80 2.5 GHz CPUs and 125 GB of RAM. The codebase is available at https://github.com/lukearcus/robust_upMDP.

We evaluate our techniques on a number of benchmark models available in the literature [98], as well as a few simpler examples, outlined below

2.5.1 Model Descriptions

2.5.1.1 UAV Motion Planning

We use a model very similar to the UAV motion planning problem introduced in [13] (with the addition of a discount factor). This model consists of a grid world in which the UAV can decide to fly in either of the six cardinal directions (N, W, S, E, up, down). States encode the position of the UAV, the current weather condition (sunny, stormy), and the general wind direction in the valley. The probabilistic outcomes are assumed to be determined only by the wind in the valley and the control action. An action moves the UAV one cell in the corresponding direction, and additionally, the wind moves the UAV one cell in the wind direction with a probability p, (dependent on the wind speed). We assume that the weather and wind-conditions change during the day and are described by a stochastic process. Concretely, parameters describe how the weather affects the UAV in different zones of the valley, and how the weather/wind may change during the day.

We make 3 different assumptions on the distribution over the parameters, and call these "uniform", "x-neg-bias" and "y-pos-bias", they are defined as follows:

2.5.1.1.1 uniform a uniform distribution over the different weather conditions in each zone;

2.5.1.1.2 x-neg-bias the probability for a weather condition inducing a wind direction that pushes the UAV westbound (i.e., into the negative x-direction) is twice as likely as in other directions;

2.5.1.1.3 y-pos-bias it is twice as likely to push the UAV northbound (i.e., into the positive y-direction).

2.5.1.2 Benchmark Models

The benchmark models ("consensus", "brp", "sav", "zeroconf") are taken from [98].

2.5.1.3 Toy Model

In order to compare all algorithms presented, we consider a very small toy model. For this model, each state has two actions, the first which may take us to the next state or the critical state, and the second which may jump us forward two states, or take us to the critical state. We consider there being two non-absorbing states. Then, the probability of an action being successful is a parameter for each state action pair, with the remaining probability being assigned to us ending up in the critical state. Our goal is simply to reach the final state without visiting the critical state.

2.5.1.4 Model Sizes

Further details on model sizes are presented in table 2.1 (where bisimulations can be used to reduce the model size we consider the reduced model). Recall that the MNE algorithm requires iterating through all deterministic policies (on the order of $|\mathcal{A}|^{|\mathcal{S}|}$), hence it should be clear why this algorithm scales so poorly, with the smallest non-toy model having already on the order of 10^{46} policies.

2.5.2 Results

We compare our subgradient algorithm (section 2.3.3) to existing techniques from [13] (column 1) and an implementation of the synthesis based on iMDP, as in [72] (column 2). Unless otherwise stated, we use a numerical tolerance of 10^{-4}, a confidence parameter of $\beta = 10^{-5}$, take $N = 200$ parameter samples, and allow 1 hour of computation time. We provide numerical values for the optimal satisfaction probability $\lambda^\star$, the theoretical risk upper bound $\bar{\epsilon}$, the runtimes, the empirical values of the risk $\tilde{\epsilon}$, and the empirical values of the risk without access to the true parameter set (if parameter access is needed to synthesise a policy, we draw a sample $v_1 \sim \mathbb{P}$ for synthesis, but test on sample, $v_2 \sim \mathbb{P}$). We highlight in grey cases where we discuss next how the methodologies under comparison are compared/outperformed by our

	#Pars	#States	#Actions	#Transitions
consensus				
(2,2)	2	153	2	332
(2,32)	2	2 793	2	6 092
(4,2)	4	22 656	4	75 232
brp				
(256,5)	2	21 032	2	29 750
(4096,5)	2	647 441	2	876 903
sav				
(6,2,2)	2	379	4	1 127
(100,10,10)	2	1 307 395	4	6 474 535
(6,2,2)	4	379	4	1 127
(10,3,3)	4	1 850	4	6 561
zeroconf				
(2)	2	88 858	4	203 550
(5)	2	494 930	4	1 133 781
UAV				
uniform	900	7642	6	56 568
x-neg-bias	900	7642	6	56 568
y-pos-bias	900	7642	6	56 568
Toy Model	4	4	2	10

Table 2.1: Model Sizes.

proposed subgradient algorithm either in terms of the quality of their probabilistic guarantees and/or the computational requirements, and use red text for empirical values which violate a bound.

Our subgradient algorithm is less conservative than a naive iMDP solution, offering non-trivial theoretical risk bounds (for the iMDP approach these bounds are often equal to one), and superior satisfaction probabilities (the iMDP approach considers the worst case probability for every transition). By superior, we mean either higher or lower depending on the optimisation direction (maximisation or minimisation, respectively, denoted by (max) and (min) in table 2.2). The approach in [13] generally outperforms our methods in speed, risk bounds and satisfaction probability. The difference on the theoretical risk bounds lies in the fact that our approach involves solving a problem with a non-trivial number of support constraints, while in [13] they have only one support constraint. As such, it offers tighter guarantees,

Model	Instance	$\lvert\mathcal{S}\rvert \cdot \lvert\mathcal{A}\rvert$	[13] Sat. Prob. $\lambda^\star$	[13] Risk U.B. $\bar{\epsilon}$	[13] Time (s)	[13] Emp. Risk $\tilde{\epsilon}$	[13] Emp. Risk (no pars)	iMDP Solver Sat. Prob. $\lambda^\star$	iMDP Solver Risk U.B. $\bar{\mu}$	iMDP Solver Time (s)	iMDP Solver Emp. Risk $\tilde{\mu}$	iMDP Solver Emp. Risk (no pars)	Subgradient (algorithm 1) Sat. Prob. $\lambda^\star$	Subgradient (algorithm 1) Risk U.B. $\bar{\epsilon}$	Subgradient (algorithm 1) Time (s)	Subgradient (algorithm 1) Emp. Risk $\tilde{\epsilon}$	Subgradient (algorithm 1) Emp. Risk (no pars)
consensus (min)	(2,2)	306	.967	.056	1	.004	.008	.967	1	3	.007	.008	.967	.187	4	.009	.011
	(2,32)	5 586	.996	.056	136	.000	.014	.996	1	76	.000	.000	.996	.516	202	.002	.000
	(4,2)	90 624	-	-	TO	-	-	.936	1	1887	.000	.000	-	-	TO	-	-
brp (max)	(256,5)	42 064	-	-	TO	-	-	.985	1	383	.001	.000	.985	1	923	.000	.001
sav (min)	(6,2,2)	1516	.834	.056	4	.015	.112	.923	1	9	.002	.004	.884	.180	218	.013	.007
	(6,2,2)	1516	.700	.056	5	.000	.017	.721	1	9	.017	.010	.720	.187	76	.011	.017
	(10,3,3)	7400	.446	.056	51	.009	.120	.524	1	50	.017	.015	.526	.147	109	.021	.018
UAV (max)	uniform	45 852	.301	.056	978	.003	.749	.065	1	238	.015	.013	.195	.245	85719[3]	.077	.089
	x-neg	45 852	.198	.056	804	.002	.802	.012	1	210	.000	.000	.102	.440	58612	.068	.071
	y-pos	45 852	.564	.056	811	.003	.755	.050	1	196	.001	.001	.444	.163	92440	.095	.101
Toy (max)	**Model**	8	.326	.056	.11	.008	.106	.323	.155	1.47	.008	.094	.325	.366	8.95	.003	.098

Table 2.2: Experimental Results; the results for the case of the MaxMin Solution are discussed in the text. TO indicates timed-out.

however, these refer to an existential statement. To see this, note that [13] compute a different policy per sample, and provide guarantees only on the existence of a policy for a new sample. On the contrary, we construct a policy that is robust with respect to all samples, and at the same time is accompanied by guarantees on its feasibility properties for unseen samples. Moreover, [13] require access to true parameters at runtime, and a non-trivial amount of online computation to solve the MDP. Without this, their risk bounds may be violated (as seen in red).

Our mixed policy solution (section 2.3.2), timed out on all but the toy model, for which the results are: Sat. Prob: .325, Risk U.B.: .147, runtime: 0.75s Emp. risk: .003, Emp. risk (no pars): .101.

2.5.3 Subgradient Algorithm Behaviour

We provide two plots here which demonstrate convergent behaviour of our proposed subgradient algorithm. The first, in fig. 2.1, is for a small test model where we can use the MNE algorithm to find a true optimal mixed policy (and hence a true optimal satisfaction probability). Here we can see that the distance to this optimal decreases across iterations, and we get remarkably close to the true optimal value.

[3]For the UAV benchmark, we allowed up to 2 days of computation time.

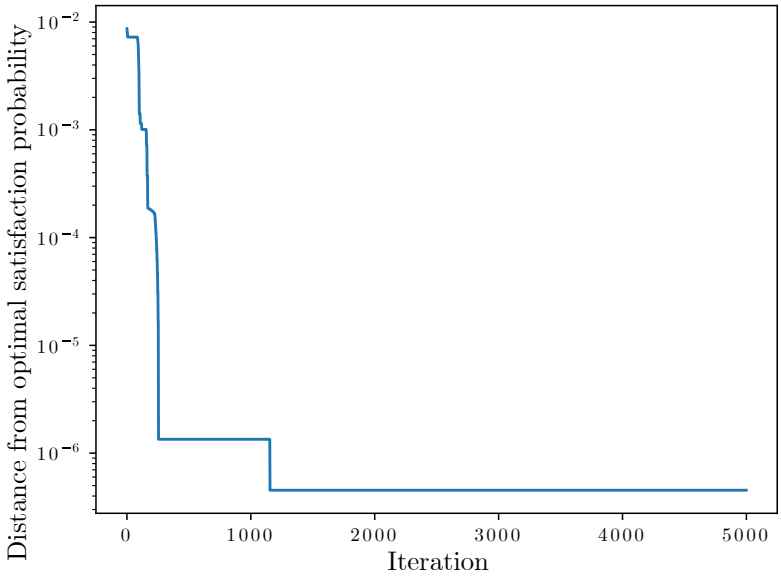


Figure 2.1: Distance from optimal satisfaction probability (found from MNE algorithm) across iterations for small test model.

Note that we do not have a formal proof that a behavioural policy can reach this theoretical optimum, so this quite a remarkable result since it demonstrates that our algorithm does appear to converge to the true optimum.

Second, in fig. 2.2, we consider the UAV benchmark from table 2.2, with uniform wind conditions, and compare the distance to the final satisfaction probability. In this case, the probability once again appears to show convergent behaviour and the distance is continually decreasing.

2.5.4 Runtimes

We have also considered how the various methods presented scale with both the number of sampled parameters, and the size of the MDP. In fig. 2.3, we see how the number of samples N affects the runtime, evidently the work of [13] scales approximately linearly (since they solve each sampled MDP). The deterministic solver based on the MaxMin game and an MNE solver making use of fictitious self play to approximately solve for the equilibrium, clearly scale linearly, since they both must iterate through all samples to construct a reward matrix, but further

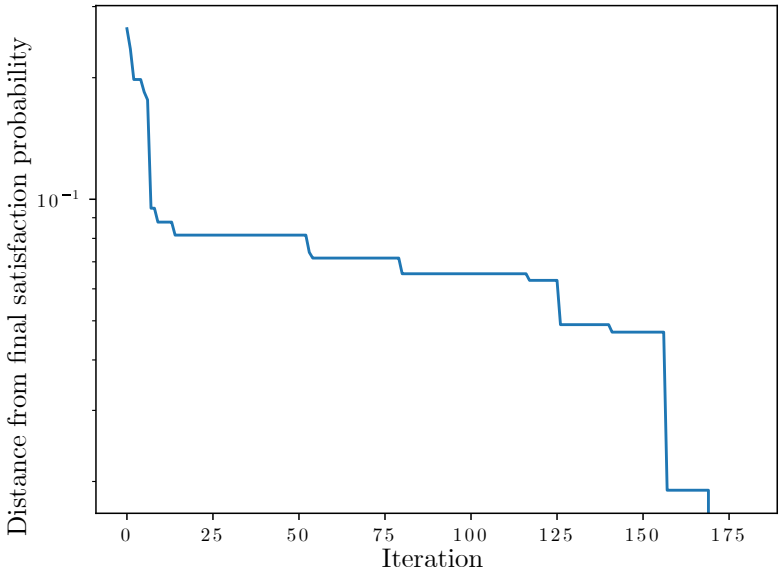


Figure 2.2: Distance from final satisfaction probability across iterations for UAV model with uniform wind.

computation is largely negligible. The solution based on iMDPs appears almost constant with respect to sample size, since we only need to find the maximum and minimum probabilities from amongst the samples this is roughly as expected (in fact we should expect linear scaling, but solving the iMDP takes longer and is independent of the sample size). Finally, for the subgradient and MNE solver using a PNS algorithm the results are more difficult to interpret, but may also be linear, as may be expected.

In fig. 2.4, we consider how the size of the MDP affects the runtimes of our various algorithms. The algorithms which require calculating a reward matrix scale approximately exponentially in the MDP size, and quickly start reaching computation limits. The other algorithms (ignoring some spurious initial behaviour), seem to scale roughly linearly with MDP size. For our subgradient algorithm this is as expected, since for each iteration we update every state-action pair individually. For the other algorithms, this is likely related to the model checker used to solve the models.

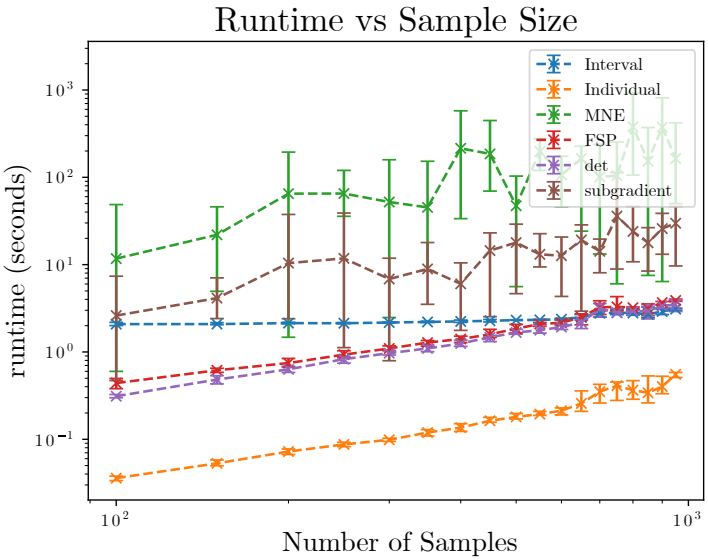


Figure 2.3: Runtime against number of samples.

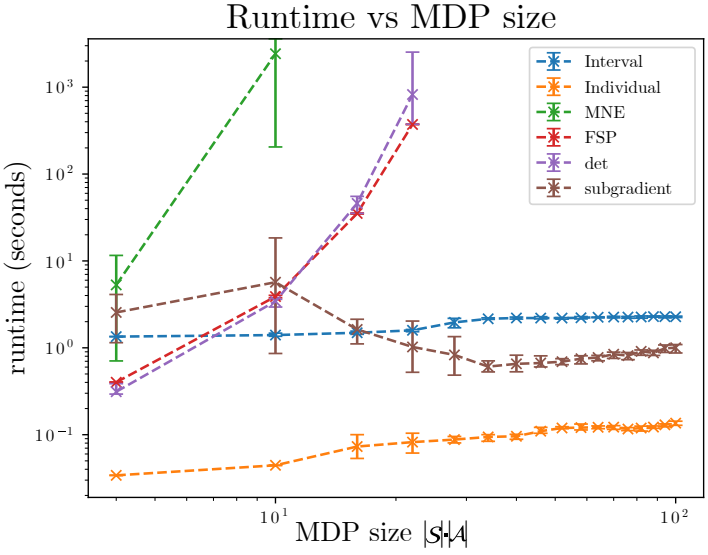


Figure 2.4: Runtime against size of MDP.

	Sat. Prob. $\lambda^\star$	Risk U.B. $\bar{\epsilon}$	Time (s)	Emp. Risk $\tilde{\epsilon}$	Emp. (no pars)
[13]	.338	.056	.07	.018	.115
iMDP Solver	.329	.155	1.39	.015	.107
MNE Algorithm	.333	.109	4.51	.043	.121
Subgradient Algorithm	.332	.285	26.12	.022	.115
Deterministic MaxMin Policy	.329	.105	0.28	.012	.105

Table 2.3: Results on Toy Model, with Deterministic Policies.

2.5.5 Results for Deterministic Policies

As is the case for our MNE algorithm, finding deterministic policies requires iterating through all possible deterministic policies. As such, we only have experimental results for our small toy model, we recall the results for our algorithms on this model, and provide results for the deterministic policy in table 2.3.

2.6 Concluding Remarks and Future Directions

We presented a novel method for learning a single robust policy for upMDPs, providing PAC guarantees on satisfaction of PCTL formulae. We have considered several policy classes, and provided guarantees for each. Our experiments demonstrate the efficacy of our methods on a number of benchmarks, and provide comparisons to previous methods.

One avenue for future work is discarding constraints [28]. Relatedly, improving the runtimes of our algorithms (perhaps at the cost of guarantees) is of interest. Finally, we leave a full technical analysis of the convergence of our subgradient method to later work. In particular, the subgradient we employ is not a true subgradient due to the approximation of $\eta^{u^\star}_{\pi^B}(s)$ and $Q^{u^\star}_{\pi^B}(s,a)$ being independent of π^B (which is not the case for trajectories which revisit a given state). We theorise that this should still constitute a descent direction, since improving the policy for the current state, assuming the future policy is fixed, should lead to at least the same increase if that

policy is employed upon revisiting the state. This analysis is made more complex by the non-convexity of the loss landscape which may only offer nice qualities in a subset of models. However, the subgradient algorithm is well-studied in the convex case and our empirical results demonstrate the algorithm does converge. Hence, an investigation of the conditions under which it does converge (and indeed if it can be shown to do so in the general case) are of significant interest.

3 Continuous State Verification for Discrete Time Systems

3.1 Introduction

Dynamical systems offer a rich class of models for describing the behaviour of diverse, complex systems [61]. It is often of importance that these systems meet certain properties, for example, stability, safety or reachability requirements [46, 65, 85, 96]. Verifying the satisfaction of these properties, also termed as specifications, is a challenging, but important, problem.

One research direction involves discretising the state space [5, 15, 104], in order to construct a finite model with guarantees generated using probabilistic or statistical model checkers [13, 101].

These finite model techniques often scale very poorly with the dimension of the problem, as this discretisation suffers from the "curse of dimensionality", with the number of finite states growing exponentially in the dimension of the continuous space. By avoiding this discretisation, we do not suffer from this problem, and hence (in general) achieve better scaling to high-dimensional systems. However, the computational complexity of our certificate-based approach can vary significantly depending on the behaviour of the system, and the property we wish to verify, whereas finite model techniques do not suffer from this variation due to the ease of model checking in finite models.

Table 3.1: Classification of Certificate Synthesis Approaches
⋆ Theorem 3.3.1, and Algorithms 2 & 3, follow a non-convex scenario approach methodology, that does not require knowledge of the Lipschitz constant of the dynamics, and offer probabilistic verification bounds that do not necessarily scale exponentially in the state space dimension as with [85, 106].

Model-Based Synthesis & Guarantees	Data-Driven Synthesis & Model-Based Guarantees	Data-Driven Synthesis & Guarantees
Sum of Squares Programming [89]	Counter-Example Guided Inductive Synthesis [3, 34, 42, 46]	Neural Network Techniques [9, 113]
MPC + Reachability analysis [4, 100]	Neural Hamilton-Jacobi Reachability Analysis [110, 121]	Convex Scenario Optimisation [85, 106]
SAT-modulo-theory synthesis[6]	Neural Certificates for Safety [66]	Theorem 3.3.1, Algorithms 2 & 3⋆

Depending on the exact finite model employed, and the technique to verify it, different guarantees are available in the finite model setting. In Chapter 2, we considered applying the so-called scenario approach to a finite MDP, thus we achieve guarantees that are of a similar nature to the ones contained in this chapter. Since the strength of these guarantees depends on the "complexity" of the problem the strength is likely to be similar between the two approaches. However, the finite model guarantees suffer in that they are applicable only to the abstraction which may itself only be probably approximately correct with respect to the true system, introducing an additional layer of approximation.

Hence, these two approaches offer different strengths. In general, we hypothesise that the certificate-based methods may provide conceptually simpler, and less conservative, guarantees while also avoiding an exponential increase in computational complexity associated with state-space discretisation. However, for low dimensional models with particularly challenging behaviour, the use of a finite model may provide benefits in computational complexity. Finally, the methods in Chapter 2 deal also with policy synthesis, which is much more difficult for continuous state models, and we therefore postpone controller synthesis to Chapter 5.

An alternative approach to verify properties of dynamical systems that does not require discretising the state space, is through the use of *certificates* [5, 8, 94]. The goal is to determine a function over the system's state space that exhibits certain properties. A well-investigated example of such certificate is that of a Lyapunov function, used to verify that dynamics satisfy some stability property [77]. Here we consider constructing reachability, safety, and reach-while-avoid (RWA) certificates for discrete time systems. Overviews of techniques for certificate learning can be found in [5, 44]; Table 3.1 summarises the related literature, which we discuss below.

One approach to certificate synthesis considers verifying the behaviour of systems assuming that a model of the underlying dynamical system is known. Restricting the class of models to polynomial functions, certificates can be obtained by solving a convex sum-of-squares problem [89]. More generally, synthesis approaches leveraging SAT-modulo-theories can alternatively be leveraged [6, 44]. Availability of a model also allows for the co-design of a controller that meets certain specifications, such as safety and reachability [4, 100], using tools based on Model Predictive Control (MPC), or on reachability analysis.

The inclusion of the model's knowledge in the synthesis procedure restricts the complexity of the models that may be studied. To alleviate this requirement, data-driven techniques, such as counter-example guided inductive synthesis (CEGIS) [3, 34, 42, 46] are able to synthesise certificates for general non-linear systems. This is achieved via the use of neural networks as certificate templates, thus allowing for the approximation of any function within a certain function space [63]. Neural networks have also been explored as a tool in [110, 121] for guaranteeing reachability, and in [66] in the context of safety analysis.

Such approaches involve data-driven synthesis, however, they still require a model of the system when it comes to providing guarantees on the synthesised certificates. Obtaining a model of the system is in general difficult, as it requires domain-specific knowledge. To alleviate these issues, in this work we follow a data-driven route

that is model-free as far both the certificate synthesis and guarantee process is concerned. One way to achieve this involves generating new validation samples [113]. An alternative approach which is also the one most closely related to our formulation, involves probabilistic property satisfaction, using a convex design to allow for the application of results on scenario optimisation [85, 106], and employing neural networks [9, 113]. However, these developments rely on the system dynamics being Lipschitz continuous and for the Lipschitz constant to be known (or a bound on this to be available). Moreover, the probabilistic guarantees provided exhibit an exponential growth with respect to the system dimension. Both issues are not present within our proposed approach.

In this work, we follow a scenario approach paradigm as in [85, 106], however, we exploit some different statistical learning theoretic developments in scenario optimisation. This allows us to remove the requirements for convexity and knowledge of the Lipschitz constant of the dynamics, and establish probabilistic verification bounds that do not necessarily scale exponentially on the state dimension, but their complexity rather depends on the complexity of the underlying property verification task. In particular, we use any parameterised function approximator as certificate template, and learn these parameters using a finite number of system trajectories treated as samples. We formulate the certificate synthesis problem as a (possibly) non-convex optimisation program, that involves minimising an appropriately designed loss function, whose minimum value implies that a given property is satisfied. To minimise that loss function we also design a subgradient descent style procedure [22]. We accompany the synthesised certificate with *probably approximately correct* (PAC) guarantees on its validity, and hence on the probability of of satisfying the underlying property, when it comes to a new system trajectory. It is to be noted that such a procedure does not require using a separate data-set for validation. To establish such PAC guarantees, we make use of bounds on the change of a quantity termed *compression set* (namely, a subset of the data which would re-

turn the same result as the entire set) [30, 50, 79], through recent advancements of the so-called *scenario approach* [26, 27, 29, 32, 53]. In particular, we are inspired by the novel theoretical *pick-to-learn framework* [88], which provides a meta-algorithm for calculating a compression set with favourable properties. Here we extend the scope of the pick-to-learn framework by providing a constructive instance of the general framework to compute the cardinality of compression sets for non-convex optimisation.

Our main contributions can be summarised as follows:

1. We develop a novel methodology for the synthesis of certificates to verify a wide class of properties, namely, reachability, safety and reach-while-avoid specifications, of discrete time dynamical systems. Our results complement the ones in [13] which are concerned with direct property verification and do not construct certificates. Our framework constitutes a first step towards control synthesis exploiting the constructed certificates.

2. Capitalising on developments on scenario optimisation using the notion of compression, we accompany the constructed certificates with probabilistic guarantees on their generalisation properties, namely, on how likely it is that the certificate remains valid when it comes to a new system trajectory. We contrast our approach with [85] and discuss the relative merits of each, both theoretically (Section 3.5) and numerically (Section 3.6).

3. As a byproduct of our certificate construction algorithm, we provide a novel mechanism to compute the *compression set*, which is instrumental in obtaining meaningful probabilistic guarantees. This results in *a posteriori* bounds which, however, scale favorably with respect to the system dimension. This process is novel per se and provides a constructive approach for the general compression set calculation in [88], opening the road for its use in general non-convex optimisation problems.

3.2 Certificates

We consider a family of certificates that allow us to make statements on the behaviour of a dynamical system. Hence, we begin by defining a dynamical system, before considering the certificates and properties they verify.

3.2.1 Discrete Time Dynamical Systems

We consider a bounded state space $X \subset \mathbb{R}^n$, and a dynamical system whose evolution starts at an initial state $x(0) \in X_I$, where $X_I \subseteq X$ denotes the set of all possible initial conditions. From an initial state, we can uncover a finite trajectory, i.e., a sequence of states $\xi = \{x(k)\}_{k=0}^{T}$, where $T \in \mathbb{N}$, by following the dynamics

$$x(k+1) = f(x(k)). \tag{3.1}$$

We define $f\colon X \to \mathbb{R}^n$, and assume it to permit unique solutions, but make no further assumptions on its properties. The set of all possible trajectories $\Xi \subseteq X_I \times X^T$ is then the set of all trajectories starting from the initial set X_I. This set-up considers only deterministic systems, but our methods are applicable to systems with stochastic dynamics - we discuss this in further detail in Section 3.3.1. Our general form of dynamical system allows for verifying systems with controllers "in the loop": for instance, our techniques allow us to verify the behaviour of a system with a predefined control law structure, such as Model Predictive Control [54].

In Section 3.3, we discuss using a finite set of trajectories in order to provide generalisation guarantees for future trajectories. Our techniques only require a finite number of samples, and are *theoretically* not restricted on the properties of such samples (for instance, we may have a finite number of samples each with an infinitely long time horizon). However, we discuss in Section 3.4 how one can synthesise a certificate in practice, and our algorithms are required to store, and perform some calculations on, these trajectories (which is not *practically* possible for T taken to

infinity, or continuous time trajectories). In Chapter 4 we discuss how to modify our algorithm to allow for verification of continuous time trajectories, using a time-discretised approximation for computations.

In order to verify the satisfaction of a property ϕ, we consider the problem of finding a *certificate* as follows.

Definition 3.2.1 (Property Verification & Certificates)**.** *Given a property $\phi(\xi)$, and a function $V\colon \mathbb{R}^n \to \mathbb{R}$, let ψ^s and $\psi^\Delta(\xi)$ be conditions such that, if*

$$\left[\exists V\colon \left(V \models \psi^s \wedge (\forall \xi \in \Xi) V \models \psi^\Delta(\xi)\right)\right] \implies \phi(\xi), \forall \xi \in \Xi,$$

then the property ϕ is verified for all $\xi \in \Xi$. We then say that such a function V is a certificate *for the property encoded by ϕ.*

In words, the implication of Definition 3.2.1 is that if a certificate V satisfies the trajectory-independent conditions in ψ^s, as well as the trajectory-dependent conditions in $\psi^\Delta(\xi)$, for all $\xi \in \Xi$, then the property $\phi(\xi)$ is satisfied for all trajectories $\xi \in \Xi$.

3.2.2 Certificates

We now provide a concrete definition for a number of these properties, and associated certificates (and certificate conditions) that meet the format of Definition 3.2.1. We assume that V is continuous, so that when considering the supremum/infimum of V over a bounded set, this is well-defined.

Property 1 (Reachability)**.** *Consider* (3.1)*, and let $X_G, X_I \subset X$ denote a goal and initial set, respectively. Assume further that X_G is compact and ∂X_G denotes its boundary. If, for all $\xi \in \Xi$,*

$$\phi_{\text{reach}}(\xi) := \exists k \in \{0, \dots, T\}\colon x(k) \in X_G, \tag{3.2}$$

holds, then we say that ϕ_{reach} *encodes a reachability property.* Ξ *denotes the set of trajectories consistent with* (3.1) *and with initial states contained within* X_I*.*

By the definition of ϕ_{reach} it follows that verifying that a system exhibits the reachability property is equivalent to verifying that all trajectories generated from the initial set enter the goal within at most T time steps. To verify this property, we consider a certificate that must satisfy a number of conditions. These conditions are summarised next. Fix $\delta > -\inf_{x\in X_I} V(x) \geq 0$. We then have

$$V(x) \leq 0,\ \forall x \in X_I, \tag{3.3}$$

$$V(x) \geq -\delta,\ \forall x \in \partial X_G, \tag{3.4}$$

$$V(x) > -\delta,\ \forall x \in X \setminus X_G, \tag{3.5}$$

$$V(x) > 0,\ \forall x \in \mathbb{R}^N \setminus X, \tag{3.6}$$

$$\begin{aligned} &V(x(k+1)) - V(x(k)) \\ &< -\frac{1}{T}\Big(\sup_{x\in X_I} V(x) + \delta\Big),\ k = 0, \ldots, k_G - 1, \end{aligned} \tag{3.7}$$

where $k_G := \min\{k \in \{0, \ldots, T\} \colon V(x(k)) \leq -\delta\}$, or $k_G = T$, if there is no such k. Conditions (3.4)-(3.6) allow characterising different parts of the state space by means of specific level sets of V. In particular, we require V to be non-positive within the initial set X_I (3.3) and positive outside the domain (3.6) (to ensure we do not leave the domain, where (3.7) may not hold), while V should be no more negative than a pre-specified level $-\delta < 0$ in the rest of the domain X (3.5), and the sublevel set V less than $-\delta$ should be contained within the goal set X_G (3.4). Conditions (3.4)-(3.5) provide a bound on the value of our function which we must reach within the time horizon.

The condition in (3.7) is a decrease condition (its right-hand side is negative due to the choice of δ), that implies V is decreasing along system trajectories till the first time the goal set is reached (by the definition of the time instance k_G). If T is allowed to tend to infinity (i.e. an infinite time horizon), the difference condition in

(3.7) is reduced to a negativity requirement, as is standard in the literature [46]. To gain some intuition on (3.7), see that if $k_G = T$, its recursive application leads to

$$V(x(T)) < V(x(0)) - T\frac{1}{T}\Big(\sup_{x\in X_I} V(x) + \delta\Big) \leq -\delta, \tag{3.8}$$

where the inequality holds since $V(x(0)) \leq \sup_{x\in X_I} V(x)$. Therefore, if the system starts within X_I, then it reaches the goal set (see (3.4)) in at most T steps.

A graphical representation of these conditions is provided in Figure 3.1. The inner sublevel set (with dashed line) is the set obtained when the certificate value is less than $-\delta$, whilst the outer one is the set obtained when the certificate is less than 0. The decrease condition then means that we do not leave the larger sublevel set without entering the smaller sublevel set.

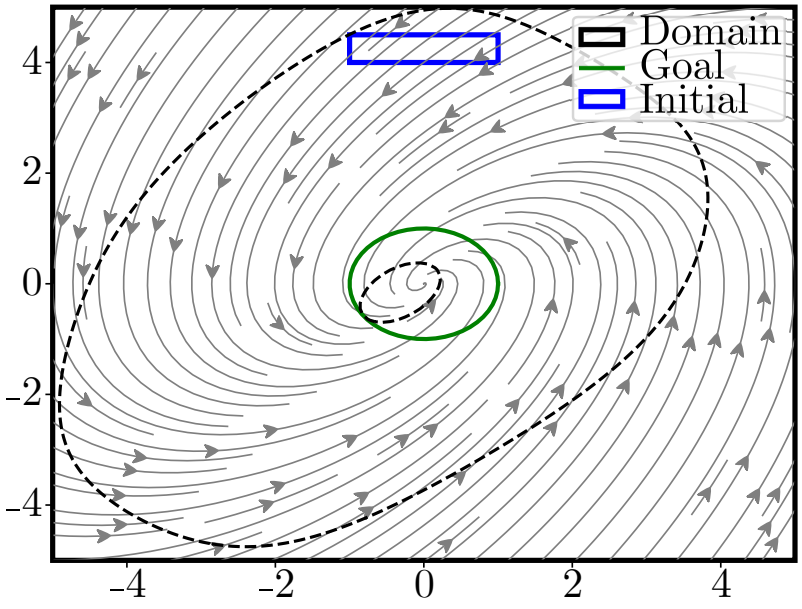


Figure 3.1: Pictorial illustration of the level sets associated with the reach certificate for the system in (3.44).

Now introduce ψ^s_{reach} to encode conditions (3.3)-(3.6), while $\psi^{\Delta}_{\text{reach}}(\xi)$ captures (3.7). Notice that the latter depends on ξ as it is enforced on consecutive states $x(k)$ and $x(k+1)$ along a trajectory.

With this in place, we can now define our first certificate.

Proposition 1 (Reachability Certificate)**.** *A function* $V\colon \mathbb{R}^n \to \mathbb{R}$ *is a reachability*

certificate if

$$V \models \psi^{s}_{\text{reach}} \wedge (\forall \xi \in \Xi) V \models \psi^{\Delta}_{\text{reach}}(\xi). \tag{3.9}$$

In words, Proposition 1 implies that a function V is a reachability certificate if it satisfies (3.3)-(3.6), and (3.7) for all trajectories generated by our dynamics. The proof is based on (3.8); provided formally below.

Proof Fix $\delta > -\sup_{x\in X_I} V(x) \geq 0$, and recall that $k_G = \min\{k \in \{0,\dots,T\} \colon V(x(k)) \leq -\delta\}$. Consider then the difference condition in (3.7), namely,

$$\begin{aligned} &V(x(k+1)) - V(x(k)) \\ &< -\frac{1}{T}\Big(\sup_{x\in X_I} V(x) + \delta\Big),\ k = 0,\dots,k_G-1, \end{aligned} \tag{3.10}$$

By recursive application of this inequality $k \leq k_G$ times,

$$\begin{aligned} V(x(k)) &< V(x(0)) - \frac{k}{T}\Big(\sup_{x\in X_I} V(x) + \delta\Big) \\ &\leq \frac{T-k}{T}\sup_{x\in X_I} V(x) - \frac{k}{T}\delta \leq -\frac{k}{T}\delta \leq 0, \end{aligned} \tag{3.11}$$

where the second inequality holds true since $V(x(0)) \leq \sup_{x\in X_I} V(x)$, as $x(0) \in X_I$. The third one holds true since $\sup_{x\in X_I} V(x) \leq 0$ as by (3.3), $V(x) \leq 0$, for all $x \in X_I$, and $k \leq k_G \leq T$, while the last inequality holds true since $\delta > 0$.

By (3.11) we then have that for all $k \leq k_G$, $V(x(k)) < 0$, which implies that $x(k)$ does not leave X for all $k \leq k_G$ (see (3.5)), while by the definition of k_G, $x(k_G) \in X_G$. Notice that if $k_G = T$, then (3.11) (besides implying that $x(k) \in X$ for all $k \leq T$), also leads to $V(x(T)) \leq -\delta$, which means that $x(T) \in X_G$ after T time steps (see (3.4)), which captures the latest time the goal set is reached.

Therefore, all trajectories that start within X_I reach the goal set X_G in at most T steps, without escaping X till then, thus concluding the proof. ■

We now consider a safety property, which is in some sense dual to reachability.

Property 2 (Safety)**.** *Consider* (3.1)*, and let* $X_I, X_U \subset X$ *with* $X_I \cap X_U = \varnothing$ *denote*

an initial and an unsafe set, respectively. If for all $\xi \in \Xi$,

$$\phi_{\text{safe}}(\xi) := \forall k \in \{0, \dots, T\}, x(k) \notin X_U,$$

holds, then we say that ϕ_{safe} *encodes a safety property.* Ξ *denotes the set of trajectories consistent with* (3.1) *and with initial state contained within* X_I.

By the definition of ϕ_{safe}, it follows that verifying that a system exhibits the safety property is equivalent to checking that all trajectories emanating from the initial set avoid the unsafe set for all time instances, until horizon T. The safety property may be constructed for unbounded X.

We now define relevant sufficient conditions for a certificate to verify this property, namely,

$$V(x) \leq 0, \forall x \in X_I, \tag{3.12}$$

$$V(x) > 0, \forall x \in X_U, \tag{3.13}$$

$$\begin{aligned} &V(x(k+1)) - V(x(k)) \\ &< \frac{1}{T}\Big(\inf_{x \in X_U} V(x) - \sup_{x \in X_I} V(x)\Big),\ k = 0, \dots, T-1. \end{aligned} \tag{3.14}$$

Notice that even if $\inf_{x \in X_U} V(x) - \sup_{x \in X_I} V(x) > 0$, i.e., in the case where the last condition encodes an increase of V along the system trajectories, the system still avoids entering the unsafe set. In particular,

$$\begin{aligned} V(x(T)) &< V(x(0)) + \Big(\inf_{x \in X_U} V(x) - \sup_{x \in X_I} V(x)\Big) \\ &\leq \inf_{x \in X_U} V(x), \end{aligned} \tag{3.15}$$

where the inequality holds since $V(x(0)) \leq \sup_{x \in X_I} V(x)$. Therefore, by (3.13), the resulting inequality implies that even if the system starts at the least negative state within X_I, it will still remain safe. Since we consider finite horizon properties,

this increase allows us to be less conservative compared with a simple negativity condition, which would be recovered if we allow T to tend to infinity.

We denote by ψ^s_{safe} the conjunction of (3.12) and (3.13), and by $\psi^{\Delta}_{\text{safe}}(\xi)$ the property in (3.14). We then have the following safety/barrier certificate.

Proposition 2 (Safety/Barrier Certificate)**.** *A function* $V\colon \mathbb{R}^n \to \mathbb{R}$ *is a safety/barrier certificate if*

$$V \models \psi^s_{\text{safe}} \wedge (\forall \xi \in \Xi) V \models \psi^{\Delta}_{\text{safe}}(\xi). \tag{3.16}$$

Proof Consider the condition in (3.14), namely,

$$\begin{aligned} &V(x(k+1)) - V(x(k)) \\ &< \frac{1}{T}\Big(\inf_{x\in X_U} V(x) - \sup_{x\in X_I} V(x)\Big),\ k = 0,\dots,T-1. \end{aligned} \tag{3.17}$$

By recursive application of this inequality for $k \leq T$ times, we obtain

$$\begin{aligned} V(x(k)) &< V(x(0)) + \frac{k}{T}\Big(\inf_{x\in X_U} V(x) - \sup_{x\in X_I} V(x)\Big) \\ &\leq \frac{T-k}{T}\sup_{x\in X_I} V(x) + \frac{k}{T}\inf_{x\in X_U} V(x) \\ &\leq \frac{k}{T}\inf_{x\in X_U} V(x) \leq \inf_{x\in X_U} V(x). \end{aligned} \tag{3.18}$$

where the second inequality holds true since $V(x(0)) \leq \sup_{x\in X_I} V(x)$, as $x(0) \in X_I$. The third inequality holds true since $\sup_{x\in X_I} V(x) \leq 0$ as by (3.12), $V(x) \leq 0$ for all $x \in X_I$ and $k \leq T$. The last inequality holds true since $\inf_{x\in X_U} V(x) \geq 0$, as by (3.13) $V(x) > 0$ for all $x \in X_U$, and $k \leq T$. We thus have

$$V(x(k)) < \inf_{x\in X_U} V(x),\ k = 1,\dots,T. \tag{3.19}$$

and hence $x(k) \notin X_U, k = 0,\dots,T$ (notice that $x(0) \notin X_U$ holds since $X_I \cap X_U = \varnothing$). The latter implies that all trajectories that start in X_I avoid entering the unsafe set X_U, thus concluding the proof. ■

Combining reachability and safety leads to richer properties. One of these is defined next.

Property 3 (Reach-While-Avoid (RWA))**.** *Consider* (3.1)*, and let* $X_I, X_U, X_G \subset X$ *with* $(X_I \cup X_G) \cap X_U = \varnothing$ *denote an initial set, an unsafe set, and a goal set, respectively. Assume further that* X_G *is compact and denote by* ∂X_G *its boundary. If for all* $\xi \in \Xi$*,*

$$\begin{aligned}\phi_{\text{RWA}}(\xi) := \forall k \in \{0, \dots, T\}, x(k) \notin X_U \cup X^c \\ \wedge\ \exists k \in \{0, \dots, T\}, x(k) \in X_G,\end{aligned}$$

holds, then we say that ϕ_{RWA} *encodes a RWA property.* Ξ *denotes the set of trajectories consistent with* (3.1) *and with initial state contained within* X_I*.*

By the definition of ϕ_{RWA}, it follows that verifying that a system exhibits the RWA property is equivalent to verifying that all trajectories emanating from the initial set X_I avoid entering the unsafe set X_U (and the set complement of the domain X), and also eventually enter the goal set X_G.

The RWA property is derived from the reachability and safety properties, thus the conditions ψ^s_{RWA}, are the conjunction of ψ^s_{reach} and ψ^s_{safe}, and $\psi^{\Delta}_{\text{RWA}}(\xi)$ is given by the conjunction of $\psi^{\Delta}_{\text{reach}}(\xi)$ with the following requirement:

$$\begin{aligned}&V(x(k+1)) - V(x(k)) \\ &< \frac{1}{T}\left(\inf_{x \in X_U} V(x) + \delta\right),\ k = k_G, \dots, T-1,\end{aligned} \tag{3.20}$$

Proposition 3 (RWA Certificate)**.** *A function* $V : \mathbb{R}^n \to \mathbb{R}$ *is a RWA certificate if*

$$V \models \psi^s_{\text{RWA}} \wedge (\forall \xi \in \Xi) V \models \psi^{\Delta}_{\text{RWA}}(\xi). \tag{3.21}$$

Proof Since we must satisy ψ_{reach}, we can conclude that, following Proposition 1,

state trajectories emanating from X_I will reach the goal set X_G in at most T time steps.

By (3.13) we have that $V(x) > 0$, for all $x \in_U$ while by (3.3) we have that $V(x) \leq 0$, for all $x \in X_I$. Therefore, $\sup_{x\in X_I} V(x) \leq 0 \leq \inf_{x\in X_U} V(x)$. At the same time by our choice for δ we have that $\delta > -\sup_{x\in X_I} V(x)$. Combining these, we infer that $\delta > -\inf_{x\in X_U} V(x)$. Thus, (3.7) implies that for all $k = 0, \ldots, k_G - 1$,

$$-\frac{1}{T}\Big(\sup_{x\in X_I} V(x) + \delta\Big) < \frac{1}{T}\Big(\inf_{x\in X_U} V(x) - \sup_{x\in X_I} V(x)\Big). \tag{3.22}$$

Therefore,

$$\begin{aligned} &V(x(k+1)) - V(x(k)) \\ &< \frac{1}{T}\Big(\inf_{x\in X_U} V(x) - \sup_{x\in X_I} V(x)\Big), k = 0, \ldots, k_G - 1. \end{aligned} \tag{3.23}$$

Note that this is identical to the difference condition for our safety property, and hence following the same arguments with the proof of Proposition 2, we can infer that state trajectories emanating from X_I will never pass through the unsafe set X_U until time $k = k_G$.

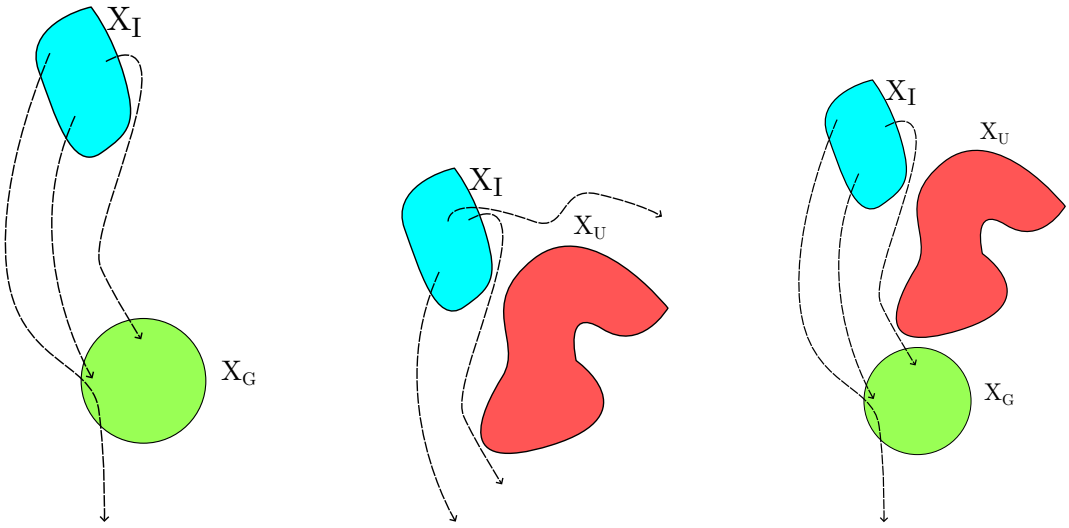
Moreover, by (3.20), we have that

$$\begin{aligned} &V(x(k+1)) - V(x(k)) \\ &< \frac{1}{T}\Big(\inf_{x\in X_U} V(x) + \delta\Big),\ k = k_G, \ldots, T-1. \end{aligned} \tag{3.24}$$

Note that this is also a difference condition identical to that for our safety property, but with δ in place of $\sup_{x\in X_I} V(x)$ (since we know that $V(x(k_G)) \leq -\delta$ by definition of k_G). Hence, we have a safety condition for all trajectories emanating from this sublevel set. We know that trajectories reach this sublevel set, and hence remain safe for $k = k_G, \ldots, T$.

Therefore, we have shown that starting at X_I trajectories reach X_G in at most T time steps, while they never pass through X_U, thus concluding the proof. ■

We provide a graphical representation of the properties in Figure 3.2.



(a) Reachability **(b)** Safety **(c)** RWA

Figure 3.2: Pictorial illustration of (a) reachability, (b) safety, and (c) RWA properties, respectively. Black lines illustrate sample trajectories that satisfy the associated properties.

To synthesise one of these deterministic certificates, we require complete knowledge of the behaviour f of the dynamical system, to allow us to reason about the space of trajectories Ξ. This may be impractical, and we therefore consider learning a certificate in a data-driven manner.

3.3 Data-Driven Certificates

We denote by $(X_I, \mathcal{F}, \mathbb{P})$ a probability space, where $\mathcal{F}$ is a σ-algebra and $\mathbb{P}\colon \mathcal{F} \to [0,1]$ is a probability measure on the set of initial states X_I. Then, the initial state of the system is randomly distributed according to $\mathbb{P}$.

To obtain our sample set, we consider N initial conditions, sampled from $\mathbb{P}$, namely $\{x^i(0)\}_{i=1}^N \sim \mathbb{P}^N$, where we assume that all samples are independent and identically distributed (i.i.d.). Initializing the dynamics from each of these initial

states, we unravel a set of trajectories $\{\xi^i\}_{i=1}^N$. Since there is no stochasticity in the dynamics, we can equivalently say that trajectories (generated from the random initial conditions) are distributed according to the same probabilistic law; hence, with a slight abuse of notation, we write $\xi \sim \mathbb{P}$. In the case of a stochastic dynamical system, the vector field would depend on some additional disturbance vector; our subsequent analysis will remain valid with $\mathbb{P}$ being replaced by the probability distribution that captures both the randomness of the initial state and the distribution of the disturbance. Hence our guarantees are directly applicable to systems with stochastic noise affecting the dynamics, since we only require a distribution over the trajectories to exist (and satisfy the following mild assumption). We focus on deterministic systems with uncertain initial states solely for the improved clarity of presentation. We impose the following mild assumption.

Assumption 3.3.1 (Non-concentrated Mass)**.** *Assume that* $\mathbb{P}\{\xi\} = 0$*, for any* $\xi \in \Xi$*.*

3.3.1 Problem Statement

Since we are now dealing with a sample-based problem, we will be constructing probabilistic certificates and hence probabilistic guarantees on the satisfaction of a given property. We will present our results for a generic property $\phi \in \{\phi_{\text{reach}}, \phi_{\text{safe}}, \phi_{\text{RWA}}\}$ and associated certificate conditions ψ^s, ψ^Δ.

Denote by V_N a certificate of property ϕ, we introduce the subscript N to emphasise that this certificate is constructed on the basis of sampled trajectories $\{\xi^i\}_{i=1}^N$.

Problem 2 (Probabilistic Property Guarantee)**.** *Consider* N *sampled trajectories, and fix a confidence level* $\beta \in (0,1)$*. We seek a property violation level, or "risk",* $\epsilon \in (0,1)$ *such that*

$$\mathbb{P}^N\Big\{\{\xi^i\}_{i=1}^N \in \Xi^N : \\ \mathbb{P}\{\xi \in \Xi \colon \neg\phi(\xi)\} \leq \epsilon\Big\} \geq 1-\beta. \tag{3.25}$$

We achieve this by considering a bound on the probability of a new trajectory satisfying our certificate conditions. Addressing this problem allows us to provide guarantees even if part of the initial set does not satisfy our specification. Our statement is in the realm of probably approximately correct (PAC) learning: the probability of sampling a new trajectory $\xi \sim \mathbb{P}$ failing to satisfy our certificate condition is itself a random quantity depending on the samples $\{\xi^i\}_{i=1}^N$, and encompasses the generalization properties of a learned certificate V_N. It is thus distributed according to the joint probability measure $\mathbb{P}^N$, hence our results hold with some confidence $(1-\beta)$.

Providing a solution to Problem 2 is equivalent to determining an $\epsilon \in (0,1)$, such that with confidence at least $1-\beta$, the probability that V_N does not satisfy the condition $\psi^s \wedge \psi^{\Delta}(\xi)$ for another sampled trajectory $\xi \in \Xi$ is at most equal to that ϵ. As such, with a certain confidence, a certificate V_N *trained* on the basis of N sampled trajectories, will remain a valid certificate with probability at least $1-\epsilon$. Therefore, we can argue that V_N is a *probabilistic* certificate, and hence the property holds (at least) with the same probability.

3.3.2 Probabilistic Guarantees

We now provide a solution to Problem 2. To this end, we refer to a mapping $\mathcal{A}$ such that $V_N = \mathcal{A}(\{\xi^i\}_{i=1}^N)$ as an algorithm that, based on N samples, returns a certificate V_N. Our main result will apply to a generic algorithm that exhibits certain properties outlined as assumptions below. In Section 3.4 we provide a specific synthesis procedure through which $\mathcal{A}$ (and hence the certificate V_N) can be constructed, and show that this algorithm satisfies the considered properties.

The following definition constitutes the backbone of our analysis.

Definition 3.3.1 (Compression Set)**.** *Fix any $\{\xi^i\}_{i=1}^N$, and let $\mathcal{C}_N \subseteq \{\xi^i\}_{i=1}^N$ be a subset of the samples with cardinality $C_N = |\mathcal{C}_N| \leq N$. Define $V_{C_N} = \mathcal{A}(\mathcal{C}_N)$. We*

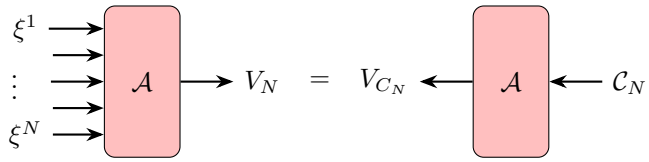


Figure 3.3: Pictorial illustration of the compression set notion of Definition 3.3.1.

say that $\mathcal{C}_N$ *is a compression of* $\{\xi^i\}_{i=1}^N$ *for algorithm* $\mathcal{A}$*, if*

$$V_{C_N} = \mathcal{A}(\mathcal{C}_N) = \mathcal{A}(\{\xi^i\}_{i=1}^N) = V_N. \tag{3.26}$$

Notice the slight abuse of notation, as the argument of $\mathcal{A}$ might be a set of different cardinality; in the following, its domain will always be clear from the context.

Figure 3.3 illustrates Definition 3.3.1 pictorially. It should be noted that compression set cardinalities may be bounded *a priori* [27], that is, without knowledge of the sample-set, or obtained *a posteriori*, and hence depending on the given set $\{\xi^i\}_{i=1}^N$ [29]. We could take a compression set as the entire sample set, resulting in a trivial property violation upper bound of 1. However, it is of benefit to determine a compression set with small (ideally minimal) cardinality, as the smaller C_N is, the smaller risk we can guarantee. In this chapter we are particularly interested in a posteriori results, since we solve a non-convex problem we cannot in general provide a non-trivial bound to the cardinality of the compression set a priori [32]. Therefore, we introduce the subscript N in our notation for $\mathcal{C}_N$ (set) and C_N (corr. cardinality), respectively.

The properties this algorithm $\mathcal{A}$ must satisfy are as follows (adapted from [30]).

Assumption 3.3.2 (Properties of $\mathcal{A}$)**.** *Assume that algorithm* $\mathcal{A}$ *exhibits the following properties:*

1. Preference: *For any pair of multisets* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *of elements of* $\{\xi^i\}_{i=1}^N$*, with* $\mathcal{C}_1 \subseteq \mathcal{C}_2$*, if* $\mathcal{C}_1$ *does not constitute a compression set of* $\mathcal{C}_2$ *for algorithm* $\mathcal{A}$*, then* $\mathcal{C}_1$ *will not constitute a compression set of* $\mathcal{C}_2 \cup \{\xi\}$ *for any* $\xi \in \Xi$.
2. Non-associativity: *Let* $\{\xi^i\}_{i=1}^{N+\bar{N}}$ *for some* $\bar{N} \geq 1$. *If* $\mathcal{C}$ *constitutes a com-*

pression set of $\{\xi_i\}_{i=1}^N \cup \{\xi\}$ *for all* $\xi \in \{\xi^i\}_{i=N+1}^{N+\bar{N}}$ *for algorithm* $\mathcal{A}$*, then* $\mathcal{C}$ *constitutes a compression set of* $\{\xi_i\}_{i=1}^{N+\bar{N}}$ *(up to a measure-zero set).*

If these are satisfied we may use the theorem below to provide probabilistic guarantees on property satisfaction.

Theorem 3.3.1 (Probabilistic Guarantees)**.** *Consider any algorithm* $\mathcal{A}$ *satisfying Assumption 3.3.2 such that* $V_N = \mathcal{A}(\{\xi^i\}_{i=1}^N) \models \bigwedge_{i=0}^N \psi^\Delta(\xi^i) \wedge \psi^s$*, with trajectories* $\{\xi^i\}_{i=1}^N$ *generated in an i.i.d. manner from a distribution satisfying Assumption 3.3.1. Fix* $\beta \in (0,1)$*, and for* $k < N$*, let* $\varepsilon(k,\beta,N)$ *be the (unique) solution to the polynomial equation in the interval* $[k/N, 1]$

$$\frac{\beta}{2N}\sum_{m=k}^{N-1}\frac{\binom{m}{k}}{\binom{N}{k}}(1-\varepsilon)^{m-N} + \frac{\beta}{6N}\sum_{m=N+1}^{4N}\frac{\binom{m}{k}}{\binom{N}{k}}(1-\varepsilon)^{m-N} = 1, \tag{3.27}$$

while for $k = N$ *let* $\varepsilon(N,\beta,N) = 1$*. We then have that*

$$\begin{aligned}&\mathbb{P}^N\big\{\{\xi^i\}_{i=1}^N \in \Xi^N : \\ &\mathbb{P}\{\xi \in \Xi\colon \neg\phi(\xi)\} \leq \varepsilon(C_N,\beta,N)\big\} \geq 1-\beta.\end{aligned} \tag{3.28}$$

Proof Fix $\beta \in (0,1)$, and for each $\{\xi^i\}_{i=1}^N$ let $\mathcal{C}_N$ be a compression set for algorithm $\mathcal{A}$. Moreover, note that letting $V_N = \mathcal{A}(\{\xi^i\}_{i=1}^N)$ we construct a mapping from samples $\{\xi\}_{i=1}^N$ to a decision, namely, V_N, while we impose as an assumption that this mapping satisfies the conditions of Assumption 3.3.2.

We first demonstrate that if the certificate conditions are not satisfied on a new sample, then there will be a change in the compression set when the algorithm is

fed all samples plus the new violating sample, as follows

$$\begin{aligned}
&\{\xi \in \Xi \colon V_N \not\models \psi^s \wedge \psi^{\Delta}(\xi))\} \\
&\qquad \subseteq \{\xi \in \Xi \colon V_N \neq \mathcal{A}(\{\xi\}_{i=1}^N \cup \{\xi\})\} \\
&\qquad = \{\xi \in \Xi \colon \mathcal{A}(\mathcal{C}_N) \neq \mathcal{A}(\mathcal{C}_N^+)\} \\
&\qquad \subseteq \{\xi \in \Xi \colon \mathcal{C}_N \neq \mathcal{C}_N^+\}, \qquad (3.29)
\end{aligned}$$

where $\mathcal{C}_N^+$ denotes a compression set for algorithm $\mathcal{A}$ when fed with $\{\xi\}_{i=1}^N \cup \{\xi\}$. The first inclusion is since for any $\xi \in \Xi$ for which V_N no longer satisfies the certificate condition $(\psi^s \wedge \psi^{\Delta}(\xi))$, we must have that the certificate changes, i.e., $\mathcal{A}(\{\xi^i\}_{i=1}^N \cup \{\xi\})$ (the output of our algorithm when fed with one more sample) is different from V_N. The opposite statement does not always hold, as having a different certificate does not necessarily mean the old one violates an existing condition for a new $\xi \in \Xi$. The equality holds as $V_N = \mathcal{A}(\mathcal{C}_N)$, and $\mathcal{A}(\{\xi\}_{i=1}^N \cup \{\xi\}) = A(\mathcal{C}_N^+)$, by definition of a compression set. Finally, the last inclusion stands since any $\xi \in \Xi$ for which $\mathcal{A}(\mathcal{C}_N) \neq \mathcal{A}(\mathcal{C}_N^+)$, should be such that $\mathcal{C}_N^+ \neq \mathcal{C}_N$. The opposite direction does not always hold, as if $\mathcal{C}_N^+ \supset \mathcal{C}_N$ then we get another compression set of higher cardinality, and hence we may still have $\mathcal{A}(\mathcal{C}_N) = \mathcal{A}(\mathcal{C}_N^+)$.

This derivation establishes the fact that the probability of V_N violating the property when it comes to a new ξ, is bounded by the probability that the compression set changes, i.e., we have that

$$\begin{aligned}
&\mathbb{P}\{\xi \in \Xi \colon V_N \not\models \psi^s \wedge \psi^{\Delta}(\xi))\} \\
&\qquad \leq \mathbb{P}\{\xi \in \Xi \colon \mathcal{C}_N \neq \mathcal{C}_N^+\}. \qquad (3.30)
\end{aligned}$$

We can now make use of [30, Theorem 7], which implies that with confidence at least $1 - \beta$, the probability that for a new $\xi \in \Xi$ the compression set changes, is at

most $\varepsilon(\mathcal{C}_N, \beta, N)$, i.e.,

$$\mathbb{P}\{\xi \in \Xi \colon \mathcal{C}_N^+ \neq \mathcal{C}_N\} \leq \varepsilon(C_N, \beta, N), \tag{3.31}$$

where the expression of$\varepsilon(k, \beta, N)$ for different values of k is given in (3.27). By (3.30) and (3.31), we have that

$$\begin{aligned}&\mathbb{P}^N\Big\{\{\xi^i\}_{i=1}^N \in \Xi^N : \\ &\mathbb{P}\{\xi \in \Xi \colon V_N \not\models \psi^s \wedge \psi^{\Delta}(\xi))\} \leq \varepsilon(C_N, \beta, N)\Big\} \geq 1 - \beta.\end{aligned}$$

By the implication in Definition 3.2.1, (3.28) follows, thus concluding the proof. ■

The following remarks are in order.

1. Notice that Theorem (3.3.1) involves evaluating $\varepsilon(k, \beta, N)$ at $k = C_N$, i.e., at the cardinality of the compression set. Due to the dependency of ε on the samples (via C_N), the proposed probabilistic bound is *a posteriori* as it is adapted to the samples we "see". As a result, this is often less conservative compared to *a priori* counterparts.

2. For cases where algorithm $\mathcal{A}$ takes the form of an optimization program that is convex with respect to the parameter vector, determining non-trivial bounds on the cardinality of compression sets is possible [27, 79], as this is related to the notion of support constraints in convex analysis. However, determining compression sets of low cardinality (necessary for small risk bounds) becomes a non-trivial task if $\mathcal{A}$ involves a non-convex optimization program and/or is iterative (as Algorithm 2). This is since, in a non-convex setting, samples that give rise to inactive constraints may still belong to a compression set, as they may affect the optimal parameter implicitly.

3. An alternative procedure is to use sampled trajectories and to check directly whether a property is satisfied for them (by checking the property definition,

rather than using the associated certificate's conditions). This is a valid alternative but has the drawback of not providing a certificate V_N, simply providing an answer as far as the property satisfaction is concerned. This direction is pursued in [13]; we review this result and compare with our approach in Section 3.5.1. Note that having a certificate is interesting per se, and opens the road for control synthesis, which we aim to pursue in future work.

3.4 Certificate Synthesis

In this section, we propose mechanisms to synthesise a certificate from sampled trajectories, thus offering a constructive approach for algorithm $\mathcal{A}$ in Theorem 3.3.1.

We treat a certificate as an appropriately parameterised "template" (e.g., neural network), and denote the parameter vector by θ. We then have that our certificate V_N depends on θ, which is a vector we seek to identify to instantiate our certificate. For the results of this section, we simply write V_θ and drop the dependency on N to ease notation.

3.4.1 Certificate and Compression Set Computation

We provide an algorithm that seeks to determine an optimal certificate parameterization $\theta^\star$, resulting in a certificate $V_{\theta^\star}$. To this end, for a $\xi \in \Xi$ and parameter vector θ, let

$$L(\theta, \xi) = l^\Delta(\theta, \xi) + l^s(\theta), \tag{3.32}$$

represent an associated loss function consisting of a sample-dependent loss l^Δ, and a sample-independent loss l^s. Without loss of generality, we assume that we can drive the sample-independent loss to be zero (see further discussions later). We impose the next mild assumption, needed to prove termination of our algorithm.

Assumption 3.4.1 (Minimisers' Existence)**.** *For any $\{\xi\}_{i=1}^N$, and any non-empty $\mathcal{D} \subseteq \{\xi\}_{i=1}^N$, the set of minimisers of $\max_{\xi \in \mathcal{D}} L(\theta, \xi)$, is non-empty.*

Algorithm 2 Certificate Synthesis and Compression Set Computation

function $\mathcal{A}(\theta, \mathcal{D})$
  Set $k \leftarrow 0$ ▷ Initialise iteration index
  Set $\mathcal{C} \leftarrow \varnothing$ ▷ Initialise compression set
  Fix $L_1 < L_0$ with $|L_1 - L_0| > \eta$ ▷ η is any fixed tolerance
  while $l^s(\theta) > 0$ **do** ▷ While sample-independent state loss is non-zero
    $g \leftarrow \nabla_\theta l^s(\theta)$ ▷ Gradient of loss function
    $\theta \leftarrow \theta - \alpha g$ ▷ Step in the direction of sample-independent gradient
  end while
  repeat
    $k \leftarrow k + 1$ ▷ Update iteration index
    $\mathcal{M} \leftarrow \{\tilde{\xi} \in \mathcal{D} \colon L(\theta, \tilde{\xi}) \geq \max_{\tilde{\xi} \in \mathcal{C}} L(\theta, \tilde{\xi})\}$ ▷ Find samples with loss greater than compression set loss
    $\overline{g}_{\mathcal{M}} \leftarrow \{\nabla_\theta L(\theta, \tilde{\xi})\}_{\tilde{\xi} \in \mathcal{M}}$ ▷ Subgradients of loss function for $\tilde{\xi} \in \mathcal{M}$
    $\overline{\xi}_{\mathcal{C}} \in \arg\max_{\tilde{\xi} \in \mathcal{C}} L(\theta, \tilde{\xi})$ ▷ Find a sample with maximum loss from $\mathcal{C}$
    $\overline{g}_{\mathcal{C}} \leftarrow \nabla_\theta L(\theta, \overline{\xi}_{\mathcal{C}})$ ▷ Approximate subgradient of loss function for $\tilde{\xi} = \overline{\xi}_{\mathcal{C}}$
    if $\exists \overline{g} \in \overline{g}_{\mathcal{M}} \colon \langle \overline{g}, \overline{g}_{\mathcal{C}} \rangle \leq 0 \wedge \overline{g} \neq 0$ **then** ▷ If there is a misaligned subgradient (take the maximum if multiple)
      $\theta \leftarrow \theta - \alpha \overline{g}$ ▷ Step in the direction of misaligned subgradient
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{\overline{\xi}\}$ ▷ Update compression set with sample corresponding to $\overline{g}$
    else
      $\theta \leftarrow \theta - \alpha \overline{g}_{\mathcal{C}}$ ▷ Step in the direction of approximate subgradient
    end if
    $L_k \leftarrow L_{k-1}$
    if $\max_{\xi \in \mathcal{C}} L(\theta, \xi) < L_{k-1}$ **then** $L_k \leftarrow \max_{\xi \in \mathcal{C}} L(\theta, \xi), \theta^\star \leftarrow \theta$ ▷ Update "running" loss value
    end if
  until $|L_k - L_{k-1}| \leq \eta$ ▷ Iterate until tolerance is met
  return $\theta, \mathcal{C}_N = \mathcal{C} \cup \arg\max_{\xi \in \mathcal{D}} L(\theta, \xi)$
end function

We aim at approximating a minimiser $\theta^\star$ of the quantity $\max_{\xi\in\mathcal{D}} L(\theta,\xi)$ when $\mathcal{D}=\{\xi\}_{i=1}^N$, which exists due to Assumption 3.4.1. We can then use that minimiser to construct $V_{\theta^\star}$. To achieve this, we employ Algorithm 2. The motivating idea is to perform a subgradient descent step where one is allowed to follow an incorrect gradient as long as it points in the right direction. We explain the main steps of Algorithm 2 with the aid of Figure 3.4, where each sample gives rise to a concave triangular constraint.

Algorithm 2 takes as input some initial (arbitrary) parameter vector θ and a set of samples $\mathcal{D}\subseteq\{\xi\}_{i=1}^N$. First, in steps 6–7, we optimise by means of a subgradient descent regime for the sample-independent loss until this loss is non-positive, which serves as a form of warm starting. Then, we follow the subgradient associated with the worst case sample and add it to the compression set $\mathcal{C}$ (step 16–17, point M_1 in Figure 3.4). The ability of the algorithm to avoid bad local minima (such as that slightly to the right of M_1) is tied to the choice of step size α, whereby a larger step size is more likely to avoid "bad" local minima (but may also pass "good" local minima). One way to balance this trade-off may be to consider a diminishing, but non-summable, step size sequence as is common in subgradient methods. When iterates get to a point like M_2, the subgradient step becomes inexact, as for the same parameter there exists a sample resulting in a higher loss (see asterisk). Such a sample is in $\mathcal{M}$, step 11 of Algorithm 2. However, the algorithm does not "jump" to that point, as the inner-product condition in step 15 of the algorithm is not yet satisfied. Graphically, this is since the point M_2 and the red asterisk are on a side of their respective constraints with the same slope. As such the algorithm performs inexact subgradient descent steps up to point M_3; this is the first instance where the condition in step 15 is satisfied (i.e., there exists another constraint with opposite slope[1]) and hence the algorithm "jumps" to a point with higher loss and subgradient of opposite sign. This procedure is then repeated as shown in the figure, with the

[1]This constraint with opposite slope may be any constraint with loss greater than the loss evaluated on the compression set, not just the maximum one.

red line indicating the iterates' path. The "jumps" serve as an exploration step to investigate the non-convex landscape, while their number (plus two for initial and final worst case sample) corresponds to the cardinality of the returned compression set. We iterate till the loss value meets a given tolerance η (see steps 24 and 22). It is to be understood that if $\mathcal{C}$ is empty (as per initialization) steps 13-14 are not performed.

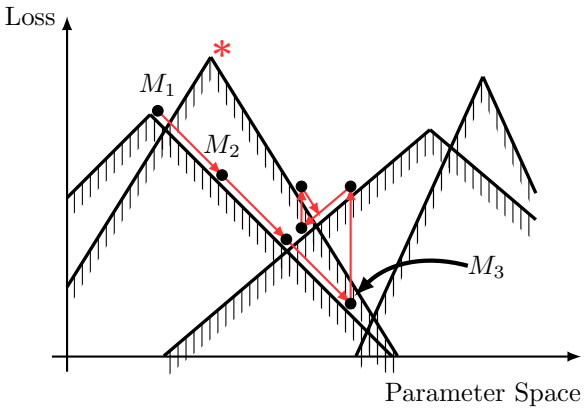


Figure 3.4: Graphical illustration of Algorithm 2.

Overall, Algorithm 2 can be viewed as a specific choice for the mapping $\mathcal{A}$ introduced in Section 3.3 when fed with $\mathcal{D} = \{\xi_i\}_{i=1}^N$, and some initial choice for θ. It follows a subgradient descent scheme with "jumps" that (i) allows minimizing a (possibly) non-convex loss function, and (ii) the mechanism that triggers the "jumps" provides the means to compute a compression set. Such a mechanism serves as an efficient alternative to existing methodologies, as we construct it iteratively. At the same time the constructed compression set is non-trivial as we avoid adding uninformative samples to it, and only add one sample per iteration in the worst case. However, the added sample has a loss higher than that of the compression samples (see step 11), and is also informative in the sense of having a misaligned subgradient that allows for exploration (see step 15). It should be highlighted that the underpinning idea of constructing the compression set by incrementing it by one sample at a time is inspired by the so called pick-to-learn paradigm proposed in [88]. That methodology is general and does not involve a gradient-descent scheme equipped

with our proposed logic. This design is thus novel and serves as a constructive instance of the general methodology of [88].

The main features of Algorithm 2 are summarised in the proposition below.

Proposition 4 (Algorithm 2 Properties)**.** *Consider Assumption 3.3.1, Assumption 3.4.1 and Algorithm 2 with* $\mathcal{D} = \{\xi_i\}_{i=1}^N$ *and a fixed (sample independent) initialization for the parameter* θ*. We then have:*

1. *Algorithm 2 terminates, returning a parameter vector* $\theta^\star$ *and a set* $\mathcal{C}_N$*.*
2. *The set* $\mathcal{C}_N$ *with cardinality* $C_N = |\mathcal{C}_N|$ *forms a compression set for Algorithm 2.*
3. *Algorithm 2 satisfies Assumption 3.3.2.*

Proof

1. By construction, Algorithm 2 creates a non-increasing sequence of iterates $\{L_k\}_{k\geq 0}$ that is bounded below by the global minimum of $\min_{\xi\in\mathcal{D}} L(\theta, \xi)$ which exists and is finite due to Assumption 3.4.1. As such, the sequence $\{L_k\}_{k\geq 0}$ is convergent, which in turn implies that Algorithm 2 terminates.

2. We need to show that the set $\mathcal{C}_N$ is a compression set in the sense of Definition 3.3.1 with $\mathcal{A}$ being Algorithm 2 with $\mathcal{D} = \{\xi_i\}_{i=1}^N$. To see this, we "re-run" Algorithm 2 from the same initial choice of the parameter vector θ but with $\mathcal{C}_N$ in place of $\mathcal{D}$. Notice that exactly the same iterates will be generated, as $\mathcal{C}_N$ contains all samples that have a misaligned subgradient and value greater than the loss evaluated on the running compression set. As a result, the same output will be returned, which by Definition 3.3.1 establishes that $\mathcal{C}_N$ is a compression set.

3. We show that all properties of Assumption 3.3.2 are satisfied by Algorithm 2.

Preference: Consider a fixed (sample independent) initialization of Algorithm 2 in terms of the parameter θ. Consider also any subsets $\mathcal{C}_1, \mathcal{C}_2$ of $\{\xi^i\}_{i=1}^N$ with $\mathcal{C}_1 \subseteq \mathcal{C}_2$.

Suppose that the compression set returned by Algorithm 2 when fed with $\mathcal{C}_2$ is different from $\mathcal{C}_1$. Fix any $\xi \in \Xi$ and consider the set $\mathcal{C}_2 \cup \{\xi\}$. We will show that the compression set returned by Algorithm 2 when fed with $\mathcal{C}_2 \cup \{\xi\}$ is different from $\mathcal{C}_1$ as well.

Case 1: The new sample ξ does not appear as a maximizing sample in step 11 of Algorithm 2, or its subgradient is such that the quantity in step 15 is positive. This implies that step 17 is not performed and the algorithm proceeds directly to step 19. As such, ξ is not added to the compression set returned by Algorithm 2, which remains the same with that returned when the algorithm is fed only by $\{\xi^i\}_{i=1}^N$. However, the latter is not equal to $\mathcal{C}_1$, thus establishing the claim.

Case 2: The new sample ξ appears as a maximizing sample in step 11 of Algorithm 2, and has a subgradient such that the quantity in step 15 is non-positive. As such, step 17 is performed and ξ is added to the compression returned by Algorithm 2. If $\xi \notin \mathcal{C}_1$ then the resulting compression set will be different from $\mathcal{C}_1$ as it would contain at least one element that is not $\mathcal{C}_1$, namely, ξ.

If $\xi \in \mathcal{C}_1$ then it must also be in $\mathcal{C}_2$ as $\mathcal{C}_1 \subseteq \mathcal{C}_2$. In that case ξ would appear twice in $\mathcal{C}_2 \cup \{\xi\}$, i.e., the set of samples with which Algorithm 2 is fed has ξ as a repeated sample (notice that this can happen with zero probability due to Assumption 3.3.1).

Once one of these repeated samples is added to the compression set returned by Algorithm 2, then the other will never be added. This is since when this other sample appears as a maximizing one in step 11 then its duplicate will already be in the compression set, and hence the exact and approximate subgradients

in steps 12 and 14 would be identical. As such, the quantity in step 15 would be non-negative (and, by positive-definiteness of the inner product, only zero when both vectors are zero-vectors) and hence step 17 will not be performed, with the duplicate not added to the compression set. As such, one of the repeated ξ's is redundant, which implies that the compression set returned by Algorithm 2 when fed with $\mathcal{C}_2 \cup \{\xi\}$ is the same with the one that would be returned when it is fed with $\mathcal{C}_2$. However, this would imply that if $\mathcal{C}_1$ is the compression returned by Algorithm 2 when fed with of $\mathcal{C}_2 \cup \{\xi\}$, it will also be the compression set for $\mathcal{C}_2$ (as the duplicate ξ would be redundant). However, the starting hypothesis has been that $\mathcal{C}_1$ is not a compression of $\mathcal{C}_2$. As such, it is not possible for $\mathcal{C}_1$ to be a compression set of $\mathcal{C}_2 \cup \{\xi\}$ as well, establishing the claim.

Non-associativity: Consider a fixed (sample independent) initialization of Algorithm 2 in terms of the parameter θ. Let $\{\xi^i\}_{i=1}^{N+\bar{N}}$ for some $\bar{N} \geq 1$. Suppose that $\mathcal{C}$ is returned by Algorithm 2 a compression set of $\{\xi^i\}_{i=1}^{N} \cup \{\xi\}$, for all $\xi \in \{\xi^i\}_{i=N+1}^{N+\bar{N}}$. Therefore, up to a measure zero set we must have that

$$\mathcal{C} \subset \bigcap_{j=N+1}^{\bar{N}} \left(\{\xi^i\}_{i=1}^{N} \cup \{\xi^j\} \right) = \{\xi^i\}_{i=1}^{N}, \tag{3.33}$$

where the inclusion is since $\mathcal{C}$ is assumed to be returned as a compression set by Algorithm 2 when this is fed with any set within the intersection, while the equality is since by Assumption 3.3.1 all samples in $\{\xi^i\}_{i=1}^{N+\bar{N}}$ are distinct up to a measure zero set. This implies that up to a measure zero set $\mathcal{C}$ should be a compression set returned by Algorithm 2 whenever this is fed with $\{\xi^i\}_{i=1}^{N}$ as any additional sample would be redundant.

Fix now any $\xi \in \{\xi^i\}_{i=N+1}^{N+\bar{N}}$, and consider Algorithm 2 with $\mathcal{D} = \{\xi^i\}_{i=1}^{N} \cup \{\xi\}$. The fact that $\mathcal{C}$ is returned as a compression set for $\{\xi^i\}_{i=1}^{N} \cup \{\xi\}$ implies that

whenever ξ is a maximizing sample in step 11 of Algorithm 2, it should give rise to a subgradient such that the quantity in step 10 of the algorithm is positive. This implies that step 19 is performed and hence ξ is not added to $\mathcal{C}$.

Considering Algorithm 2 this time with $\mathcal{D} = \{\xi^i\}_{i=1}^{N+\bar{N}}$, i.e., fed with all samples at once, due to the aforementioned arguments, whenever a $\xi \in \{\xi^i\}_{i=N+1}^{N+\bar{N}}$ is a maximizing sample in step 11, then the algorithm would proceed to step 19, and steps 16–17 will not be executed. As such, no such ξ will be added to $\mathcal{C}$.

Hence, the compression set returned by Algorithm 2 when fed with $\{\xi^i\}_{i=1}^{N+\bar{N}}$ would be the same with the one that would be returned if the algorithm was fed with $\{\xi^i\}_{i=1}^{N}$. By (3.33) this then implies that the returned set should be $\mathcal{C}$ up to a measure zero set. ■

Proposition 4 implies that we can construct a certificate $V_N = V_{\theta^\star}$, while the algorithm that returns this certificate satisfies Assumption 3.3.2 and admits a compression set $\mathcal{C}_N$ with cardinality C_N. As such, Algorithm 2 offers a constructive mechanism to synthesise a certificate, and, if the loss is driven to zero (the assumed minimum value), then all certificate conditions are met and hence the probabilistic guarantees obtained refer to guarantees on the probability of satisfaction of the underlying property. It should also be noted that in in the numerical simulations presented below, we equipped the subgradient descent scheme with a momentum term thus constructing a deterministic version (as the step size is deterministic) of the so called Adam algorithm [68] to boost performance.

3.4.2 Discarding Mechanism

In some cases, the parameter returned by Algorithm 2 may result in a value of the loss function that is considered as undesirable (and as a result the constructed certificate might be far from meeting the desired conditions). To achieve a lower

loss, we make use of a sample-and-discarding procedure [28, 105]. To this end, consider Algorithm 3. At each iteration of this algorithm, the compression set returned by Algorithm 2 (step 5) is discarded from $\mathcal{D}$, and added to a record of the set of removed samples $\mathcal{R}$ (steps 6–7). We repeat the process till the worst case loss $\max_{\xi\in\mathcal{D}} L(\theta,\xi) \geq 0$ becomes zero (its minimum value). This implies that Algorithm 2 is invoked each time with fewer samples as its input, while the set $\mathcal{R}$ progressively increases. The set of samples that are removed across the algorithm's iterations is denoted by $\mathcal{R}_N$, and forms a compression set for Algorithm 3. However, it has higher cardinality compared to the original compression set, implying that improving the loss comes at the price of an increased risk level ε as the cardinality of the compression set increases.

Algorithm 3 Compression Set Update with Discarding

1: Fix $\{\xi^i\}_{i=1}^N$
2: Set $\widetilde{\mathcal{C}} \leftarrow \varnothing$ ▷ Initialise compression set
3: Set $\mathcal{D} \leftarrow \{\xi^i\}_{i=1}^N$ ▷ Initialise "running" samples
4: **while** $\max_{\xi\in\mathcal{D}} L(\theta,\xi) > 0$ **do**
5: $\theta, \mathcal{C} \leftarrow \mathcal{A}(\theta, \mathcal{D})$ ▷ Call Algorithm 2
6: $\mathcal{D} \leftarrow \mathcal{D} \setminus \mathcal{C}$ ▷ Discard compression set $\mathcal{C}$ from $\mathcal{D}$
7: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{C}$ ▷ Store discarded samples
8: **end while**
9: **return** θ, $\mathcal{R}_N = \mathcal{R}$

This algorithm can be thought of as an add-on to Algorithm 2, and in general to the procedure of [88], as it offers the means to trade the size of the compression set to performance. Unlike Algorithm 2 and [88, Algorithm 1] that iteratively increase the samples used for learning, Algorithm 3 gradually decreases the number of samples used as input to $\mathcal{A}$ across iterations.

Proposition 5 (Algorithm 3 Properties)**.** *Consider Assumption 3.3.1, Assumption 3.4.1 and Algorithm 3 with $\mathcal{D} = \{\xi_i\}_{i=1}^N$ and a fixed (sample independent) initialization for the parameter θ. We then have:*

1. *Algorithm 3 converges to a minimum loss value of zero, returning a parameter*

vector $\theta^\star$ and a set $\mathcal{R}_N$.

2. *The set $\mathcal{R}_N$ with cardinality $R_N = |\mathcal{R}_N|$ forms a compression set for Algorithm 3.*

3. *Algorithm 3 satisfies Assumption 3.3.2.*

Proof

1. At every iteration, Algorithm 2, is called with fewer samples, and initialised on the optimal parameter set from the previous iteration. Hence, the loss value is a non-increasing sequence. If all samples are removed, the loss is zero, since we optimise only the sample-independent loss. Hence, the sequence converges to zero (in the worst-case upon removing all samples).

2. Consider Algorithm 3 with $\mathcal{D} = \{\xi_i\}_{i=1}^N$. Denote by $\mathcal{C}_i$ the set returned at step 5 of Algorithm 3, and recall that $\mathcal{C}_i \subseteq \mathcal{D}$ is the compression set returned by Algorithm 2 when this is invoked at that part of the process. Notice then that the set $\mathcal{R}_N$ returned by Algorithm 3 can be expressed as $\mathcal{R}_N = \bigcup_i \mathcal{C}_i$.
   We need to show that $\mathcal{R}_N$ is a compression set in the sense of Definition 3.3.1 with $\mathcal{A}$ being Algorithm 3 with $\mathcal{D} = \{\xi_i\}_{i=1}^N$. To see this, we "re-run" Algorithm 3 from the same initial choice of the parameter vector θ but with $\mathcal{R}_N$ in place of $\mathcal{D}$. At the first iteration, the set returned in step 5 is $\mathcal{C}_1$ (and the parameter returned would be the same with the one that would be obtained if all samples were employed) as this is a compression set for Algorithm 2 invoked at that step with $\mathcal{D} = \mathcal{R}_\mathcal{N}$. As such, in step 6 and 7 we would, respectively, have that $\mathcal{D} = \mathcal{R}_N \setminus \mathcal{C}_1$, and $\mathcal{R} = \mathcal{R}_N$ since $\mathcal{C}_1$ is already in $\mathcal{R}_N$. Proceeding analogously, we have that Algorithm 3 terminates with the set $\mathcal{R}$ remaining intact to $\mathcal{R}_N$ and $\mathcal{D}$ being empty, and $\mathcal{R} = \mathcal{R}_N$. This establishes that $\mathcal{R}_N$ is a compression set for Algorithm 3.

3. Since Algorithm 2 satisfies Assumption 3.3.2, and we simply call this algorithm repeatedly, then Algorithm 3, also inherits these properties and satisfies

Assumption 3.3.2. ■

Since it establishes that Algorithm 3 satisfies Assumption 3.3.2, we have that Algorithm 3 enjoys the guarantees of Theorem 3.3.1 with $\mathcal{R}_N$ in place of $\mathcal{C}_N$.

The cardinality of the compression set does not necessarily increase with the state space dimension, but is rather dependent on the complexity of the problem. For example, a problem where some trajectories approach or even enter the unsafe set presents a more challenging synthesis problem than one where trajectories all move in the opposite direction to the unsafe set, thus we expect the former to have a larger compression set even if the problem is smaller in dimension. This claim is supported numerically by the results of Section 3.6.

3.4.3 Choices of Loss Function

We now provide some choices of the loss function $L(\theta, \xi) = l^{\Delta}(V_\theta, \xi) + l^s(V_\theta)$ so that minimizing that function we obtain a parameter vector $\theta^\star$, and hence also a certificate $V_{\theta^\star}$, which satisfies the conditions of the property under consideration, namely, reachability, safety, or RWA. Note that when calculating subgradients to these functions, which as we will see below are non-convex, we effectively have the so-called Clarke subdifferential [38].

We provide some expressions for l^s and l^{Δ} for the reachability property in Property 1. For the other properties, the loss functions can be defined in an analogous manner. To this end, we define

$$\begin{aligned} l^s(V_\theta) := &\int_{X \setminus X_G} \max\{0, -\delta - V_\theta(x)\} \, \mathrm{d}x \\ &+ \int_{X_I} \max\{0, V_\theta(x)\} \, \mathrm{d}x + \int_{\mathbb{R}^N \setminus X} \max\{0, -V_\theta(x)\} \, \mathrm{d}x. \end{aligned} \tag{3.34}$$

Focusing on the first of these integrals, if $V(x) > -\delta$ then $\max\{0, -\delta - V_\theta(x)\} = 0$, i.e., no loss is incurred, implying satisfaction of (3.4), (3.5). Under a similar reasoning, the other integrals account for (3.3) and (3.6), respectively. For a sufficiently

expressive function approximator, we can find a certificate V which satisfies the state constraints and hence has a sample-independent loss of zero.

In practice, we replace integrals with a summation over points generated deterministically within the relevant domains. These points are generated densely enough across the domain of interest, and hence offer an accurate approximation. This generation may happen through gridding the relevant domain, or sampling according to a fixed synthetic distribution; these samples are considered here to be fixed and they are not related with the ones used to provide probabilistic guarantees. For the last term, we only enforce the positivity condition on the border of the domain X. Thus, we take a deterministically generated discrete set of N_{states} points on each domain $\mathcal{X}_{\overline{G}}$ for points in the domain but outside the goal region, $\mathcal{X}_I$ from the initial set, and $\mathcal{X}_\partial$ for the border of the domain X. Our loss function takes then the form:

$$\begin{aligned}\hat{l^s}(V_\theta) := \frac{1}{|\mathcal{X}_{\overline{G}}|}\sum_{x\in\mathcal{X}_{\overline{G}}}\max\{0,-\delta-V_\theta(x)\} \\ +\frac{1}{|\mathcal{X}_I|}\sum_{x\in\mathcal{X}_I}\max\{0,V_\theta(x)\}+\frac{1}{|\mathcal{X}_\partial|}\sum_{x\in\mathcal{X}_\partial}\max\{0,-V_\theta(x)\}.\end{aligned} \tag{3.35}$$

We define l^Δ by

$$\begin{aligned}l^\Delta(V_\theta,\xi) := \max\Big\{0, \max_{k=0,\dots,k_G-1}\Big(V_\theta(x(k+1))-V_\theta(x(k))\Big) \\ -\frac{1}{T}\Big(\sup_{x\in\mathcal{X}_I}V_\theta(x)+\delta\Big)\Big\}.\end{aligned} \tag{3.36}$$

The value of l^Δ encodes a loss if the condition in (3.7) is violated. If both l^s and l^Δ evaluate to zero for all $\{\xi\}_{i=1}^N$, then we have that

$$l^s(V_\theta)+\max_{i=1,\dots,N}l^\Delta(V_\theta,\xi^i)=0, \tag{3.37}$$

which by Certificate 1 implies that the constructed certificate V_θ is such that

$$V_\theta \models \psi^s_{\text{reach}} \wedge (i = 1, \dots, N) V_\theta \models \psi^{\Delta}_{\text{reach}}(\xi^i). \tag{3.38}$$

Analogous conclusions hold for all other certificates.

3.4.4 Computational Complexity

With this in place we can provide bounds on the performance of Algorithm 2, in the specific case that our function approximator takes the form of a neural network with N_{layers} layers and N_{neurons} per layer. Then the computational complexity of one iteration of the main loop is $\mathcal{O}((T + N_{\text{states}}) \cdot N \cdot N_{\text{layers}} \cdot N^3_{\text{neurons}})$. We can upper bound the number of iterations by $\frac{L_0 - L^\star}{\eta}$, where $L^\star$ is the value of the loss associated with one of the minimisers.

3.5 Comparison with Related Work

3.5.1 Direct Property Evaluation

As is known in the case of Lyapunov stability theory, the existence of a certificate is useful per se, and allows one to translate a property to a scalar function. However, if one is not interested in the construction of a certificate and only in such guarantees, then Theorem 2 in [13] provides an alternative.

Proposition 6 (Theorem 2 in [13]). *Fix $\beta \in (0, 1)$, and for $r = 0, \dots, N-1$, determine $\varepsilon(r, \beta, N)$ such that*

$$\sum_{k=0}^{r} \binom{N}{k} \varepsilon^k (1-\varepsilon)^{N-k} = \frac{\beta}{N}, \tag{3.39}$$

while for $r = N$ let $\varepsilon(N, \beta, N) = 1$. Denote by R_N the number of samples in $\{\xi^i\}_{i=1}^N$

for which $\phi(\xi^i)$ is violated. We then have that

$$\mathbb{P}^N\Big\{\{\xi^i\}_{i=1}^N \in \Xi^N : \\ \mathbb{P}\{\xi \in \Xi\colon \neg\phi(\xi)\} \leq \varepsilon(R_N, \beta, N)\Big\} \geq 1-\beta. \tag{3.40}$$

This is an *a posteriori* result, as R_N can be determined only once the samples are observed. In this case, we have a compression set which is the set of all discarded samples, plus an additional one to support the solution after discarding. Since this additional sample is always present, we incorporate it in the formula in (3.39).

We remark that one could obtain different bounds through alternative statistical techniques, such as Hoeffding's inequality [62] or Chernoff's bound [35]. Since these bounds are of different nature, we do not pursue that avenue further here.

We compare the risk levels ε computed by each approach on a benchmark example in (3.44) under a *safety* specification; general conclusions are case dependent, as both bounds are *a posteriori*. For a fixed β, Figure 3.5 shows the resulting risk levels for varying N across 5 independently sampled sets of trajectories. The difference between the orange curve and the blue one can be interpreted as the price related to certificate generation, as per Theorem 3.3.1. For sufficiently large N, this price is marginal. As the specification is deterministically safe, no discarding is performed for Proposition 6, resulting in a smooth curve without variability. For non-zero R_N we expect variability as R_N will be randomly distributed.

3.5.2 Certificate Synthesis as in [85]

The results in [85] constitute the closest to our work. As no results on reachability and RWA problems were provided in [85], we limit our discussion to the *safety* property. As with our work, a sample-based construction is performed, where samples therein are pairs (state, next-state), as opposed to trajectories as in our work. However, the probabilistic bounds established in [85] are structurally different and of

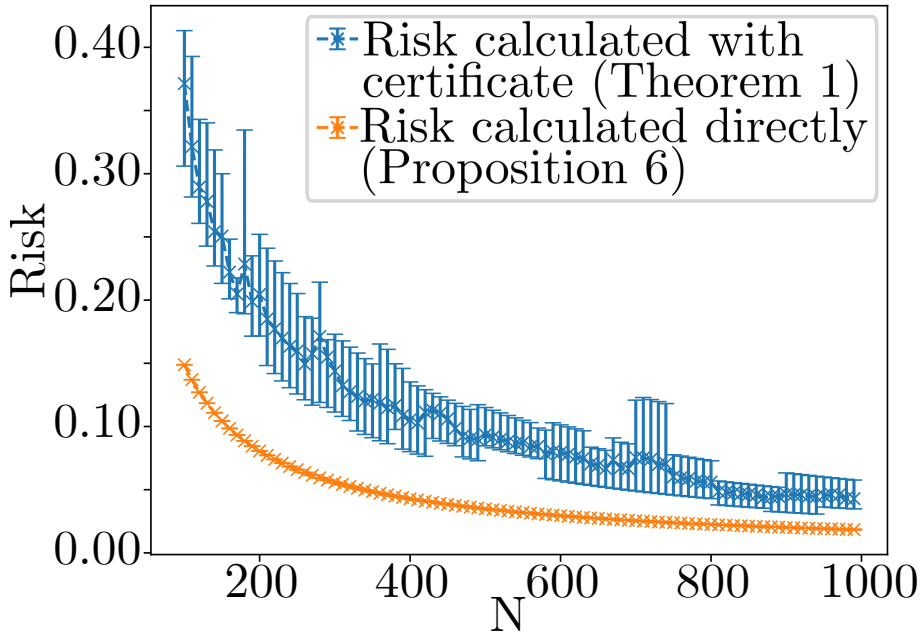


Figure 3.5: Comparison of the bounds in Theorem 3.3.1 and Proposition 6 for direct property evaluation. Median values across the 5 runs are shown with a cross, and ranges are indicated by error bars.

complementary nature to our work: next, we review the main result in [85], adapted to our notation.

Theorem 3.5.1 (Theorem 5.3 in [85])**.** *Consider* (3.1)*, with initial and unsafe sets* $X_I, X_U \subset X \subset \mathbb{R}^n$*, respectively. Consider also* N *samples* $\{x_i, f(x_i)\}_{i=1}^N$ *from* X*, and assume that the loss function in* (3.32) *is Lipschitz continuous with constant* $\mathcal{L}$*. Consider then the problem*

$$
\begin{aligned}
\eta_N^\star \in & \underset{d=(\gamma,\lambda,c,\theta),\eta\in\mathbb{R}}{\arg\min} \quad \eta \\
st. \quad & V_\theta(x) - \gamma \leq \eta, \ \forall x \in X_I \\
& V_\theta(x) - \lambda \geq -\eta, \ \forall x \in X_U \\
& \gamma + cT - \lambda - \mu \leq \eta, \ c \geq 0, \\
& V_\theta(f(x_i)) - V_\theta(x_i) - c \leq \eta, \ i = 1, \dots, N,
\end{aligned} \tag{3.41}
$$

where θ *parameterises* V_θ*, and all other decision variables are scalars leading to level*

sets of V_θ. Let $\kappa(\delta)$ be such that

$$\kappa(\delta) \leq \mathbb{P}\{\mathbb{B}_\delta(x)\}, \forall \delta \in \mathbb{R}_{\geq 0}, \forall x \in X, \tag{3.42}$$

where $\mathbb{B}_\delta(x) \subset X$ is a ball of radius δ, centered at x. Fix $\beta \in (0,1)$ and determine $\epsilon(|d|, \beta, N)$ from (3.39), with $r = d$ and by replacing the right hand-side with β. If $\eta_N^\star \leq \mathcal{L}\kappa^{-1}(\epsilon(|d|, \beta, N)$, we have that

$$\mathbb{P}^N\left\{\{\xi^i\}_{i=1}^N \in \Xi^N : \ \phi_{\text{safe}}(\xi), \ \forall \xi \in \Xi\right\} \geq 1 - \beta. \tag{3.43}$$

The following remarks are in order.

1. The result in [85], capitalizing on the developments of [82], is *a priori*, as opposed to the *a posteriori* assessments of our analysis that are in turn based on [30]. Moreover, [85] offers a guarantee that, with a certain confidence, the safety property is *always* satisfied. This is in contrast to Theorem 3.3.1 where we provide such guarantees in probability (up to a quantifiable risk level ε). However, these "always" guarantees come with potential challenges. In particular, the constraint in (3.42) involves the measure of a "ball" in the uncertainty space. The measure of this ball grows exponentially in the dimension of the uncertainty space (see also Remark 3.9 in [82]), while it depends linearly on the dimension of the decision space $|d|$ (see dependence of ε below (3.42)). This dependence in the results of [85] raises computational challenges to obtain useful bounds: we demonstrate this numerically in Section 3.6 employing one of the examples considered in [85]. On the contrary, Theorem 3.3.1 depends only on the cardinality of the compression set.

2. The result in [85] requires inverting $\kappa(\delta)$, which may not have an analytical form in general. Moreover, it implicitly assumes some knowledge of the distribution to obtain κ, and of the Lipschitz constants of the system dynamics,

which we do not require in our analysis.

3. The results of [85] can be extended to continuous time dynamical systems, which is also possible for our results but outside the scope of this article: we refer to Chapter 4 for extensions to such cases.

3.6 Numerical Results

For all numerical simulations, we considered a confidence level of $\beta = 10^{-5}$, $N = 1000$ samples; our results are averaged across 5 independent repetitions, each with different multi-samples. By sample complexity, we refer to the number of *trajectory samples*, separate to the states used for the sample-independent loss since these samples can be obtained without accessing the system dynamics.

3.6.1 Benchmark Dynamical System

To demonstrate the efficacy of our techniques across all certificates presented, we use the following dynamical system as benchmark, with state vector $x(k) = (x_1(k), x_2(k)) \in \mathbb{R}^2$, namely,

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} x_1(k) - \frac{T_\mathrm{d}}{2} x_2(k) \\ x_2(k) + \frac{T_\mathrm{d}}{2}(x_1(k) - x_2(k)) \end{bmatrix}, \tag{3.44}$$

where $T_\mathrm{d} = 0.1$ and we use time horizon $T = 100$ steps. We used a neural network with 2 hidden layers, and 5 neurons per layer, with sigmoid activation functions thus leading to a parameter vector of size 51. The phase plane plot for these dynamics is in Figure 3.6 alongside different sets related to the definition of reachability, safety and RWA properties are shown.

Surface plots of the reachability, barrier and RWA certificate are shown in Figure 3.7, Figure 3.8 and Figure 3.9, respectively. The zero and $-\delta$-sublevel sets of these

The codebase is available at https://github.com/lukearcus/fossil_scenario

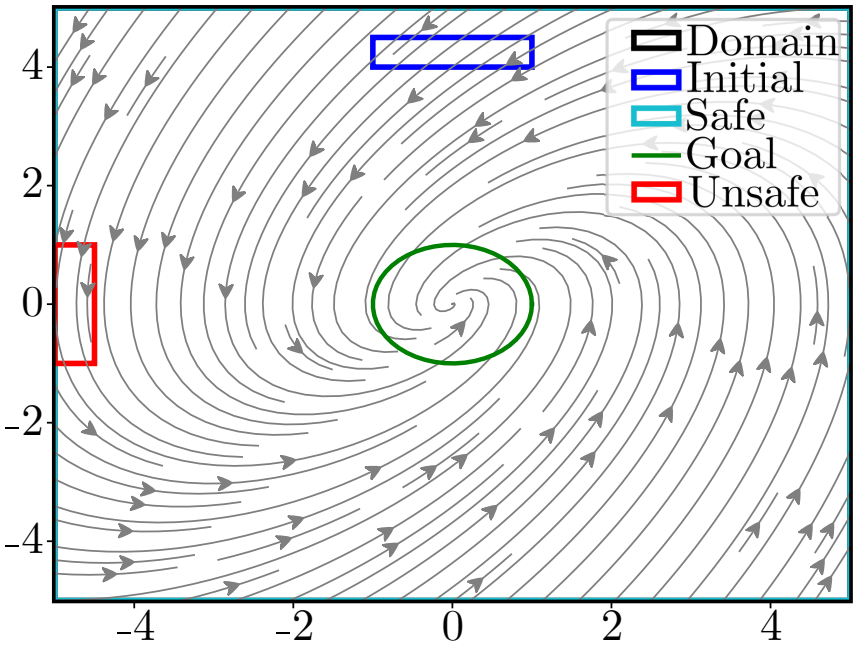


Figure 3.6: Phase plane plot for the dynamical system of (3.44). The different sets shown are related to the sets that appear in the definitions of the reachability, safety and RWA property. For each case, only the relevant sets are considered.

certificates are highlighted with dashed black lines. With reference to Figure 3.7 notice that the zero-sublevel set includes both the initial and the goal set, and no states outside the domain as expected. Similarly, in Figure 3.8 the zero-sublevel set of the barrier function does not pass through the unsafe set, while the zero-sublevel set of the RWA certificate does not pass through the unsafe set, and does not include states outside the domain.

The constructed certificates depend on N samples. By means of Algorithm 3 and Theorem 3.3.1, these certificates are associated with a theoretical risk bound ε (that bounds the probability that the certificate will not meet the conditions of the associated property when it comes to a new sample/trajectory). Table 3.2 shows this risk bound as computed via Theorem 3.3.1. We quantified empirically this property; namely, we generated additional samples and calculated the number of samples for which the computed certificate violated the associated certificate's conditions, or the underlying property. The number violating the certificate conditions (empirical

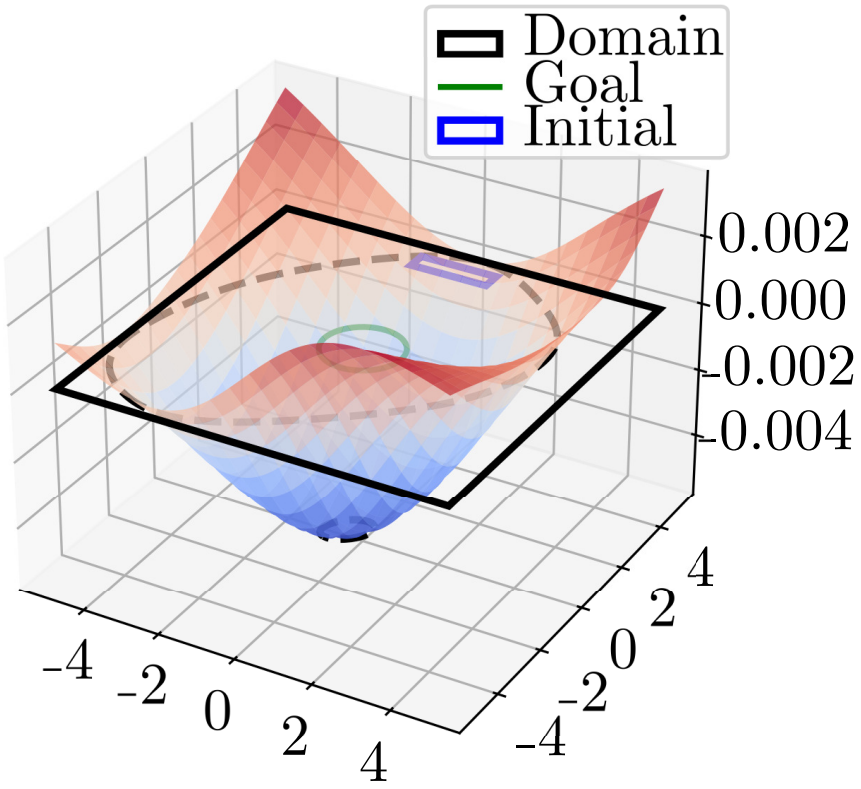


Figure 3.7: Surface plot of the reachability certificate.

certificate risk) is shown in the second column of Table 3.2, and the number violating the property (empirical property risk) is shown in the fifth column. Note that, as expected, the empirical values are lower than the theoretical bounds.

The fourth column of Table 3.2 provides the risk bound ε that would be obtained for direct property violation statements (however, without allowing for certificate construction) as per Proposition 6, this always results in a risk of 0.01825 as no samples are discarded, since the system can be shown to be deterministically safe. Recall that the results in the first column of Table 3.2 bound (implicitly) the probability of property violation, as discussed in the second remark after the proof of Theorem 3.3.1.

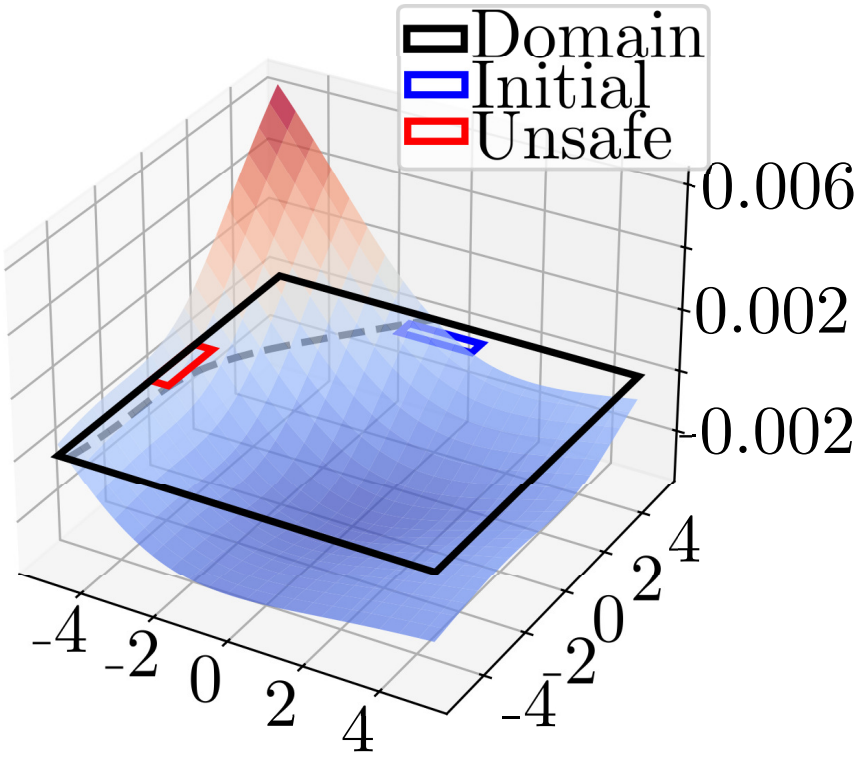


Figure 3.8: Surface plot of the safety/barrier certificate.

3.6.2 Dynamical System of Higher Dimension

We now investigate a dynamical system of higher dimension with a state $x(k) \in \mathbb{R}^8$, governed by

$$\begin{aligned} x_i(k+1) &= x_i(k) + 0.1x_{i+1}(k), \ i = 1 \dots 7, \\ x_8(k+1) &= x_8(k) - 0.1(576x_1(k) + 2400x_2(k) \\ &\quad + 4180x_3(k) + 3980x_4(k) + 2273x_5(k) \\ &\quad + 800x_6(k) + 170x_7(k) + 20x_8(k)). \end{aligned} \tag{3.45}$$

We define $X = [-2.2, 2.2]^8, X_I = [0.9, 1.1]^8, X_U = [-2.2, -1.8]^8$ and use a neural network with 2 hidden layers, and 10 neurons per layer, with sigmoid activation functions thus leading to a parameter vector of size 211. Once again, the entire of the initial set can be shown to be safe, so we aim to generate a guarantee as close to 0 as possible. We employ Algorithm 2 to generate a safety certificate. This required an average of 0.273 seconds, with a standard deviation of 0.018 seconds.

This certificate is computed much faster than those in Table 3.2, which is possible

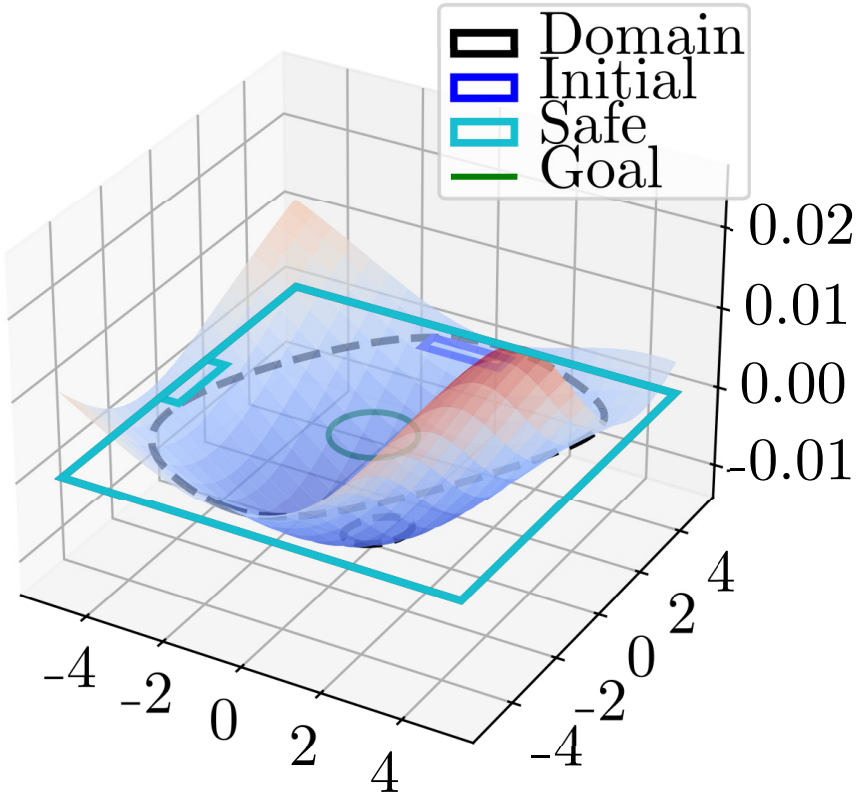


Figure 3.9: Surface plot of the RWA certificate.

since the runtime of our algorithm is primarily constrained by how many samples need to be removed by Algorithm 3 in order to bring the loss to 0. This can be seen as a measure of how "hard" the problem is. In this example, it is likely that the sets are easy to separate whilst still maintaining the difference condition, whereas the system in the previous section required more computation since trajectories move towards the unsafe set, before moving away from it.

Due to the higher-dimensional state space, this certificate is not illustrated pictorially. It is accompanied by a probabilistic certificate $\varepsilon = 0.019$ (standard deviation 0.001) computed by means of Theorem 3.3.1. Using Proposition 6, we find a guarantee of 0.018 (standard deviation 0), after 2.26 seconds (standard deviation 0.05s).

3.6.3 Partially Unsafe Systems

We now consider the problem of safety certificate construction for the system in (3.44) with an enlarged unsafe region (see Figure 3.10). We employ the same neural

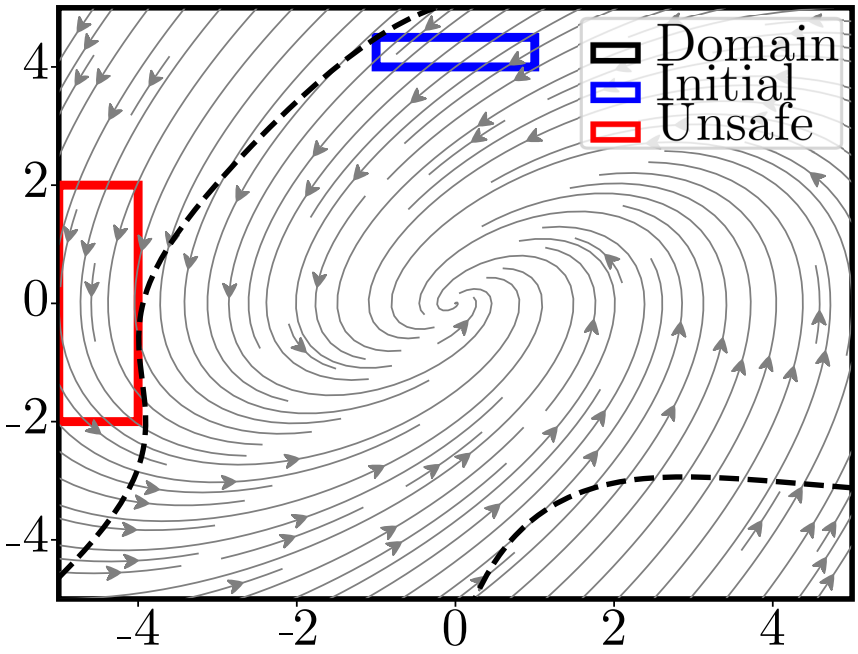


Figure 3.10: Phase plane plot, initial and unsafe set for partially unsafe system.

network as in Section 3.6.1. We refer to this system as partially unsafe, as some sampled trajectories enter the unsafe set. Unlike existing techniques which require either a deterministically safe system [46], or stochastic dynamics [95], we are still able to synthesise a probabilistic barrier certificate. The zero-sublevel set of the constructed safety certificate is shown by a dashed line in both Figures 3.10 and 3.11. Figure 3.11 provides a surface of the constructed certificate, and demonstrates that it separates the initial and the unsafe set. The computation time was 17971 seconds (standard deviation 1414s).

For this certificate, we obtained a theoretical risk bound $\varepsilon = 0.388$ (standard deviation 0.035) by means of Theorem 3.3.1, and an empirical property risk of $\hat{\varepsilon} = 0.011$ (standard deviation 0.002). Proposition 6 gives risk bound 0.042 (standard deviation 0.004) after 3.2 seconds (standard deviation 0.1s).

3.6.4 Comparison with [85]

We extend the comparison of our work with that in [85], which has been reviewed in Section 3.5. To this end, we construct a safety certificate for a two-dimensional

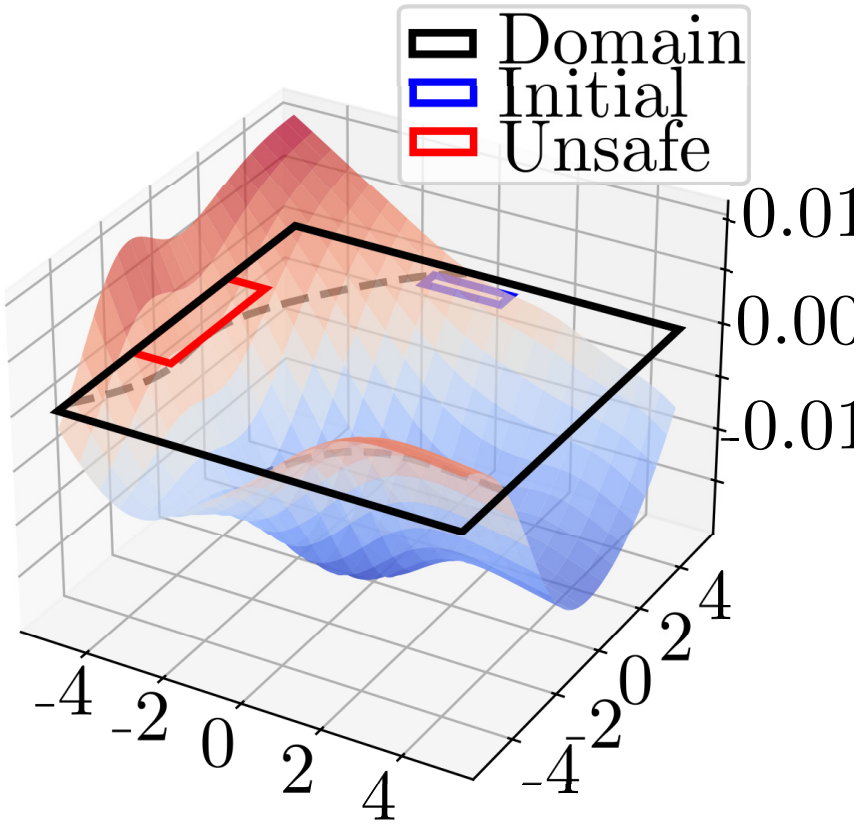


Figure 3.11: Surface plot of the safety/barrier certificate for the partially unsafe system of Figure 3.10.

DC Motor as considered in [85], using a neural network with 1 hidden layer, and 5 neurons per layer, with sigmoid activation functions thus leading to a parameter vector of size 21. We first replicate the methodology of [85], using the Lipschitz constants they provide.

The methodology of [85] required 257149 samples and 307 seconds (standard deviation 44s) of computation time to compute a barrier certificate with confidence at least equal to 0.99. Using 1000 samples and 1.4 seconds of computation time (standard deviation 0.1s), we obtained $\varepsilon = 0.01$ (standard deviation 0), i.e. we can bound safety with a risk of 1%, for the same confidence. It can thus be observed that the numerical computation savings (in terms of number of samples – this might be an expensive task – and computation time) are significant. Figure 3.12 illustrates a phase plane plot and the initial and unsafe sets for this problem. The dotted lines correspond to the sublevel sets constructed in [85] (one lower bounding the

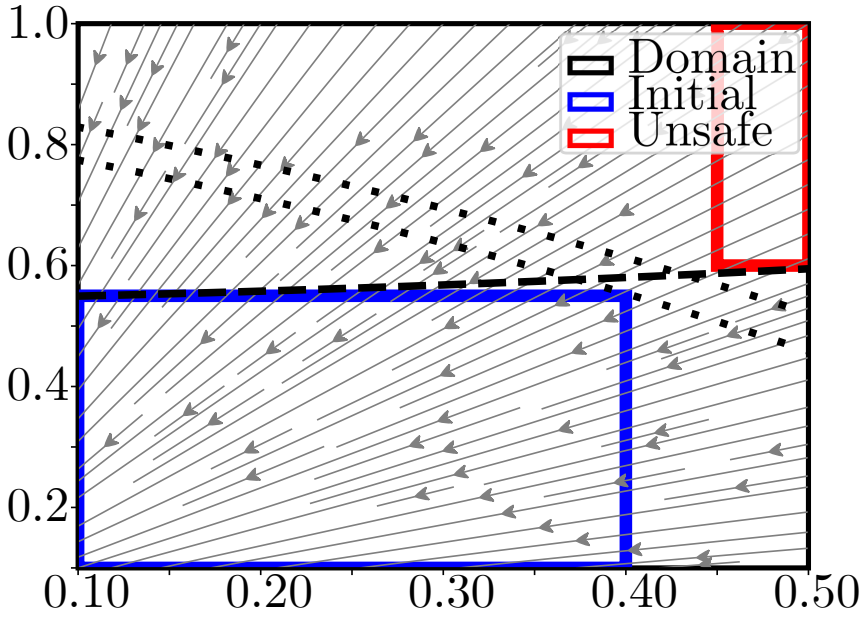


Figure 3.12: Comparison with [85]. The zero-level set of the safety certificate of our approach is dashed; level sets that separate the initial and unsafe sets (i.e. γ- and λ- level sets) from [85] are dotted.

unsafe set, the other upper bounding the initial set). The dashed line depicts the zero-sublevel set of the certificate constructed by our approach.

We also performed a comparison on the following four-dimensional system, a discretised version of a model taken from [46], with the required Lipschitz constants estimated using the technique in [120].

$$\begin{aligned} x_1(k+1) &= x_1(k) + 0.1\left(\frac{x_1(k)x_2(k)}{5} - \frac{x_3(k)x_4(k)}{2}\right), \\ x_2(k+1) &= x_2(k) + 0.1\cos(x_4(k)), \\ x_3(k+1) &= x_3(k) + 0.001\sqrt{|x_1(k)|}, \\ x_4(k+1) &= x_4(k) + 0.1\left(-x_1(k) - x_2(k)^2 + \sin(x_4(k))\right). \end{aligned} \tag{3.46}$$

Due to the reasons outlined in Section 3.5, the approach of [85] with 10^{19} samples results in a confidence of at least 10^{-30}, which is not practically useful. In contrast, with our techniques with 1000 samples we obtain a risk level of $\varepsilon = 0.02039$, with confidence at least $1 - 10^{-5}$.

3.7 Conclusions

We have proposed a method for synthesis of certificates for dynamical systems, based only on a finite number of trajectories from the system, in order to verify a number of core temporally extended specifications. These certificates allow providing assertions on the satisfaction of the properties of interest. In order to synthesise a certificate, we considered a novel algorithm for solving a non-convex optimization program where the loss function we seek to minimise encodes different conditions on the certificate to be learned.

As a byproduct of our algorithm, we determine a quantity termed "compression set", which is instrumental in obtaining scalable probabilistic guarantees. Our numerical experiments demonstrate the efficacy of our methods on a number of examples, involving comparison with related methodologies in the literature.

Table 3.2: Probabilistic guarantees for the system in (3.44). Standard deviations are shown in parentheses alongside means.

	Certificate Risk Bound ε in Theorem 3.3.1	Empirical Certificate Risk $\hat{\varepsilon}$	Algorithm 3 Computation Time (s)	Property Risk Bound ε in Prop. 6	Empirical Property Risk $\hat{\varepsilon}$	Direct Bound Computation Time (s)
Reach Certificate (Proposition 1)	0.026 (0.004)	0 (0)	16597 (9157)	0.018 (0)	0 (0)	2.5 (0.3)
Safety Certificate (Proposition 2)	0.052 (0.017)	0 (0)	7924 (505)	0.018 (0)	0 (0)	7 (1)
RWA Certificate (Proposition 3)	0.037 (0.021)	0 (0)	20120 (15783)	0.018 (0)	0 (0)	7 (1)

4 Continuous State Verification for Continuous Time Systems

4.1 Introduction

Ensuring the safety of continuous time dynamical systems is of critical importance in an increasingly autonomous world [46, 85, 103]. It is often infeasible to model system behaviour precisely, thus making direct use of system data to verify behaviour is of interest [2, 71].

In Chapter 3, we considered certificate synthesis for discrete time systems. However, the guarantees contained in Chapter 3 are no longer valid when it comes to continuous time systems and we now consider extending such techniques to continuous time systems. This extension brings a number of technical challenges due to the inability to store continuous trajectories (these now being infinite dimensional), and the question of derivative access which is likely to be unavailable in general. We resolve these challenges by using a finite number of discretised trajectories of the continuous time system; however, we provide guarantees on the safety of a new unseen continuous time trajectory. We consider slightly different certificate conditions, replacing the discrete time difference condition with a derivative condition, and then using a first-order Euler approximation for this derivative, to avoid requiring derivative access. This derivative condition is then tightened to account for any discretisation error, allowing the guarantees to transfer to the underlying continuous

time system.

In terms of data requirements, the continuous time formulation uses similar data (trajectories), and the number of trajectories is similar. The key difference arises in the need to discretise the trajectories here, which may require a finer discretisation to accurately model the true continuous trajectory, hence needing many more samples per trajectory. This condition is not a consideration in discrete time.

The probabilistic guarantees take the same form and have a similar dependence on the size of the compression set. In this continuous time setting, the tightened constraint may lead to a larger compression set as more iterations are required to satisfy the constraint. Additionally, by tightening this constraint some conservatism is introduced which may make the problem harder to solve for a given data set.

Our modelling assumptions are similar between the two settings, both allowing for general dynamical systems. However the continuous time setting introduces the necessity to relax the assumptions on model knowledge in Chapter 3, and we now consider having an upper bound on the Lipschitz constant of the dynamics. This bound is required to bound the discretisation gap and provide guarantees on unseen continuous time trajectories.

Finally, the constraint tightening requires the loss to be driven to a more negative value which we expect to lead to an increased number of iterations, and therefore an increased computational complexity. This cost is, however, a reasonable trade-off for obtaining guarantees on continuous time systems, which are common in practice, and cannot be handled by the discrete time approach.

Our contributions can be summarised as follows:

- We develop novel theoretical results that extend the results of Chapter 3 (which was limited in scope to discrete time systems) to verify entire continuous trajectories, using discretised approximations for computations;
- We complement the results in [85], following an alternative approach for data-driven certificate synthesis which, however, does not scale exponentially with

respect to the dimension of the underlying state space;

- Our approach is entirely data-driven, avoiding the use of satisfiability modulo theories (SMT) solvers, which are computationally expensive and require a system model.

4.2 Continuous Time Dynamics

For the purposes of this section, we now consider a continuous time dynamical system defined as follows. From an initial state, we unravel a finite trajectory, i.e., a continuous sequence of states $\xi = \{x(t)\}_{t\in[0,T]}$, where $T \in \mathbb{R}_{\geq 0}$ and $x(t)\colon [0,T] \to \mathbb{R}^n$, by following the dynamics

$$\dot{x}(t) = f(x(t)). \tag{4.1}$$

We only require $f\colon X \to \mathbb{R}^n$ to be Lipschitz continuous, thus allowing for existence and uniqueness of solutions. The set of all possible trajectories $\Xi \subseteq X_I \times X^{(0,T]}$ is then the set of all trajectories starting from the initial set X_I.

In Section 4.4, we discuss how to use a finite set of trajectories to build safety certificates, and accompany them with generalisation guarantees with respect to their validity for a new, unseen trajectory. We will also consider time-discretised versions of continuous time trajectories. To this end, define the time-discretised trajectory

$$\tilde{\xi} = \{x(t)\}_{t\in\{0,t_1,\ldots,t_M\}} \in \tilde{\Xi} \subseteq X_I \times X^M, \tag{4.2}$$

for M sampled time steps $t_1, \ldots, t_M$.

4.3 Continuous Time Properties and Certificates

We now define a safety property, and certificate, for a continuous time dynamical system. Reachability and Reach-While-Avoid certificates may be defined analogously.

Property 4 (Safety). *Consider* (4.1)*, and let* $X_I, X_U \subset X$ *with* $X_I \cap X_U = \varnothing$ *denote an initial and an unsafe set, respectively. We say that* ϕ *encodes a safety property for a continuous time trajectory* $\xi \in \Xi$ *if,*

$$\phi(\xi) := \forall t \in [0, T], x(t) \notin X_U.$$

By the definition of ϕ, it follows that verifying that a trajectory exhibits the safety property is equivalent to verifying that a trajectory emanating from the initial set avoids the unsafe set for all time instances, until the horizon $T < \infty$.

We now define the relevant criteria for a certificate B to verify a safety property. Assume that B is continuous, so that when considering the supremum/infimum of B over X (already assumed to be bounded) or over some of its subsets, this is well-defined. Consider:

$$B(x) \leq 0, \forall x \in X_I, \tag{4.3}$$

$$B(x) > 0, \forall x \in X_U, \tag{4.4}$$

$$\left.\frac{dB}{dt}\right|_{x\in\xi} < \frac{1}{T}\Big(\inf_{x\in X_U} B(x) - \sup_{x\in X_I} B(x)\Big), \tag{4.5}$$

where $\frac{dB}{dt} = \frac{\partial B}{\partial x} f(x)$, and hence recognise that this depends on the system dynamics $f(x)$.

Assumption 4.3.1 (Lipschitz Derivative). *Assume that* $\frac{dB}{dt}$ *is Lipschitz continuous.*

Note that even if $\inf_{x\in X_U} B(x) - \sup_{x\in X_I} B(x) > 0$, i.e., if the last condition encodes an increase of B along the system trajectories, the system still avoids entering the unsafe set. This is established in the proof of Proposition 7 below.

Denote by ψ^s the conjunction of (4.3) and (4.4), and by $\psi^{\Delta}(\xi)$ the property in (4.5). Notice that the latter depends on ξ as it relates to the derivative along a trajectory. We define a continuous time safety/barrier certificate as follows.

Definition 4.3.1 (Continuous Time Property Verification & Certificates). *Given a*

safety property $\phi(\xi)$, and a function $B\colon \mathbb{R}^n \to \mathbb{R}$, let ψ^s and $\psi^\Delta(\xi)$ be conditions such that, if

$$\exists B\colon B \models \psi^s \wedge (\forall \xi \in \Xi) B \models \psi^\Delta(\xi) \implies \phi(\xi), \forall \xi \in \Xi,$$

then the property ϕ is verified for all $\xi \in \Xi$. We then say that such a function B is a barrier certificate.

In words, the implication of Definition 4.3.1 is that if a barrier certificate B satisfies the conditions in ψ^s, as well as the conditions in $\psi^\Delta(\xi)$, for all $\xi \in \Xi$, then the safety property $\phi(\xi)$ is satisfied for all trajectories $\xi \in \Xi$.

Proposition 7 (Continuous Time Safety/Barrier Certificate)**.** *A function $B\colon \mathbb{R}^n \to \mathbb{R}$ is a safety/barrier certificate if*

$$B \models \psi^s \wedge (\forall \xi \in \Xi) B \models \psi^\Delta(\xi). \tag{4.6}$$

Proof. It suffices to show that satisfaction of (4.5) implies safety. Integrating (4.5) up to $t \leq T$, we obtain

$$\begin{aligned} B(x(t)) &< B(x(0)) + \frac{t}{T}\Big(\inf_{x\in X_U} B(x) - \sup_{x\in X_I} B(x)\Big) \\ &\leq \frac{T-t}{T} \sup_{x\in X_I} B(x) + \frac{t}{T} \inf_{x\in X_U} B(x) \\ &\leq \frac{t}{T} \inf_{x\in X_U} B(x) \leq \inf_{x\in X_U} B(x). \end{aligned} \tag{4.7}$$

where the second inequality holds true since $B(x(0)) \leq \sup_{x\in X_I} B(x)$, as $x(0) \in X_I$. The third inequality holds true since $\sup_{x\in X_I} B(x) \leq 0$ due to (4.3), and the last one is since $t \leq T$. We thus have

$$B(x(t)) < \inf_{x\in X_U} B(x),\ t \in [0, T], \tag{4.8}$$

i.e. the maximum value along a trajectory is less than the infimum over the unsafe region and hence $x(t) \notin X_U, t = [0, T]$ (notice that $x(0) \notin X_U$ holds since $X_I \cap X_U = \varnothing$). The latter implies that all trajectories that start in X_I avoid entering the unsafe set X_U, thus concluding the proof. ■

To synthesise a certificate, we require complete knowledge of the behaviour f of the dynamical system, to allow us to evaluate the derivative $\frac{dB}{dt}$. This may be impractical, and we therefore wish to use the data-driven techniques developed earlier to learn a continuous time certificate.

4.4 Data-Driven Guarantees

As with discrete time systems, we will treat the initial state as random, distributed according to $\mathbb{P}$. Then, initializing the continuous time dynamics from each of these initial states, we unravel a set of continuous time trajectories $\{\xi^i\}_{i=1}^N$, also referred to as a multi-sample.

As before, our guarantees are applicable to systems with stochastic noise affecting the dynamics, since we only require a distribution over the trajectories to exist (and satisfy the following mild assumption). However, as we will see later, we also require an upper bound on the Lipschitz constant of the dynamics, with stochastic dynamics this bound must still be applicable, restricting the class of noise distributions we may be able to consider. Provided this bound is satisfied, our subsequent analysis holds without any modifications.

Remark *We note that one may be tempted to naively apply Theorem 3.3.1 to the time-discretised trajectories. However, this approach is not compatible with continuous time trajectories. In particular, the inner probability refers to the probability of a new time-discretised trajectory $\tilde{\xi}$ being (un)safe. As a result, states along the state trajectory ξ of the continuous time system under study may violate the safety property (chiefly, between sampled states $x(t_i)$ and $x(t_{i+1})$).*

4.4.1 Extension to the Continuous Time Case

In this section we show how to extend Theorem 3.3.1 to offer guarantees on newly sampled *continuous time trajectories*, even though, to allow for practical applications, the barrier certificate we will construct will still be based only on time-discretised ones. To achieve this, we replace (4.5) by a discretised version based on a first-order Euler approximation (hence we do not require access to the true derivative values). Moreover, we tighten it to enforce an increase upper bound condition between sample times. Denote this condition (over discretised trajectories) as $\psi_d^{\Delta}(\tilde{\xi})$, defined by the inequality

$$\begin{aligned} \max_{k=1,\dots,M} \frac{B(x(t_k)) - B(x(t_{k-1}))}{t_k - t_{k-1}} \\ < \frac{1}{T}\Big(\inf_{x\in X_U} B(x) - \sup_{x\in X_I} B(x)\Big) - d, \end{aligned} \tag{4.9}$$

where $d \in \mathbb{R}$ is a tightening parameter. Define by $\mathcal{L}_B$ and $\mathcal{L}_f$ the Lipschitz constants of the certificate derivative $\frac{\partial B_{\theta^\star}}{\partial x}$ and of the dynamics $f(x)$ respectively, by $\mathcal{M}_B, \mathcal{M}_f$ bounds on their norms, i.e. $\sup_x \|\frac{\partial B_{\theta^\star}}{\partial x}\| \leq \mathcal{M}_B$ and $\sup_x \|f(x)\| \leq \mathcal{M}_f$, and for $\bar{t} = \max_{k=1,\dots,M}(t_k - t_{k-1})$, define d as

$$d = \bar{t}\mathcal{M}_f\left(\mathcal{M}_B\mathcal{L}_f + \mathcal{M}_f\mathcal{L}_B\right). \tag{4.10}$$

Theorem 4.4.1 (Continuous Time Guarantees)**.** *Consider the conditions of Theorem 3.3.1, Assumption 4.3.1 and* (4.10)*. Then,*

$$\begin{aligned} &\mathbb{P}^N\big\{\{\tilde{\xi}^i\}_{i=1}^N \in \tilde{\Xi}^N\colon \\ &\mathbb{P}\{\xi \in \Xi\colon B_N \not\models \psi^s \wedge \psi^{\Delta}(\xi))\} \leq \varepsilon(C_N, \beta, N)\big\} \geq 1-\beta. \end{aligned} \tag{4.11}$$

Proof We aim at finding a bound on the discretisation gap $L(\theta,\xi) - L(\theta,\tilde{\xi})$ so that, for sufficiently small loss evaluated on the time-discretised approximations

$L(\theta, \tilde{\xi})$, we also achieve a negative loss on the continuous trajectories $L(\theta, \xi)$.

$$\begin{aligned} L(\theta,\xi) - L(\theta,\tilde{\xi}) &= l^{\Delta}(\theta,\xi) - l^{\Delta}(\theta,\tilde{\xi}) \\ &= \max_{x\in\xi} \left.\frac{dB}{dt}\right|_x - \max_{k=1,\dots,M} \frac{B(x(t_k)) - B(x(t_{k-1}))}{t_k - t_{k-1}}. \end{aligned} \tag{4.12}$$

Replace the first maximisation with one between time instances, and exchange the order of the max operators,

$$\begin{aligned} &\max_{k=1,\dots,M} \max_{t\in[t_{k-1},t_k]} \left.\frac{dB}{dt}\right|_{x(t)} \\ &\qquad - \max_{k=1,\dots,M} \frac{B(x(t_k)) - B(x(t_{k-1}))}{t_k - t_{k-1}}, \end{aligned} \tag{4.13}$$

$$\leq \max_{k=1,\dots,M} \left[\max_{t\in[t_{k-1},t_k]} \left.\frac{dB}{dt}\right|_{x(t)} - \frac{B(x(t_k)) - B(x(t_{k-1}))}{t_k - t_{k-1}} \right]. \tag{4.14}$$

We can now replace the difference term with an integral,

$$\max_{k=1,\dots,M} \frac{\int_{t_{k-1}}^{t_k} \max_{t\in[t_{k-1},t_k]} \left.\frac{dB}{dt}\right|_{x(t)} - \left.\frac{dB}{dt}\right|_{x(\tau)} \, \mathrm{d}\tau}{t_k - t_{k-1}}.$$

Letting $\mathfrak{L} = \mathcal{M}_B\mathcal{L}_f + \mathcal{M}_f\mathcal{L}_B$ (refer to (4.10) for the definition of the various constants), the previous derivations lead to

$$\begin{aligned} &L(\theta,\xi) - L(\theta,\tilde{\xi}) \\ &\leq \max_{k=1,\dots,M} \frac{\int_{t_{k-1}}^{t_k} \|x(\tau) - \max_{t\in[t_{k-1},t_k]} x\| \mathfrak{L} \, \mathrm{d}\tau}{t_k - t_{k-1}} \\ &\leq \max_{k=1,\dots,M} \mathfrak{L} \frac{\int_{t_k}^{t_{k-1}} \mathcal{M}_f (t_k - t_{k-1}) \, \mathrm{d}\tau}{t_k - t_{k-1}}, \qquad (4.15) \\ &= \max_{k=1,\dots,M} \mathfrak{L} \int_{t_k}^{t_{k-1}} \mathcal{M}_f \, \mathrm{d}\tau = \bar{t}\mathfrak{L}\mathcal{M}_f. \qquad (4.16) \end{aligned}$$

where the second inequality is since $\sup_x \|f(x)\| \leq \mathcal{M}_f$, and the last one since $\bar{t} = \max_{k=1,\dots,M}(t_k - t_{k-1})$.

This results then to a discretisation gap as in (4.10). By Theorem 3.3.1, and noticing that violating the conditions with ψ_d^Δ in place of ψ^Δ, is equivalent to $L(\theta^\star, \tilde{\xi}) > -d$, we have

$$\mathbb{P}^N \left\{ \{\tilde{\xi}^i\}_{i=1}^N \colon \mathbb{P}\{\tilde{\xi} \colon L(\theta^\star, \tilde{\xi}) > -d\} \leq \varepsilon(C_N, \beta, N) \right\} \geq 1 - \beta.$$

Since $L(\theta^\star, \xi) \leq L(\theta^\star, \tilde{\xi}) + d$, this then implies that

$$\mathbb{P}^N \left\{ \{\tilde{\xi}^i\}_{i=1}^N \colon \mathbb{P}\{\xi \colon L(\theta^\star, \xi) > 0\} \leq \varepsilon(C_N, \beta, N) \right\} \geq 1 - \beta,$$

thus concluding the proof. ■

4.5 Continuous Time Certificate Synthesis

For the results of this section, we again write B_θ and drop the dependency on N to ease notation. We make use of Algorithm 2 as an inner loop optimisation procedure, replacing trajectories with their time-discretised approximations throughout, and loss function defined in the sequel.

For $\tilde{\xi} \in \tilde{\Xi}$ and parameter vector θ, let

$$L(\theta, \tilde{\xi}) = l^\Delta(\theta, \tilde{\xi}) + l^s(\theta), \tag{4.17}$$

represent an associated loss function comprised of sample-dependent loss l^Δ, and

sample-independent loss l^s

$$\begin{aligned} l^s(\theta) &:= \frac{1}{|\mathcal{X}_I|}\sum_{x\in\mathcal{X}_I}\max\{0, B_\theta(x)\} \\ &\quad + \frac{1}{|\mathcal{X}_U|}\sum_{x\in\mathcal{X}_U}\max\{0, \delta - B_\theta(x)\}. \\ l^\Delta(\theta,\tilde{\xi}) &:= -\frac{1}{T}\Big(\inf_{x\in\mathcal{X}_U} B_\theta(x) - \sup_{x\in\mathcal{X}_I} B_\theta(x)\Big) \\ &\quad + \max_{k=1,\dots,M}\frac{B_\theta(x(t_k)) - B_\theta(x(t_{k-1}))}{t_k - t_{k-1}}, \end{aligned}$$

We again assume these points are generated densely enough across the domain of interest, and hence offer an accurate approximation. Since they do not require access to the dynamics, we consider them separately from $\{\tilde{\xi}^i\}_{i=1}^N$.

To see the choice of the loss functions, consider the first summation in $l^s(\theta)$, and notice that if $B_\theta(x) \leq 0$ then $\max\{0, B_\theta(x)\} = 0$, i.e., no loss is incurred, implying satisfaction of (4.3). Similar reasoning relates the other summation and expression of $l^\Delta(\theta,\tilde{\xi})$ to (4.4) and (4.9), respectively.

The following mild assumption ensures a minimiser of the loss functions exists and hence our algorithm terminates.

Assumption 4.5.1 (Minimisers' Existence)**.** *For any $\{\tilde{\xi}\}_{i=1}^N$, and any nonempty $\mathcal{D} \subseteq \{\tilde{\xi}\}_{i=1}^N$, the set of minimisers of $\max_{\tilde{\xi}\in\mathcal{D}} L(\theta,\tilde{\xi})$, is nonempty.*

To terminate our algorithm we require knowledge of d (which depends on $\theta^\star$). To resolve this, we propose two approaches.

1. At every iteration j calculate d_j using the current best parameters θ_j, terminate if $\max_i \left[L(\theta_j,\tilde{\xi}^i)\right] < -d_j$.
2. Choose a parameter set, and consider the supremum across that set to determine an upper bound on $\mathcal{L}_B, \mathcal{M}_B$. Use these upper bounds to calculate d.

The second option is likely to be conservative but removes the need for calculat-

ing Lipschitz constants in the loop, hence is computationally more efficient. To determine Lipschitz constants for neural networks we refer to [49, 116].

4.5.1 Sample Discarding

Due to the tightening introduced, to minimise the loss function, we would like its value to be lower than or equal to $-d$. However, in some cases, the parameter returned by Algorithm 2 may result in a loss value greater than $-d$, thus preventing us from verifying safety. To ensure the loss is below that threshold, we employ a sampling-and-discarding procedure as in Algorithm 3, with an adjustment to the termination criteria to ensure that we meet the tightened loss criteria. In particular, we continue discarding while the following condition is satisfied

$$(\max_{\tilde{\xi}\in\mathcal{D}} l^{\Delta}(\theta,\tilde{\xi}) > -d) \vee (l^{s}(\theta) > 0), \tag{4.18}$$

since we wish to ensure the sample-dependent loss is sufficiently tightened, but have no such requirements on the sample-independent loss.

4.6 Numerical Results

We consider constructing a safety certificate for the nonlinear, two-dimensional jet engine model as considered in [85],

$$\dot{x}_1(t) = -x_2(t) - \frac{3}{2}x_1^2(t) - \frac{1}{2}x_1^3(t), \quad \dot{x}_2(t) = x_1(t), \tag{4.19}$$

using a time horizon $T = 5$. Figure 4.1 provides a graphical representation of the dynamics, subdomains under study, the 0-level set produced by our certificate, and the level sets calculated by the methods in [85] (one lower bounding the unsafe set, the other upper bounding the initial set). We used 5 independent repetitions

The codebase is available at https://github.com/lukearcus/fossil_scenario

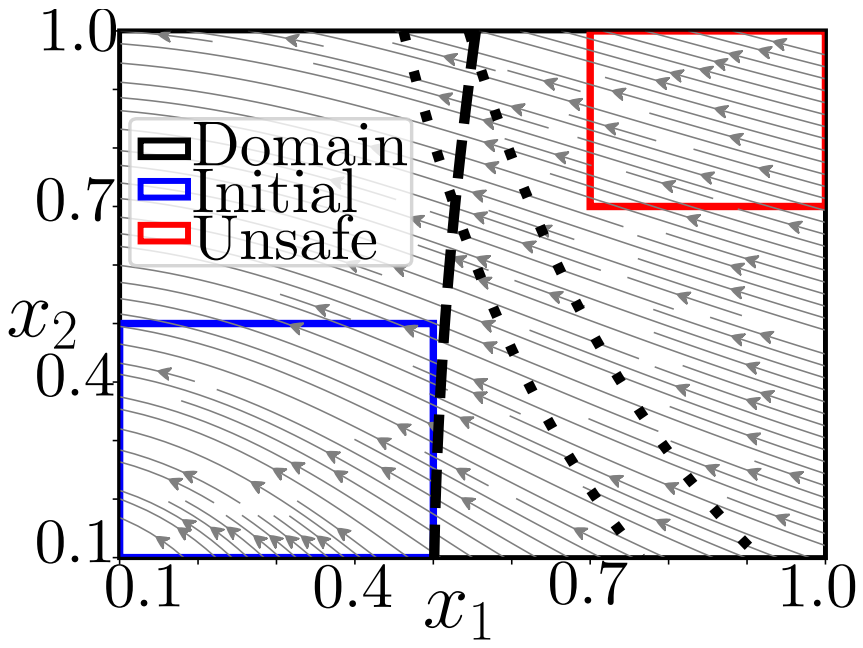


Figure 4.1: Phase plane plot, initial and unsafe set for (4.19). The zero-level set for our certificate is dashed; level sets that bound the initial and unsafe sets in [85] are dotted.

(each with different multi-samples) of $1,000$ sampled trajectories and 367 seconds of computation time (standard deviation 139s), to obtain $\varepsilon = 0.01492$ (standard deviation 0.00140) with confidence 0.99. The methodology of [85] required 257149 state pair samples and 5123 seconds (standard deviation 449s) of computation time to compute a barrier certificate with the same confidence (however, this holds deterministically). We estimate the Lipschitz constants using the methodology in [120], noting that convergence is guaranteed asymptotically. Figure 4.2 contains a 3D plot of the certificate.

Beyond these numerical results, we briefly discuss the theoretical differences between our approach and [85]. The results in [85] offer a guarantee that, with a certain confidence, the safety property is *always* satisfied, in contrast to Theorem 4.4.1 where we provide such guarantees in probability (up to a quantifiable risk level ε). However, these "always" guarantees, although very useful, come with some challenges. Firstly, they are not applicable when part of the initial set is unsafe, whereas we can still bound the probability of a new trajectory being safe even if some of the

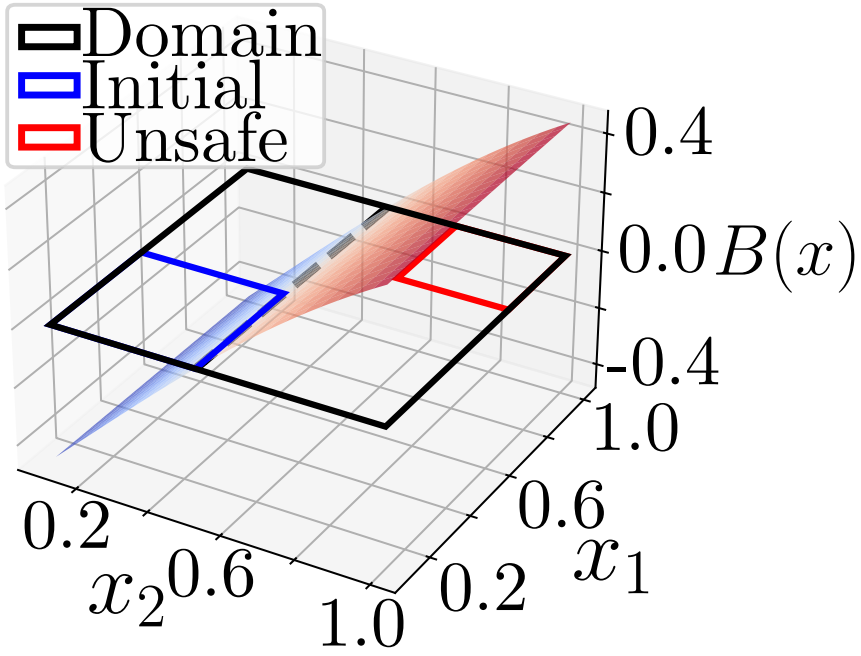


Figure 4.2: Surface plot of the safety/barrier certificate, generated by our techniques, for the system of Figure 4.1.

sampled trajectories were unsafe. Second, they implicitly require knowledge of the underlying probability distribution to instantiate their confidence bound and scale exponentially with respect to the state space dimension.

Related to this last point, we performed a comparison on the following four-dimensional system taken from [46].

$$\begin{aligned}
\dot{x}_1(t) &= x_1(t) + \frac{x_1(t)x_2(t)}{5} - \frac{x_3(t)x_4(t)}{2}, \\
\dot{x}_2(t) &= \cos(x_4(t)), \\
\dot{x}_3(t) &= 0.01\sqrt{|x_1(t)|}, \\
\dot{x}_4(t) &= -x_1(t) - x_2(t)^2 + \sin(x_4(t)),
\end{aligned} \tag{4.20}$$

using $T = 4$. We applied also the approach of [85] which, with 10^{19} samples, returned a confidence 10^{-30}, an uninformative result as it is close to zero. With only 100 samples our techniques obtain a risk level $\varepsilon = 0.21450$ (standard deviation 0.00910), confidence $1 - 10^{-5}$ (close to one).

4.7 Conclusion

We have proposed a method for synthesis of certificates for continuous time dynamical systems, based only on a finite number of trajectories from a system. Our numerical experiments demonstrate the efficacy of our methods on a number of examples, involving comparison with related methodologies in the literature. Current work concentrates towards extending to controlled systems, thus co-designing a controller and a certificate at the same time.

5 Controller Synthesis with Verification for Uncertain Parametric Models

5.1 Introduction

The ability to control a dynamical system towards satisfying a given property is an important task in the modern world [70]. These properties may encompass reachability, safety, or some combination of these [46, 65, 85, 96]. Often, it is not sufficient or preferable to separately design a control law, and verify the system with this control law. Instead, we seek to learn a controller, and a guarantee on the behaviour of this controller, simultaneously.

One technique to achieve this involves taking a continuous state dynamical system, and constructing a finite state abstraction [5, 15, 104]. By constructing a finite state abstraction, the process of controller synthesis, and verification, becomes much simpler [13, 101], and existing probabilistic or statistical model checkers may be employed. This technique is, however, computationally expensive, even for low-dimensional systems. Further, the granularity of this abstraction may introduce an error preventing controller synthesis and property verification.

As in Chapters 3 and 4, we consider a data-driven approach to learning, but, in contrast to these chapters, this chapter considers controller synthesis, extending

significantly on the results which were previously limited to autonomous systems. The model we study in this chapter is similar to that explored in Chapter 2, but now with a continuous state space.

This model choice allows us to acknowledge that most systems exhibit some known structure (e.g. due to physical laws), whilst leaving some parameters of the system (e.g. the weight of a vehicle) as uncertain. Hence, we consider *uncertain parametric* models [18, 25], that is, we assume to have some model of the system, but our model contains a number of parameters that are distributed according to some (potentially unknown) distribution. This allows us to incorporate existing model knowledge without requiring knowledge of all parameters (which may be difficult or even impossible). This framework naturally represents many realistic models [48, 109, 114], for example we may know the equations of motion for a vehicle, but the loading of the vehicle cannot be known a priori.

We consider taking a finite number of samples of the parameters, and use the so-called *scenario approach* [26, 27, 29, 32, 53] to provide guarantees on the generalisability of our obtained controller and certificate to new parameters. We make use of the general algorithm introduced in Chapter 3 for non-convex scenario programs, and extend this to controller synthesis. We make use of any parameterised function approximator as templates for both certificate and controller, and learn these parameters using a finite number of sampled system parameters and initial states. Using these sampled parameters, we can use an intermediate controller to uncover trajectories at every iteration, which may then be used to improve our controller and certificate, in order to achieve this we assume access to a model of the system once the parameters are fixed. Our synthesised controller and certificate are then accompanied by *probably approximately correct* (PAC) guarantees on their validity, and hence on the probability of the controlled system satisfying the underlying property, when it comes to a new system parameter and initial state. It is to be noted that such a procedure does not require using a separate data-set for validation. To

establish such PAC guarantees, we make use of the theoretical advancements in Chapter 3, which provides bounds on the change of a quantity termed *compression set* (namely, a subset of the data which would return the same result as the entire set) [30, 50, 79].

Our main contributions can be summarised as follows:

1. We develop a novel methodology for the synthesis of certificates to verify a wide class of properties, namely, reachability, safety and reach-while-avoid specifications, of uncertain parametric discrete time dynamical systems. Our results complement the ones in Chapter 3 which do not allow for the incorporation of any model knowledge.
2. We provide techniques for the synthesis of controllers, which allow us to "steer" a dynamical system towards satisfying a given property. Our controllers do not depend on the uncertain parameters of the system, and we instead seek a controller that is *robust* to different instantiations of the parameters.
3. Our algorithm is designed such that we can accompany our synthesised controller and certificate with probabilistic guarantees on their generalisation properties, in particular, the probability that our certificate remains valid for the controlled system under a new trajectory generated according to a new system parameter.

5.2 Certificates and Control

5.2.1 Parametrically Uncertain Discrete Time Dynamical Systems

We consider a bounded state space $X \subset \mathbb{R}^n$, and a control-affine dynamical system whose evolution starts at an initial state $x(0) \in X_I$, where $X_I \subseteq X$ denotes the set of all possible initial conditions. The dynamics include a tunable controller $u \colon X \to \mathcal{U}$,

where $\mathcal{U} \subset \mathbb{R}^m$ is a hyperrectangular set of valid control inputs. With this controller, we can uncover a finite trajectory starting from an initial state, i.e., a sequence of states $\xi = \{x(k)\}_{k=0}^{T}$, where $T \in \mathbb{N}_+$, by following the dynamics

$$x(k+1) = f_v(x(k), u(x(k))). \tag{5.1}$$

The evolution is governed by a randomly distributed set of parameters $v \in \mathcal{V}$ (this distribution is formally described in Section 5.3), and we define $f\colon \mathcal{V} \times X \to \mathbb{R}^n$ so that $f_v(\cdot)$ refers to the function parameterised with v. We define $\Xi_{v,u}$ as the set of trajectories for the system parameterised with v, under control law u and, with a slight abuse of notation, we write $\Xi_{v,\mathcal{U}} = \bigcup_{u\in\mathcal{U}} \Xi_{v,u}, \Xi_{\mathcal{V},u} = \bigcup_{v\in\mathcal{V}} \Xi_{v,u}$ and $\Xi = \bigcup_{v\in\mathcal{V}} \Xi_{v,\mathcal{U}} = \bigcup_{u\in\mathcal{U}} \Xi_{\mathcal{V},u}$. This set-up considers only deterministic systems, but our methods are applicable to systems with stochastic dynamics - we discuss this in further detail in Section 5.3. This model allows us to incorporate knowledge about a system (for example through physical laws), whilst allowing for uncertainty in some model parameters. Importantly, we seek to find a controller that is *robust* to different parameter instantiations, as opposed to finding a controller that depends on the parameters.

In Section 5.3, we discuss using a finite set of trajectories in order to provide generalisation guarantees for future trajectories. Our techniques only require a finite number of samples, and are *theoretically* not restricted on the properties of such samples (for instance, we may have a finite number of samples each with an infinitely long time horizon). However, we discuss in Section 5.4 how one can synthesise a certificate in practice, and our algorithms are required to store, and perform some calculations on, these trajectories (which is not *practically* possible for T taken to infinity, or continuous time trajectories).

5.2.2 Control Certificates

We consider the problem of synthesising a controller that verifies the satisfaction of a property ϕ. To this end, we investigate finding a *certificate* and controller as follows.

Definition 5.2.1 (Property Verification & Control Certificates)**.** *Given a property $\phi(\xi)$, and functions $V\colon \mathbb{R}^n \to \mathbb{R}, u \in \mathcal{U}$, let ψ^s and $\psi^{\Delta}(\xi)$ be conditions such that, if*

$$\left[\exists V\colon \left(V \models \psi^s \wedge (\forall \xi \in \Xi) V \models \psi^{\Delta}(\xi)\right)\right] \implies \phi(\xi), \forall \xi \in \Xi_{u,\mathcal{V}},$$

then the property ϕ is verified for all $\xi \in \Xi_{u,\mathcal{V}}$. We then say that such a function V is a control certificate *for the property encoded by ϕ.*

For brevity, we will conduct our analyses with the Reach certificate recalled below, but note that our techniques apply to all certificates outlined in Section 3.2.

We again assume that V is continuous, so that when considering the supremum/infimum of V over a bounded set, this is well-defined.

Property 5 (Reachability)**.** *Consider* (5.1)*, and let $X_G, X_I \subset X$ denote a goal and initial set, respectively. Assume further that X_G is compact and ∂X_G denotes its boundary. If, for all $\xi \in \Xi_{u,\mathcal{V}}$,*

$$\phi_{\text{reach}}(\xi) := \exists k \in \{0, \dots, T\}\colon x(k) \in X_G, \tag{5.2}$$

holds, then we say that ϕ_{reach} encodes a reachability property. $\Xi_{u,\mathcal{V}}$ denotes the set of trajectories consistent with (3.1)*, under controller u, with initial states contained within X_I and with with any possible parameter set.*

By the definition of ϕ_{reach} it follows that verifying that a system exhibits the reachability property is equivalent to verifying that all possible trajectories generated from the initial set enter the goal within at most T time steps. To verify this property, we consider a certificate that must satisfy a number of conditions. These

conditions are summarised next. Fix $\delta > -\sup_{x\in X_I} V(x) \geq 0$. We then have

$$V(x) \leq 0,\ \forall x \in X_I, \tag{5.3}$$

$$V(x) \geq -\delta,\ \forall x \in \partial X_G, \tag{5.4}$$

$$V(x) > -\delta,\ \forall x \in X \setminus X_G, \tag{5.5}$$

$$V(x) > 0,\ \forall x \in \mathbb{R}^N \setminus X, \tag{5.6}$$

$$\begin{aligned} &V(f_v(x(k), u(x(k)))) - V(x(k)) \\ &< -\frac{1}{T}\Big(\sup_{x\in X_I} V(x) + \delta\Big),\ k = 0, \ldots, k_G - 1, \forall v \in \mathcal{V} \end{aligned} \tag{5.7}$$

where $k_G := \min\{k \in \{0, \ldots, T\} \colon V(x(k)) \leq -\delta\}$, or $k_G = T$, if there is no such k. Conditions (5.4)-(5.6) allow characterizing different parts of the state space by means of specific level sets of V. In particular, we require V to be non-positive within the initial set X_I (5.3) and positive outside the domain (5.6) (to ensure we do not leave the domain, where (5.7) may not hold), while V should be no more negative than a pre-specified level $-\delta < 0$ in the rest of the domain X (5.5), and the sublevel set V less than $-\delta$ should be contained within the goal set X_G (5.4). Conditions (5.4)-(5.5) provide a bound on the value of our function which we must reach within the time horizon.

The condition in (5.7) is a decrease condition (its right-hand side is negative due to the choice of δ), that implies V is decreasing along system trajectories till the first time the goal set is reached (by the definition of the time instance k_G). Note that, unlike in previous chapters, this decrease condition now also depends on the choice of controller, hence we are able to, and seek to, find both a controller and certificate that jointly satisfy this decrease condition.

5.3 Data-Driven Control Certificates

We denote by $(X_I, \mathcal{F}_x, \mathbb{P}_x)$ a probability space, where $\mathcal{F}_x$ is a σ-algebra and $\mathbb{P}_x \colon \mathcal{F}_x \to [0,1]$ is a probability measure on the set of initial states X_I. Then, the initial state of the system is randomly distributed according to $\mathbb{P}_x$. Denote also by $(\mathcal{V}, \mathcal{F}_v, \mathbb{P}_v)$ a probability space, similar to above, but now $\mathbb{P}_v$ is a probability measure on the set of possible parameters $\mathcal{V}$. Then, the specific parameter set is randomly distributed according to $\mathbb{P}_v$. Finally, we denote by $(X_I \times \mathcal{V}, \mathcal{F}, \mathbb{P})$, the product probability space over initial states and parameters. Importantly, we do not require that $\mathbb{P}_x$ and $\mathbb{P}_v$ be independent for our analysis.

To obtain our sample set, we consider N initial condition and parameter pairs, sampled from $\mathbb{P}$, namely $\{x^i(0), v^i\}_{i=1}^N \sim \mathbb{P}^N$, where we assume that all samples are independent and identically distributed (i.i.d.). Initializing the dynamics from each of these initial states, with the associated parameter set and with some fixed controller u, we unravel a set of trajectories $\{\xi^i\}_{i=1}^N \in \Xi_{u,\mathcal{V}}$.

We start by unravelling trajectories according to some controller u_0, this may be a controller which takes a uniform random action across states, or may be initialised using some domain knowledge to improve convergence. Since there is no stochasticity in the dynamics, we can equivalently say that, for a fixed controller u, trajectories (generated from the random initial conditions) are distributed according to the same probabilistic law; hence, with a slight abuse of notation, we write $\xi \sim \mathbb{P}_u$. Where we now use the subscript u to denote the dependency of this distribution on the controller. In the case of a stochastic dynamical system, the vector field would depend on some additional disturbance vector; our subsequent analysis will remain valid with $\mathbb{P}_u$ being replaced by the probability distribution that captures both the randomness of the initial state and the distribution of the disturbance. Further discussion on the practical aspects of controller and certificate synthesis for stochastic systems is included in Section 5.4. We impose the following mild assumption.

Assumption 5.3.1 (Non-concentrated Mass)**.** *We assume that* $\mathbb{P}_u\{\xi\} = 0$*, for any* $\xi \in \Xi_{u,\mathcal{V}}$.

5.3.1 Problem Statement

Since we are now dealing with a sample-based problem, we will be constructing probabilistic certificates and hence probabilistic guarantees on the satisfaction of a given property. We will present our results for a generic property $\phi \in \{\phi_{\text{reach}}, \phi_{\text{safe}}, \phi_{\text{RWA}}\}$ and associated certificate conditions ψ^s, ψ^Δ.

Denote by V_N a certificate of property ϕ, and by u_N an associated controller, we introduce the subscript N to emphasise that this certificate is constructed on the basis of sampled trajectories $\{\xi^i\}_{i=1}^N$.

Problem 3 (Probabilistic Property Guarantee)**.** *Consider* N *sampled trajectories, and fix a confidence level* $\beta \in (0,1)$*. We seek to learn a controller* u_N*, and an associated property violation level, or "risk",* $\epsilon \in (0,1)$ *such that*

$$\mathbb{P}_{u_0}^N\left\{\{\xi^i\}_{i=1}^N \in \Xi_{u_0,\mathcal{V}}^N : \mathbb{P}_{u_N}\{\xi \in \Xi_{u_N,\mathcal{V}} \colon \neg\phi(\xi)\} \leq \epsilon\right\} \geq 1-\beta, \tag{5.8}$$

where the initial set of trajectories are generated by some initial controller u_0.

It is important to note that we begin with some arbitrary controller u_0, and ultimately learn a controller u_N (this being a controller that has been optimised to allow the system to satisfy the certificate conditions, details of this optimisation are presented in Section 5.4), we then provide guarantees on the behaviour of the system with this learnt controller applied. We solve this problem by considering a bound on the probability of a new trajectory, generated with the learnt controller, satisfying our certificate conditions. Our statement is in the realm of probably approximately correct (PAC) learning: the probability of sampling a new trajectory $\xi \sim \mathbb{P}_{u_N}$ failing to satisfy our certificate condition is itself a random quantity depending on the samples $\{\xi^i\}_{i=1}^N$, and encompasses the generalisation properties of a learnt

certificate V_N. It is thus distributed according to the joint probability measure $\mathbb{P}_{u_0}^N$, hence our results hold with some confidence $(1-\beta)$.

Providing a solution to Problem 3 is equivalent to determining an $\epsilon \in (0,1)$, such that with confidence at least $1-\beta$, the probability that V_N does not satisfy the condition $\psi^s \wedge \psi^{\Delta}(\xi)$ for another sampled trajectory $\xi \in \Xi_{u_N,\mathcal{V}}$ is at most equal to that ϵ. As such, with a certain confidence, a certificate and controller V_N, u_N *trained* on the basis of N sampled trajectories, will remain a valid certificate for the dynamics with the optimal controller, with probability at least $1-\epsilon$. Therefore, we can argue that V_N is a *probabilistic* control certificate, and hence the property holds (at least) with the same probability.

5.3.2 Probabilistic Guarantees

Our technique for providing a solution to this problem is similar to the method in Chapter 3, but note that we now begin with trajectories generated according to one controller, but provide guarantees on the behaviour of the learnt controller.

Assumption 5.3.2 (Properties of $\mathcal{A}$)**.** *Assume that algorithm $\mathcal{A}$ exhibits the following properties:*

1. Preference: *For any pair of multisets $\mathcal{C}_1$ and $\mathcal{C}_2$ of elements of $\{\xi^i\}_{i=1}^N$, with $\mathcal{C}_1 \subseteq \mathcal{C}_2$, if $\mathcal{C}_1$ does not constitute a compression set of $\mathcal{C}_2$ for algorithm $\mathcal{A}$, then $\mathcal{C}_1$ will not constitute a compression set of $\mathcal{C}_2 \cup \{\xi\}$ for any $\xi \in \Xi$.*
2. Non-associativity: *Let $\{\xi^i\}_{i=1}^{N+\bar{N}}$ for some $\bar{N} \geq 1$. If $\mathcal{C}$ constitutes a compression set of $\{\xi_i\}_{i=1}^N \cup \{\xi\}$ for all $\xi \in \{\xi^i\}_{i=N+1}^{N+\bar{N}}$ for algorithm $\mathcal{A}$, then $\mathcal{C}$ constitutes a compression set of $\{\xi_i\}_{i=1}^{N+\bar{N}}$ (up to a measure-zero set).*
3. Controller Evaluation: *The outcome of the algorithm is unchanged if fed with data $\{\xi_i\}_{i=1}^N \in \Xi_{u_N,\mathcal{V}}^N$ generated according to the learnt controller u_N.*

Note that we introduce an additional assumption compared to Assumption 3.3.2, relating to the learnt controller.

Theorem 5.3.1 (Probabilistic Guarantees)**.** *Consider any algorithm $\mathcal{A}$ satisfying Assumption 5.3.2 such that $V_N = \mathcal{A}(\{\xi^i\}_{i=1}^N) \models \bigwedge_{i=0}^N \psi^\Delta(\xi^i) \wedge \psi^s$, with trajectories $\{\xi^i\}_{i=1}^N$ generated in an i.i.d. manner from a distribution satisfying Assumption 5.3.1. Fix $\beta \in (0,1)$, and for $k < N$, let let $\varepsilon(k,\beta,N)$ be the (unique) solution to the polynomial equation in the interval $[k/N, 1]$*

$$\frac{\beta}{2N}\sum_{m=k}^{N-1}\frac{\binom{m}{k}}{\binom{N}{k}}(1-\varepsilon)^{m-N} + \frac{\beta}{6N}\sum_{m=N+1}^{4N}\frac{\binom{m}{k}}{\binom{N}{k}}(1-\varepsilon)^{m-N} = 1, \tag{5.9}$$

while for $k = N$ let $\varepsilon(N,\beta,N) = 1$. We then have that

$$\begin{aligned}&\mathbb{P}_{u_0}^N\big\{\{\xi^i\}_{i=1}^N \in \Xi_{u_0,\mathcal{V}}^N : \\ &\mathbb{P}_{u_N}\{\xi \in \Xi_{u_N,\mathcal{V}} \colon \neg\phi(\xi)\} \leq \varepsilon(C_N,\beta,N)\big\} \geq 1-\beta.\end{aligned} \tag{5.10}$$

Proof Fix $\beta \in (0,1)$, and consider the data-set $\{\xi^i\}_{i=1}^N \in \Xi_{u_0,\mathcal{V}}$, with compression set $\mathcal{C}$ for algorithm $\mathcal{A}$ (when fed with trajectories generated from these samples). We then construct a mapping from samples to a decision, namely, $V_N, u_N = \mathcal{A}(\{\xi^i\}_{i=1}^N)$, while we impose as an assumption that this mapping satisfies the conditions of Assumption 5.3.2.

As in the proof of Theorem 3.3.1, we have that if the certificate conditions are not satisfied on a new sample, then there will be a change in the compression set when the algorithm is fed all samples plus the new violating sample. Then, we also have (by Assumption 5.3.2) that if we run the algorithm using data from the optimal controller u_N the outcome is unchanged, hence we also know that if the certificate conditions are violated on a new trajectory generated by the controller u_N, then there must also be a change in the compression set.

This derivation establishes the fact that the probability of V_N violating the property when it comes to a new ξ, is bounded by the probability that the compression

set changes, i.e., we have that

$$\mathbb{P}_{u_N}\{\xi \in \Xi_{u_N,\mathcal{V}} \colon V_N \not\models \psi^s \wedge \psi^{\Delta}(\xi))\} \\ \leq \mathbb{P}_{u_N}\{\xi \in \Xi_{u_N,\mathcal{V}} \colon \mathcal{C}_N \neq \mathcal{C}_N^+\}. \tag{5.11}$$

We can now make use of [30, Theorem 7], which implies that with confidence at least $1-\beta$, the probability that for a new $\xi \in \Xi$ the compression set changes, is at most $\varepsilon(\mathcal{C}_N, \beta, N)$, i.e.,

$$\mathbb{P}_{u_N}\{\xi \in \Xi_{u_N,\mathcal{V}} \colon \mathcal{C}_N^+ \neq \mathcal{C}_N\} \leq \varepsilon(C_N, \beta, N), \tag{5.12}$$

where the expression of$\varepsilon(k, \beta, N)$ for different values of k is given in (5.9). By (5.11) and (5.12), we have that

$$\mathbb{P}_{u_0}^N\Big\{\{\xi^i\}_{i=1}^N \in \Xi_{u_0,\mathcal{V}}^N : \\ \mathbb{P}_{u_N}\{\xi \in \Xi_{u_N,\mathcal{V}} \colon V_N \not\models \psi^s \wedge \psi^{\Delta}(\xi))\} \leq \varepsilon(C_N, \beta, N)\Big\} \geq 1-\beta.$$

By the implication in Definition 5.2.1, (5.10) follows, thus concluding the proof. ■

We discuss how our algorithm satisfies Assumption 5.3.2 in the following section.

5.4 Control & Certificate Synthesis

We treat our certificate and controller as an appropriately parameterised "template" (e.g., neural network), and denote the parameter vectors θ, η respectively. We then have that our certificate V_N depends on θ, which is a vector we seek to identify to instantiate our certificate. Similarly u_N depends on η, a separate vector we wish to identify. For the results of this section, we simply write V_θ and u_η, dropping the dependency on N to ease notation. In order to ensure our controller meets the

control limits, we make the following transformation to u_η

$$u_\eta \leftarrow (u_{\max} - u_{\min}) \tanh(u_\eta) + \frac{u_{\min} + u_{\max}}{2}, \tag{5.13}$$

where the tanh function is applied elementwise, and $u_{\max}, u_{\min}$ are vectors defining the limits of the hyperrectangular control region $\mathcal{U}$.

We seek to determine an optimal control parameterization $\eta^\star$, as well as a certificate parameterization $\theta^\star$ that verifies the behaviour of the controller. To this end, for a $\xi \in \Xi$ and parameter vector θ, let

$$L(\theta, \eta, \xi) = l^\Delta(\theta, \eta, \xi) + l^s(\theta), \tag{5.14}$$

represent an associated loss function consisting of a sample-dependent loss l^Δ, and a sample-independent loss l^s. Without loss of generality, we assume that we can drive the sample-independent loss to be zero (see further discussions later). We impose the next mild assumption, needed to prove termination of our algorithm.

Assumption 5.4.1 (Minimisers' Existence)**.** *For any $\{\xi\}_{i=1}^N$, and any non-empty $\mathcal{D} \subseteq \{\xi\}_{i=1}^N$, the set of minimisers of $\max_{\xi \in \mathcal{D}} L(\theta, \eta, \xi)$, is non-empty.*

We aim at approximating minimisers $\theta^\star, \eta^\star$ of the quantity $\max_{\xi \in \mathcal{D}} L(\theta, \eta, \xi)$ when $\mathcal{D} = \{\xi\}_{i=1}^N \in \Xi_{u_\eta, \mathcal{V}}^N$, which exists due to Assumption 5.4.1. We can then use that minimiser to construct $V_{\theta^\star}$ and $u_{\eta^\star}$. To achieve this, we employ a slightly modified version of Algorithm 2, as an inner loop (see Algorithm 4, and construct an outer loop procedure which iteratively updates the controller parameters in Algorithm 5.

Algorithm 4 is similar to Algorithm 2, with modifications to gradient computations to allow for controller parameters, and a change in the jump condition which shows improved performance. In particular, rather than jumping for a misaligned subgradient, we iterate until the parameters demonstrate zero loss on the running compression set, before then jumping to the current maximum sample. This provides an alternative technique for trading between exploitation and exploration, in

Algorithm 4 Inner Loop Optimisation

function $\mathcal{A}(\theta, \eta, \mathcal{D})$
  Set $k \leftarrow 0$ ▷ Initialise iteration index
  Set $\mathcal{C} \leftarrow \varnothing$ ▷ Initialise compression set
  Fix $L_1 < L_0$ with $|L_1 - L_0| > \zeta$ ▷ ζ is any fixed tolerance
  while $l^s(\theta) > 0$ **do** ▷ While sample-independent state loss is non-zero
    $g \leftarrow \nabla_\theta l^s(\theta)$ ▷ Gradient of loss function
    $\theta \leftarrow \theta - \alpha g$ ▷ Step in the direction of sample-independent gradient
  end while
  repeat
    $k \leftarrow k + 1$ ▷ Update iteration index
    $\overline{\xi} \in \arg\max_{\tilde{\xi} \in \mathcal{D}} L(\theta, \eta, \tilde{\xi})$ ▷ Find a sample with maximum loss from $\mathcal{D}$
    $\overline{g}_{\mathcal{D}} \leftarrow \big(\nabla_\theta L(\theta, \eta, \overline{\xi}), \nabla_\eta L(\theta, \eta, \overline{\xi})\big)$ ▷ Subgradients of loss function
    $\overline{\xi}_{\mathcal{C}} \in \arg\max_{\tilde{\xi} \in \mathcal{C}} L(\theta, \eta, \tilde{\xi})$ ▷ Find a sample with maximum loss from $\mathcal{C}$
    $\overline{g}_{\mathcal{C}} \leftarrow \big(\nabla_\theta L(\theta, \eta, \overline{\xi}_{\mathcal{C}}), \nabla_\eta L(\theta, \eta, \overline{\xi}_{\mathcal{C}})\big)$ ▷ Approximate subgradient of loss function for $\tilde{\xi} = \overline{\xi}_{\mathcal{C}}$
    if $L(\theta, \eta, \overline{\xi}_{\mathcal{C}}) \leq 0$ **then** ▷ If loss on compression set is non-positive
      $\theta \leftarrow \theta - \alpha \overline{g}^{\theta}_{\mathcal{D}}$ ▷ Step in the direction of true θ subgradient
      $\eta \leftarrow \eta - \alpha \overline{g}^{\eta}_{\mathcal{D}}$ ▷ Step in the direction of true η subgradient
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{\overline{\xi}\}$ ▷ Update compression set with maximising sample
    else
      $\theta \leftarrow \theta - \alpha \overline{g}^{\theta}_{\mathcal{C}}$ ▷ Step in the direction of approximate θ subgradient
      $\eta \leftarrow \eta - \alpha \overline{g}^{\eta}_{\mathcal{C}}$ ▷ Step in the direction of approximate η subgradient
    end if
    $L_k \leftarrow L_{k-1}$
    if $\max_{\xi \in \mathcal{C}} L(\theta, \eta, \xi) < L_{k-1}$ **then** $L_k \leftarrow \max_{\xi \in \mathcal{C}} L(\theta, \eta, \xi), \theta^\star \leftarrow \theta, \eta^\star \leftarrow \eta$
    end if
  until $|L_k - L_{k-1}| \leq \zeta$ ▷ Iterate until tolerance is met
  return $\theta^\star, \eta^\star, \mathcal{C}_N = \mathcal{C} \cup \arg\max_{\xi \in \mathcal{D}} L(\theta, \eta, \xi)$
end function

Algorithm 5 Controller Synthesis

1: Set $\theta_0, \eta_0 \leftarrow$ Uniform random ▷ Initialise parameters
2: Fix $\mathcal{D}_0 \leftarrow \{x^i(0), v^i\}_{i=1}^N \sim \mathbb{P}^N$ ▷ Sample states and parameters
3: Set $\mathcal{S}_0 \leftarrow \{\xi^i\}_{i=1}^N \in \prod_{i=1}^N \Xi_{u_{\eta_0}, v^i}$ ▷ Uncover trajectories, using data in $\mathcal{D}$
4: Set $\mathcal{I} \leftarrow [1, \dots, N]$ ▷ Initialise index set
5: Set $k \leftarrow 0$
6: **while** $\max_{\xi \in \mathcal{S}_k} L(\theta_k, \eta_k, \xi) > 0$ **do**
7: Set $\widetilde{\mathcal{C}} \leftarrow \varnothing$ ▷ Initialise compression set
8: **repeat**
9: $k \leftarrow k + 1$
10: $\theta_k, \eta_k, \mathcal{C} \leftarrow \mathcal{A}(\theta_{k-1}, \eta_{k-1}, \mathcal{S}_{k-1})$ ▷ Call Algorithm 4
11: $\widetilde{\mathcal{C}} \leftarrow \widetilde{\mathcal{C}} \cup \mathcal{C}$ ▷ Update compression set
12: $\mathcal{I}_{\widetilde{\mathcal{C}}} \leftarrow \mathcal{I}_{\widetilde{\mathcal{C}}} \cup \{i \in [1, |\mathcal{S}_{k-1}|] \colon \mathcal{S}_{k-1}^i \in \widetilde{\mathcal{C}}\}$ ▷ Update running indices
13: $\mathcal{S}_k \leftarrow \{\xi^i\}_{i \in \mathcal{I}} \in \prod_{i \in \mathcal{I}} \Xi_{u_{\eta_k}, v^i}$ ▷ Update trajectories from remaining data
14: **until** $\eta_k = \eta_{k-1}$
15: $\mathcal{S}_k \leftarrow \mathcal{S}_{k-1} \setminus \widetilde{\mathcal{C}}$ ▷ Discard compression set $\widetilde{\mathcal{C}}$ from trajectories for next iteration
16: $\mathcal{I} \leftarrow \{\mathcal{I}^i \in \mathcal{I} \colon i \notin \mathcal{I}_{\widetilde{\mathcal{C}}}\}$ ▷ Update index set
17: **end while**
18: **return** $\theta_k, \eta_k, \mathcal{R}_N = i\mathcal{D}_0 \setminus \mathcal{D}_0^{\mathcal{I}}$

particular, we optimise for the current sample until the controller can satisfy the conditions on that sample (exploitation), and then add a new sample to our compression set once this is achieved (exploration). We hypothesise that, for a sufficiently random initial controller, a single trajectory provides information on a large portion of the state space, and hence optimising for this trajectory provides a controller with reasonable performance across the remaining trajectories.

We provide a graphical representation of this modification in Figure 5.1. Now, we have that the algorithm jumps at M_2, when the loss on the compression set is below 0, and then at M_3, the loss converges on the compression set so we experience another jump (note that this jump would occur after exiting the inner loop algorithm).

Algorithm 5 has been modified to allow for trajectories to be updated each loop to account for the changing controller (see line 13). We now use a set of initial states and parameters (line 2), and generate a set of trajectories from these in line 3. This set of trajectories is updated on every loop *according to the controller uncovered on*

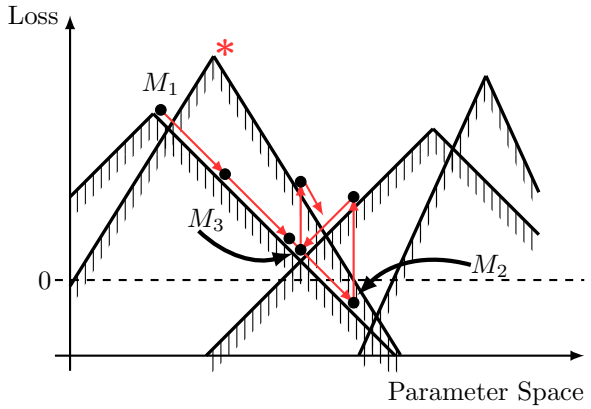


Figure 5.1: Graphical illustration of Algorithm 4.

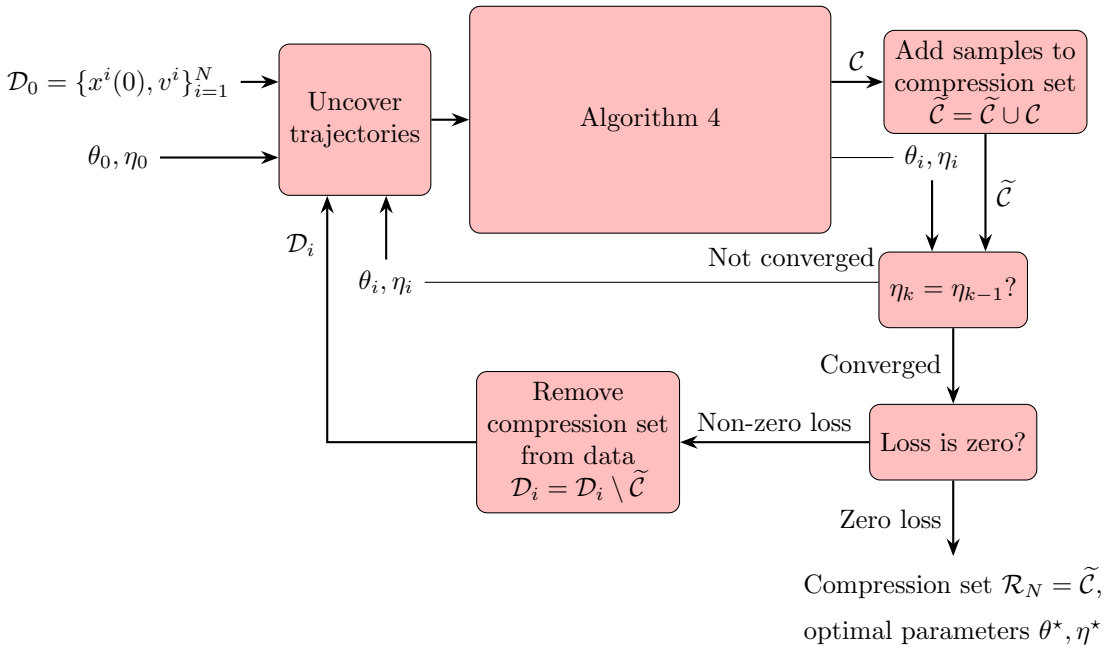


Figure 5.2: Overview of Algorithm 5.

that loop (line 13), whilst a set of indices is used to select the undiscarded data from $\mathcal{D}_0$ for this update (see lines 4,12,16). Then, the discarded samples (and hence the compression set), is calculated as the data contained in $\mathcal{D}_0$ whose index is not in $\mathcal{I}$ (the index set). A visual overview of the algorithm is presented in Figure 5.2.

As discussed in Section 5.3, we may consider stochastic dynamics, in this case it is necessary that we have access to $N \cdot T$ samples of the noise vector (and that the noise distribution be state-independent). Provided these conditions are met, then in line 13 we may make use of these noise samples to uncover a trajectory that

comes from the distribution over trajectories with parameters v^i and controller u_η. Alternatively, if we have the ability to generate samples of the noise we may relax the assumption on state independence, since we can then obtain a trajectory from the true trajectory distribution using these generated samples. If these conditions are not met then we cannot guarantee that the uncovered trajectories are correctly distributed and are unable to make correct guarantees.

We remark that, unlike Chapters 3 and 4, we now require access to a model in order to uncover trajectories generated by the intermediate controller u_{η_k}, and to calculate gradients for our controller. Theoretically, this model may be completely uncertain (albeit with a known structure) if all parameters are uncertain. Without knowledge of the structure of the dynamics, and ability to sample parameters, it would not be possible to apply our techniques.

Throughout this thesis, we employ scenario approach guarantees that depend on the size of the compression set. In the convex case it is known that the size of the compression set scales with the dimension of the uncertainty space [31]. With this in mind, we expect that if more model knowledge is available (in the form of more known, fixed, parameters, and hence a reduced dimension uncertainty space) then the size of the compression set will be reduced. Since we are considering a highly non-convex problem, a straightforward theoretical link between the dimension of the uncertainty space and the size of the compression set is not readily available, and is left to future work.

The sample-independent loss functions are defined identically to those of Section 3.4, whilst the sample-dependent loss is of the following form (the equation provided is for a reach certificate, but other certificates follow analogously)

$$\begin{aligned} l^{\Delta}(\theta, \eta, \xi, v) := \max \Big\{ 0, \max_{k=0,\dots,k_G-1} \Big(V_\theta(f_v(x(k), u_\eta(x(k))) - V_\theta(x(k)) \Big) \\ - \frac{1}{T} \Big(\sup_{x \in \mathcal{X}_I} V_\theta(x) + \delta \Big) \Big\}, \end{aligned} \tag{5.15}$$

where v is the sampled parameter corresponding to trajectory ξ.

5.4.1 Algorithm Properties

Proposition 8 (Algorithm 4 Properties)**.** *Consider Assumption 5.3.1, Assumption 5.4.1 and Algorithm 4 with $\mathcal{D} = \{\xi_i\}_{i=1}^N$ and a fixed (sample independent) initialization for the parameters θ and η. We then have:*

1. *Algorithm 4 terminates, returning parameter vectors $\theta^\star, \eta^\star$ and a set $\mathcal{C}_N$.*
2. *The set $\mathcal{C}_N$ with cardinality $C_N = |\mathcal{C}_N|$ forms a compression set for Algorithm 4.*
3. *Algorithm 4 satisfies Assumption 3.3.2.*

Proof

1. By construction, Algorithm 4 creates a non-increasing sequence of iterates $\{L_k\}_{k\geq 0}$ that is bounded below by the global minimum of $\min_{\xi\in\mathcal{C}} L(\theta, \eta, \xi)$ which exists and is finite due to Assumption 5.4.1. As such, the sequence $\{L_k\}_{k\geq 0}$ is convergent, which in turn implies that Algorithm 4 terminates.

2. We need to show that the set $\mathcal{C}_N$ is a compression set in the sense of Definition 3.3.1 with $\mathcal{A}$ being Algorithm 4 with $\mathcal{D} = \{\xi_i\}_{i=1}^N$. To see this, we "re-run" Algorithm 4 from the same initial choice of the parameter vectors θ, η but with $\mathcal{C}_N$ in place of $\mathcal{D}$. Notice that exactly the same iterates will be generated, as, once the loss on the initial sample is driven to 0, $\mathcal{C}_N$ contains the maximum element in $\mathcal{D}$ by construction, and this is also true for all future iterations. As a result, the same output will be returned, which by Definition 3.3.1 establishes that $\mathcal{C}_N$ is a compression set.

3. We show that all properties of Assumption 3.3.2 are satisfied by Algorithm 4.

Preference: Consider a fixed (sample independent) initialization of Algorithm 4 in terms of the parameters θ, η. Consider also any subsets $\mathcal{C}_1, \mathcal{C}_2$ of $\{\xi^i\}_{i=1}^N$ with $\mathcal{C}_1 \subseteq \mathcal{C}_2$.

Suppose that the compression set returned by Algorithm 4 when fed with $\mathcal{C}_2$ is different from $\mathcal{C}_1$. Fix any $\xi \in \Xi$ and consider the set $\mathcal{C}_2 \cup \{\xi\}$. We will show that the compression set returned by Algorithm 4 when fed with $\mathcal{C}_2 \cup \{\xi\}$ is different from $\mathcal{C}_1$ as well.

When the algorithm reaches step 18 the current maximising sample from $\mathcal{D}$ is added to the compression set, if this is not ξ for any iteration then the algorithm is unchanged across all iterations. Thus the compression set remains the same with that returned when the algorithm is fed only by $\{\xi^i\}_{i=1}^N$. However, the latter is not equal to $\mathcal{C}_1$, thus establishing the claim.

Alternatively, if the new sample ξ appears as a maximizing sample for step 18 of Algorithm 4 ξ is added to the compression returned by Algorithm 4. If $\xi \notin \mathcal{C}_1$ then the resulting compression set will be different from $\mathcal{C}_1$ as it would contain at least one element that is not $\mathcal{C}_1$, namely, ξ.

If $\xi \in \mathcal{C}_1$ then it must also be in $\mathcal{C}_2$ as $\mathcal{C}_1 \subseteq \mathcal{C}_2$. In that case ξ would appear twice in $\mathcal{C}_2 \cup \{\xi\}$, i.e., the set of samples with which Algorithm 4 is fed has ξ as a repeated sample (notice that this can happen with zero probability due to Assumption 5.3.1).

Once one of these repeated samples is added to the compression set returned by Algorithm 4, then the other will never be added. This is since when this other sample appears as a maximizing one in step 11 then its duplicate will already be in the compression set. As such, in order to run step 18, the loss on this sample would have to be non-positive, however, it must also be the maximum (from $\mathcal{D}$), and hence the loss on all samples is non-positive and the algorithm would terminate. As such, one of the repeated ξ's is redundant, which implies that the compression set returned by Algorithm 4 when fed with

$\mathcal{C}_2 \cup \{\xi\}$ is the same with the one that would be returned when it is fed with $\mathcal{C}_2$. However, this would imply that if $\mathcal{C}_1$ is the compression returned by Algorithm 4 when fed with of $\mathcal{C}_2 \cup \{\xi\}$, it will also be the compression set for $\mathcal{C}_2$ (as the duplicate ξ would be redundant). However, the starting hypothesis has been that $\mathcal{C}_1$ is not a compression of $\mathcal{C}_2$. As such, it is not possible for $\mathcal{C}_1$ to be a compression set of $\mathcal{C}_2 \cup \{\xi\}$ as well, establishing the claim.

Non-associativity: Consider a fixed (sample independent) initialization of Algorithm 4 in terms of the parameter θ. Let $\{\xi^i\}_{i=1}^{N+\bar{N}}$ for some $\bar{N} \geq 1$. Suppose that $\mathcal{C}$ is returned by Algorithm 4 a compression set of $\{\xi^i\}_{i=1}^{N} \cup \{\xi\}$, for all $\xi \in \{\xi^i\}_{i=N+1}^{N+\bar{N}}$. Therefore, up to a measure zero set we must have that

$$\mathcal{C} \subset \bigcap_{j=N+1}^{\bar{N}} \left(\{\xi^i\}_{i=1}^{N} \cup \{\xi^j\} \right) = \{\xi^i\}_{i=1}^{N}, \tag{5.16}$$

where the inclusion is since $\mathcal{C}$ is assumed to be returned as a compression set by Algorithm 4 when this is fed with any set within the intersection, while the equality is since by Assumption 5.3.1 all samples in $\{\xi^i\}_{i=1}^{N+\bar{N}}$ are distinct up to a measure zero set. This implies that up to a measure zero set $\mathcal{C}$ should be a compression set returned by Algorithm 4 whenever this is fed with $\{\xi^i\}_{i=1}^{N}$ as any additional sample would be redundant.

Fix now any $\xi \in \{\xi^i\}_{i=N+1}^{N+\bar{N}}$, and consider Algorithm 4 with $\mathcal{D} = \{\xi^i\}_{i=1}^{N} \cup \{\xi\}$. The fact that $\mathcal{C}$ is returned as a compression set for $\{\xi^i\}_{i=1}^{N} \cup \{\xi\}$ implies that whenever ξ is a maximizing sample in step 11 of Algorithm 4 the loss on the compression set is positive. This implies that steps 20–21 are performed and hence ξ is not added to $\mathcal{C}$.

Considering Algorithm 4 this time with $\mathcal{D} = \{\xi^i\}_{i=1}^{N+\bar{N}}$, i.e., fed with all samples at once, due to the aforementioned arguments, whenever a $\xi \in \{\xi^i\}_{i=N+1}^{N+\bar{N}}$ is a maximizing sample in step 11, then the algorithm would proceed to step 20,

and steps 16–18 will not be executed. As such, no such ξ will be added to $\mathcal{C}$.

Hence, the compression set returned by Algorithm 4 when fed with $\{\xi^i\}_{i=1}^{N+\bar{N}}$ would be the same with the one that would be returned if the algorithm was fed with $\{\xi^i\}_{i=1}^{N}$. By (5.16) this then implies that the returned set should be $\mathcal{C}$ up to a measure zero set. ■

Note that, on its own, Algorithm 4 does not satisfy the additional condition in Assumption 5.3.2, and hence we require the addition of Algorithm 5 to provide our guarantees.

Proposition 9 (Algorithm 5 Properties)**.** *Consider Assumption 5.3.1, Assumption 5.4.1 and Algorithm 5 with $\mathcal{D} = \{\xi_i\}_{i=1}^{N}$ and a fixed (sample independent) initialization for the parameters θ and η. We then have:*

1. *Algorithm 5 converges to a minimum loss value of zero, returning a parameter vector $\theta^\star, \eta^\star$ and a set $\mathcal{R}_N$.*
2. *The set $\mathcal{R}_N$ with cardinality $R_N = |\mathcal{R}_N|$ forms a compression set for Algorithm 5.*
3. *Algorithm 5 satisfies Assumption 5.3.2.*

Proof

1. Unlike Algorithm 3, we now have a step which updates the trajectories (line 13), hence the loss value may not necessarily be uniformly decreasing. However, we still have that, if all samples are removed, the loss is zero, since we optimise only the sample-independent loss (and assume this can be driven to zero). Hence, there exists at least one iteration with a zero loss (in the worst-case upon removing all samples). Empirically, we found that it was possible to obtain zero loss without discarding any samples if the system could be controlled to reach the goal set.

2. Consider Algorithm 5 with $\mathcal{D}_0 = \{x^i(0), v^i\}_{i=1}^N$. Denote by $\mathcal{C}_i$ the set returned at step 10 of Algorithm 5, and recall that this set is the compression set returned by Algorithm 4 when this is invoked at that part of the process. Notice then that the set $\mathcal{R}_N$ returned by Algorithm 5 can be expressed as the union of all samples which gave rise to a trajectory contained in any $\mathcal{C}_i$, for all i.

   We need to show that $\mathcal{R}_N$ is a compression set in the sense of Definition 3.3.1 with $\mathcal{A}$ being Algorithm 5 with $\mathcal{D}_0 = \{x^i(0), v^i\}_{i=1}^N$. To see this, we "re-run" Algorithm 5 from the same initial choice of the parameter vector θ but with $\mathcal{R}_N$ in place of $\mathcal{D}_0$. At the first iteration, the set returned in step 10 is $\mathcal{C}_1$ (and the parameter returned would be the same with the one that would be obtained if all samples were employed) as this is a compression set for Algorithm 4 invoked at that step with $\mathcal{D} = \mathcal{R}_\mathcal{N}$. As such, in step 15 we would, have that $\mathcal{D}$ is equal to $\mathcal{R}_N$ without any samples that gave rise to $\mathcal{C}_1$ (since this is a compression set for Algorithm 4 and only samples that are not in any compression set have been removed, and no samples added). Proceeding analogously, we have that Algorithm 5 terminates with the set $\mathcal{R}_N$ remaining intact and $\mathcal{D}$ being empty/ This establishes that $\mathcal{R}_N$ is a compression set for Algorithm 5.

3. Consider first the loop contained in lines 8–14. We know that Algorithm 4 satisfies the preference and non-associativity properties (by Proposition 8). We then update the trajectories according to the current controller u_{η_k}, this having no effect on the compression set, and hence preserving preference and non-associativity. We repeat the loop and obtain a new compression set from Algorithm 4, since the result is added to the existing compression set, then preference and non-associativity are maintained. Thus, this loop satisfies Assumption 3.3.2, and we know that discarding does not affect these properties, so the entire algorithm must satisfy Assumption 3.3.2. Finally, we consider

controller evaluation, due to the iterative nature of our algorithm, the outcome depends on initial conditions θ_0, η_0, when we refer to the algorithm being re-run with trajectories generated according to $u_{\eta^\star}$, it is to be assumed that the parameters are initialised at $\theta^\star, \eta^\star$. Then, since the final loop is conducted with trajectories generated using the controller from the previous loop, and the controller parameters are unchanged (as per line 14), then rerunning the algorithm would lead to immediate termination, leaving θ and η unchanged. Hence, the algorithm satisfies Assumption 5.3.2. ■

5.5 Numerical Results

For all numerical simulations, we considered a confidence level of $\beta = 10^{-5}$, $N = 1000$ samples; our results are averaged across 5 independent repetitions, each with different multi-samples. By sample complexity, we refer to the number of *trajectory samples*, separate to the states used for the sample-independent loss since these samples can be obtained without accessing the system dynamics. Unless otherwise stated, we use a uniform distribution (but our techniques naturally extend to unknown distributions, see Section 5.3).

5.5.1 Reachability for Unstable System

First, we consider the model

$$x(k+1) = \begin{bmatrix} 0.7 & 0.8 \\ v & 1.4 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(x(k)), \tag{5.17}$$

with v being randomly distributed between -0.3 and -0.5. With no control input this system is unstable for all parameter values, however, we wish to design a controller (with limits $\mathcal{U} = [-1, 1]$), which can get to within 1 unit of the origin, with $X_I = [-1, 1] \times [4, 4.5]$. After an average of 60,090 seconds (standard deviation

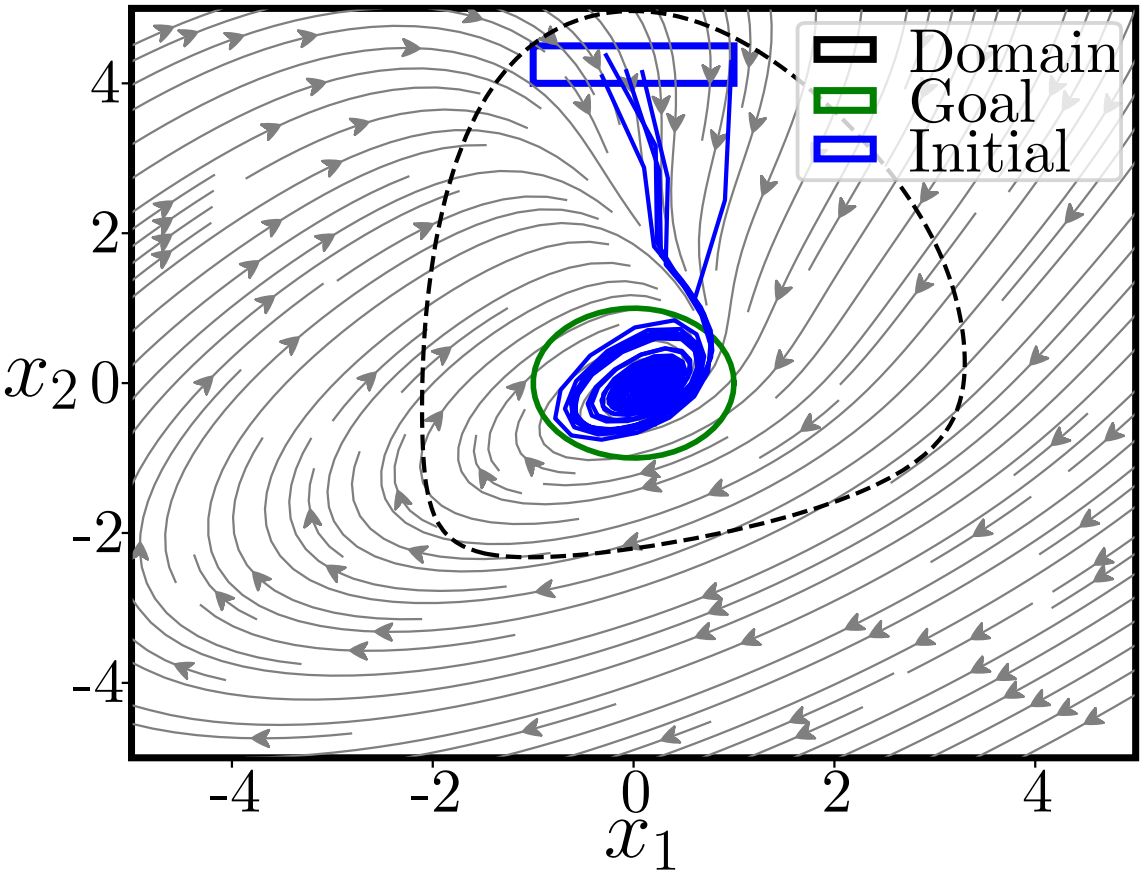


Figure 5.3: Phase plane plot for a sampled system with optimal controller, additional randomly generated trajectories (for other systems) are shown in blue, and the 0-level set is portrayed by a dashed line.

16,660s) we obtain a reach certificate, with risk 0.068 (standard deviation 0.013). We provide a phase plane plot of one sampled system in Figure 5.3, the 0- and $-\delta$- level sets are shown with a dashed line. In Figure 5.4 we provide a plot of the optimal control input for each state, this demonstrates that the controller learns to apply a strong negative input initially, which is then reduced in magnitude as the system approaches the goal set.

5.5.2 Safety Certificate

We also consider a model of the form

$$\begin{aligned} x_1(k+1) &= x_1(k) - Tx_2(k) + Tu_1(x(k)) \\ x_2(k+1) &= x_2(k) + T(x_1(k) + vx_2(k)) + Tu_2(x(k)), \end{aligned} \tag{5.18}$$

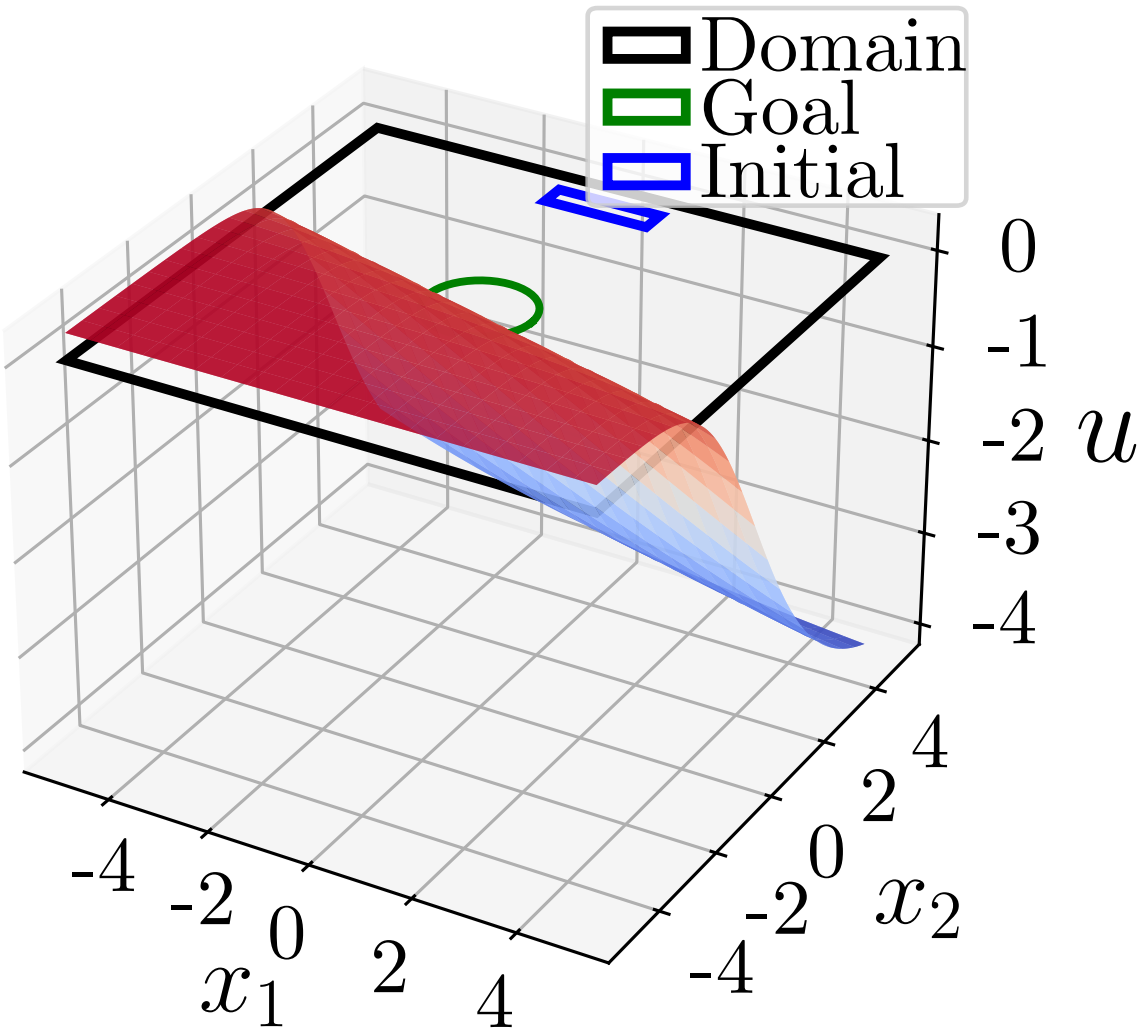


Figure 5.4: Optimal controller values across domain.

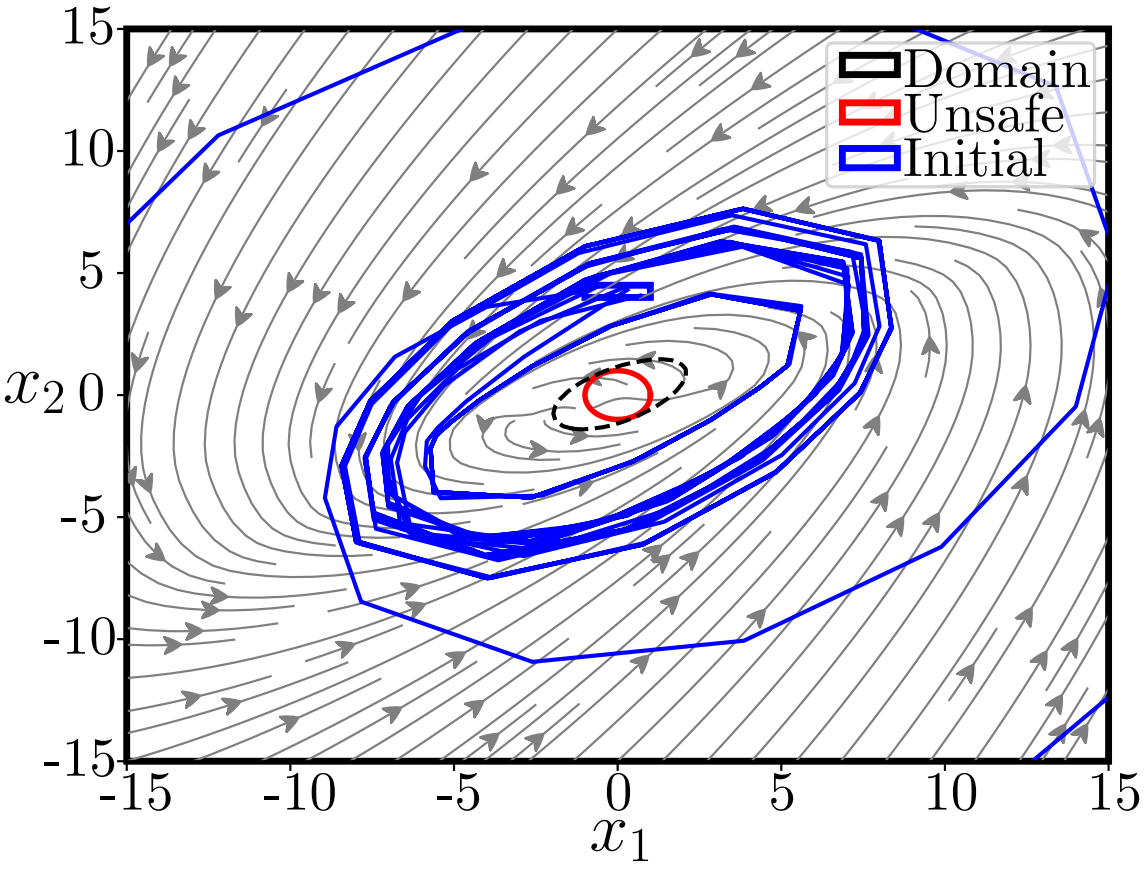


Figure 5.5: Phase plane plot for a sampled system with optimal controller, additional randomly generated trajectories (for other systems) are shown in blue, and the 0-level set is portrayed by a dashed line.

with $v \in [-2, 0]$ and $\mathcal{U} = [-2, 2]^2$. Note that this model is very similar to one studied in Section 3.6, and is naturally stable for all parameter values. We seek to find a controller that guarantees we do not enter within 1 unit of the origin with $X_I = [-1, 1] \times [4, 4.5]$. Our techniques required an average of 5760 seconds (standard deviation 630s) to obtain a safety certificate with risk 0.113 (standard deviation 0.013). In Figure 5.5, we provide a phase plane plot for a sampled system, as well as trajectories for 5 further sampled systems and initial conditions, the 0-level set of the certificate is shown with a dashed line. From this figure we can see that the learned controller drives the system away from the origin, safely avoiding the unsafe region around the origin.

5.6 Conclusion

We have proposed a method for synthesis of robust controllers for uncertain parametric dynamical systems, based only a finite number of sampled system parameters. These controllers are optimised to verify a number of core temporally extended specifications. In addition, we accompany our controller (and dynamical system) with a certificate which provides assertions on the property of interest We provide PAC guarantees on the validity of our certificate for the controlled system when it comes to a new parameter.

Our numerical experiments demonstrate the efficacy of our methods on a number of examples.

6 Summary and Future Research Directions

Verifying complex properties of dynamical systems using data is an exciting and important area of ongoing research. In this thesis, we have considered two approaches for data-driven verification, one based on the use of an abstraction, and one abstraction-free technique. We have also explored how we can solve the problem of co-design in order to synthesise a controller (or policy), as well as providing guarantees on the behaviour of that controller, without requiring an additional verification dataset. Below we summarise the main results of each chapter of this thesis, as well as providing some avenues for future research.

6.1 Chapter 2: Finite State Control/Policy Synthesis

In Chapter 2 we investigated the synthesis of a robust policy to verify a probabilistic computation tree logic (PCTL) formula on an uncertain parametric Markov decision process (upMDP). These upMDPs provide a valuable modelling framework, since they provide a simple abstraction of a complex system, but still allow us to consider uncertainty arising due to either modelling uncertainty (for example uncertainty in the loading of a vehicle) or due to external factors (for example uncertainty in wind direction). These models can also be viewed as a probabilistic counterpart to robust MDPs, where the goal is to be robust to all values of the uncertainty. In contrast

with this approach, upMDPs allow us to avoid being overly conservative since we seek to be robust *up to some risk level*. The use of PCTL formulae allows us to consider complex specifications which can be expressed in a succinct manner.

We developed novel techniques for the synthesis of probabilistically robust policies. This is in contrast to prior work in [13], which considered finding a policy for each parameter instantiation. By developing a policy that is probabilistically robust to parameter instantiations we do not require knowledge of the parameters in order to apply our policy. We considered policies within three different policy classes, namely deterministic, behavioural and mixed policies, and provided techniques for synthesisng robust policies within each class.

We then provide probably approximately correct (PAC) guarantees on the satisfaction of the given PCTL formula. This guarantees were calculated by capitalising on recent advancements in the so-called scenario approach theory. Due to differences in the structure of the synthesis procedure for policies in each policy class, we provided separate guarantees for each policy class.

Finally, we applied our techniques in a number of numerical experiments, providing a comparison with alternative procedures for synthesisng policies for upMDPs, as well as a comparison between the various policy classes.

This work opened up a number of interesting avenues for future research. First, our techniques were found to be computationally expensive, requiring many hours to synthesise a policy. In order to aid practical application, it is of interest to improve this performance, possible methods to achieve this include using faster techniques to find an approximate subgradient at each step, or performing some form of warm-starting to initialise our optimisation (e.g. by starting from a policy that is optimal for at least one parameter instantiation). Such improvements in performance would allow us to explore a further avenue in sample-and-discarding results, which provides a framework to trade guarantees for performance (i.e. by allowing us to discard a sample on which our formula is particularly difficult to satisfy).

Additionally, our techniques for synthesis of deterministic policies was particularly expensive, but these offer an arguably more understandable policy class. Hence, we may wish to investigate techniques for less computationally expensive synthesis of these policies, possibly by utilising a subgradient descent algorithm to steer us towards the optimal deterministic policy. Such a procedure is closely related to solving mixed integer programs, and may suffer from similar drawbacks (e.g. the optimal deterministic policy may be arbitrarily far from the optimal behavioural policy).

Finally, we were unable to provide any convergence guarantees for our subgradient algorithm for behavioural policy synthesis. Such guarantees would be a valuable addition to this work, but require further consideration due to the non-convex nature of the problem.

6.2 Chapters 3 and 4: Continuous State Verification

In Chapters 3 and 4 we consider the problem of certificate synthesis for continuous-state systems in both discrete and continuous time. We considered certificates which allowed us to verify a number of properties, namely, reachability, safety and reach-while-avoid.

By synthesising certificates directly on the continuous-state system we avoid solving the complex task of constructing an abstraction, as well as any approximation errors introduced by such an abstraction. In order to allow for the synthesis of certificates for any general continuous-state system we allow for the use of any parameterised function approximator as certificate template. This allows us to use, for example, neural networks as certificate templates, thus allowing for the approximation of any function within a certain function space [63].

Many existing certificate synthesis techniques consider having access to a model of the system in order to verify the certificate conditions. However, obtaining such a

model of the system is in general difficult, and requires domain-specific knowledge. Furthermore, the guarantees provided by such techniques are strongly dependent on the quality of the models. Hence, we avoid building a model of the system, and learn the certificate directly from sampled trajectories from the system. Hence we seek to solve a so-called scenario program, with these trajectories entering our certificate synthesis optimisation program through constraints.

We wish to make use of all our data in order to synthesise a certificate, and then provide guarantees on the validity of our certificate when it comes to a new, and unseen, system trajectory. Once again, we turn to scenario approach techniques to obtain PAC guarantees, making use of the most recent extensions to obtain such guarantees.

Synthesising one of these certificates requires, in general, solving a non-convex optimisation problem. Solving such a problem is non-trivial, and applying the scenario approach theory to such problems is difficult. To solve this, we introduce a novel algorithm, which makes use of a generalised version of subgradients to iteratively improve our solution, whilst avoiding uninformative data in order to allow for strong guarantees. This algorithm has potential for application to any general non-convex scenario program.

In addition, we include a discarding mechanism to allow for trading guarantees for optimality, and, in particular for certificate synthesis, to allow us to discard trajectories that will not satisfy the certificate conditions for any certificate. This is necessary so that we can provide a guarantee on a new trajectory satisfying the certificate conditions (and hence the underlying property under study).

In Chapter 4, we investigated applying our techniques to continuous time systems. In contrast to discrete time systems, continuous time models have trajectories that have infinite-dimension (since they map to a state for all continuous time points). This prevents us performing computations on these trajectories, and we therefore consider a time-discretised approximation, taking samples of states at a

finite number of time instances along the trajectory, and using these to estimate a derivative at each sampled time. However, directly applying our techniques to these time-discretised trajectories suffers a significant drawback since we are unable to guarantee property satisfaction between sample times. Hence, we considered finding a bound on the change in the certificate conditions between sampled points, and therefore, provided we can exceed the satisfaction of the certificate conditions by at least this bound, we guarantee property satisfaction on the entire continuous time trajectory through computations only on its time-discretised approximation.

We provide proof that the algorithm both with, and without, discarding eventually terminates, and that with discarding this can be shown to occur at the global minimum. Finally, we apply our techniques to a number of benchmark problems, and compare our results to similar existing results in the literature for certificate synthesis.

These certificates were limited to autonomous systems (or systems with a pre-calculated controller), and we address the problem of controller synthesis in Chapter 5. Some other avenues for extending this research include extensions to more complex properties (and compositions of properties) to allow for verification of general logical formulae, as well as further investigations on the properties of our algorithm. As with Chapter 2, our algorithm makes use of subgradient-like techniques in a non-convex landscape, and the existence of convergence guarantees in such a set-up is an open question. As discussed, our algorithm should be applicable to other non-convex scenario programs, and investigating its application there is another exciting avenue of future research.

Finally, our guarantees throughout have been of a probably approximately correct nature (with two layers of probability). In some instances (such as in Table 3.2), it is preferable to make the inner probability as close to 0 as possible. In these cases, a relevant area of research is that which bridges the gap between scenario programs and robust programs to achieve only a single layer of probability [47], which have

previously been applied to certificate-based problems [85, 106]. Therefore, we may be interested in applying such techniques to our framework.

However, to do so would require some additional knowledge which have not assumed access to thus far. In particular, we would require some knowledge about the distribution from which trajectories are drawn, as well as bounds on the Lipschitz constants of the problem. With this information, it may then be possible to obtain a guarantee with a single layer of probability, provided a sufficient amount of data.

Due to the very high-dimensional nature of our sample space (each sample being a trajectory), and the fact that these techniques' data requirements scale exponentially with the dimension of the sample space, it is likely that implementing these bounds would require vast amounts of data. In contrast, the PAC guarantees we employ allow for lower data requirements, and reduced conservatism at the cost of this 2 layer probability. Such robust techniques are still of interest, and exploring if there may be some inherent structure in the samples- for example, due to the dynamics- that could be exploited to reduce the data requirements is an interesting open question.

6.3 Chapter 5: Controller Synthesis with Verification for Uncertain Parametric Models

Finally, in Chapter 5 we consider the problem of controller synthesis with guarantees. In particular, we seek to optimise a controller to satisfy a given specification, and accompany the resulting controlled system with a certificate. Once again, we seek to utilise all available data for training, rather than saving some data for testing.

In this chapter, we considered an uncertain parametric dynamical model, in some ways similar to the upMDP of Chapter 2, but now with a continuous state space. These uncertain parametric models provide a natural way for us to model known physical laws, whilst maintaining a formal approach to uncertainty. We consider nonlinear control models and require only that the model has unique solutions for

controllers taking values within the valid control set.

In order to achieve this, we adapted Algorithms 2 and 3 to optimise controller parameters, as well as iteratively updating trajectories based on the sampled parameters and initial states. Then, we verified the properties of the algorithms to ensure we could still provide PAC guarantees using the scenario approach theory. Finally, we provided an evaluation of our proposed techniques on a number of numerical experiments.

This work offers one of the first combinations of data-driven certificate and controller synthesis, and, to the best of our knowledge, the first that considers a parametrically uncertain system. This work provides a starting point for a number of exciting directions of future work. Perhaps the most interesting, and challenging, of these is to allow for purely data-driven synthesis and guarantees, that is, a method for learning a controller and providing guarantees on its behaviour, without the need to update trajectories to match candidate controllers at each step. Such a method may be possible, but requires some further research into the scenario approach theory, in particular, this would induce a distribution shift on trajectories (from being generated by an initial controller to an optimal one), and providing guarantees in such a setting is difficult.

Additionally, there is potential to investigate the link between our techniques, and other techniques for controller synthesis. In particular, there may be a link between the certificate difference conditions we study, and other techniques such as reinforcement learning through the Bellman equation (and the Hamilton–Jacobi–Bellman equation in continuous time). In the future, we hope to explore this link to relate certificate learning to reinforcement learning and optimal control, and potentially to provide probabilistic guarantees on these methods.

Bibliography

[1] Control for Societal-scale Challenges: Road Map 2030 | IEEE Control Systems Society. https://www.ieeecss.org/control-societal-scale-challenges-road-map-2030.

[2] A. Abate. Formal verification of complex systems: model-based and data-driven methods. In *MEMOCODE*, pages 91–93. ACM, 2017.

[3] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo. FOSSIL: A software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *HSCC*, pages 24:1–24:11. ACM, 2021.

[4] A. Abate, M. Giacobbe, and D. Roy. Stochastic omega-regular verification and control with supermartingales. In A. Gurfinkel and V. Ganesh, editors, *Computer Aided Verification*, pages 395–419, 2024.

[5] A. Abate, M. Giacobbe, D. Roy, and Y. Schnitzer. Model checking and strategy synthesis with abstractions and certificates. In *Principles of Verification (2)*, volume 15261 of *Lecture Notes in Computer Science*, pages 360–391. Springer, 2024.

[6] D. Ahmed, A. Peruffo, and A. Abate. Automated and sound synthesis of lyapunov functions with smt solvers. In *Proceedings of TACAS, LNCS 12078*, pages 97–114, 2020.

[7] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding. In *AAAI*, pages 2669–2678. AAAI Press, 2018.

[8] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *ECC*, pages 3420–3431. IEEE, 2019.

[9] M. Anand and M. Zamani. Formally verified neural network control barrier certificates for unknown systems. *IFAC-PapersOnLine*, 56(2):2431–2436, 2023.

[10] K. J. Arrow, E. W. Barankin, D. Blackwell, R. Bott, N. Dalkey, M. Dresher, D. Gale, D. B. Gillies, I. Glicksberg, O. Gross, S. Karlin, H. W. Kuhn, J. P. Mayberry, J. W. Milnor, T. S. Motzkin, J. Von Neumann, H. Raiffa, L. S. Shapley, M. Shiffman, F. M. Stewart, G. L. Thompson, and R. M. Thrall. *Contributions to the Theory of Games (AM-28), Volume II*. Princeton University Press, 1953.

[11] D. Aussel, J.-N. Corvellec, and M. Lassonde. Mean Value Property and Subdifferential Criteria for Lower Semicontinuous Functions. *Transactions of the American Mathematical Society*, 347(10):4147–4161, 1995.

[12] T. S. Badings, A. Abate, N. Jansen, D. Parker, H. A. Poonawala, and M. Stoelinga. Sampling-Based Robust Control of Autonomous Systems with Non-Gaussian Noise. In *AAAI*, pages 9669–9678. AAAI Press, 2022.

[13] T. S. Badings, M. Cubuktepe, N. Jansen, S. Junges, J.-P. Katoen, and U. Topcu. Scenario-based verification of uncertain parametric MDPs. *Int. J. Softw. Tools Technol. Transf.*, 24(5):803–819, 2022.

[14] T. S. Badings, L. Romao, A. Abate, and N. Jansen. Probabilities are not enough: Formal controller synthesis for stochastic dynamical models with epistemic uncertainty. In *AAAI*, pages 14701–14710. AAAI Press, 2023.

[15] T. S. Badings, L. Romao, A. Abate, D. Parker, H. A. Poonawala, M. Stoelinga, and N. Jansen. Robust control for dynamical systems with non-gaussian noise via formal abstractions. *Journal of Artificial Intelligence Research*, 76:341–391, 2023.

[16] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[17] C. Banks, S. Coogan, and M. Egerstedt. LTL Cross Entropy Optimization for Quadcopter Task Orchestration. page 35.

[18] C. Belcastro. Parametric uncertainty modeling: an overview. In *ACC*, volume 2, pages 992–996 vol.2. IEEE, 1998.

[19] C. Belta, B. Yordanov, and E. Aydin Gol. *Formal Methods for Discrete-Time Dynamical Systems*, volume 89 of *Studies in Systems, Decision and Control*. Springer International Publishing, Cham, 2017.

[20] J. Bhandari and D. Russo. Global optimality guarantees for policy gradient methods. *CoRR*, abs/1906.01786, 2019.

[21] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams. A Probabilistic Particle-Control Approximation of Chance-Constrained Stochastic Predictive Control. *IEEE Trans. Robotics*, 26(3):502–517, 2010.

[22] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014.

[23] 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. *Annu. Rev. Control. Robotics Auton. Syst.*, 5:411–444, 2022.

[24] G. C. Calafiore and M. C. Campi. The scenario approach to robust control design. *IEEE Trans. Autom. Control.*, 51(5):742–753, 2006.

[25] M. E. Campbell and S. C. O. Grocott. Parametric uncertainty model for control design and analysis. *IEEE Trans. Control. Syst. Technol.*, 7(1):85–96, 1999.

[26] M. Campi and S. Garatti. *Introduction to the Scenario Approach*. SIAM Series on Optimization, 2018.

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

[28] M. C. Campi and S. Garatti. A sampling-and-discarding approach to chance-constrained optimization: Feasibility and optimality. *Journal of Optimization Theory and Applications*, 148(2):257–280, 2011.

[29] M. C. Campi and S. Garatti. Wait-and-judge scenario optimization. *Math. Program.*, 167(1):155–189, 2018.

[30] M. C. Campi and S. Garatti. Compression, generalization and learning. *J. Mach. Learn. Res.*, 24:339:1–339:74, 2023.

[31] M. C. Campi, S. Garatti, and M. Prandini. The scenario approach for systems and control design. *Annu. Rev. Control.*, 33(2):149–157, 2009.

[32] M. C. Campi, S. Garatti, and F. A. Ramponi. A general scenario theory for nonconvex optimization and decision making. *IEEE Trans. Autom. Control.*, 63(12):4067–4078, 2018.

[33] P. C. Chandrasekharan. *Robust Control of Linear Dynamical Systems*. Academic, London, 1996.

[34] Y. Chang, N. Roohi, and S. Gao. Neural lyapunov control. In *NeurIPS*, pages 3240–3249, 2019.

[35] H. Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *The Annals of Mathematical Statistics*, 23(4):493 – 507, 1952.

[36] E. M. Clarke, A. Fehnker, Z. Han, B. H. Krogh, O. Stursberg, and M. Theobald. Verification of Hybrid Systems Based on Counterexample-Guided Abstraction Refinement. In H. Garavel and J. Hatcliff, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 9th International Conference, TACAS 2003, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings*, volume 2619 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2003.

[37] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-Guided Abstraction Refinement. In *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2000.

[38] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Society for Industrial and Applied Mathematics, 1990.

[39] R. Coppola, A. Peruffo, and M. M. Jr. Data-driven abstractions for verification of linear systems. *IEEE Control. Syst. Lett.*, 7:2737–2742, 2023.

[40] J. Coulson, J. Lygeros, and F. Dörfler. Data-enabled predictive control: In the shallows of the deepc. In *ECC*, pages 307–312. IEEE, 2019.

[41] M. Cubuktepe, N. Jansen, S. Junges, J.-P. Katoen, and U. Topcu. Convex Optimization for Parameter Synthesis in MDPs. *IEEE Trans. Autom. Control.*, 67(12):6333–6348, 2022.

[42] H. Dai, B. Landry, M. Pavone, and R. Tedrake. Counter-example guided synthesis of neural network lyapunov functions for piecewise linear systems. In *CDC*, pages 1274–1281. IEEE, 2020.

[43] C. Daws. Symbolic and Parametric Model Checking of Discrete-Time Markov Chains. In *ICTAC*, volume 3407 of *Lecture Notes in Computer Science*, pages 280–294. Springer, 2004.

[44] C. Dawson, S. Gao, and C. Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Trans. Robotics*, 39(3):1749–1767, 2023.

[45] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk. A Storm is Coming: A Modern Probabilistic Model Checker. In *CAV (2)*, volume 10427 of *Lecture Notes in Computer Science*, pages 592–600. Springer, 2017.

[46] A. Edwards, A. Peruffo, and A. Abate. Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models. In *HSCC*, pages 26:1–26:10. ACM, 2024.

[47] P. M. Esfahani, T. Sutter, and J. Lygeros. Performance Bounds for the Scenario Approach and an Extension to a Class of Non-convex Programs. *IEEE Trans. Autom. Control.*, 60(1):46–58, Jan. 2015.

[48] M. V. Faronov and I. G. Polushin. Sliding mode control of rotary drilling systems with parametric uncertainty. In *CCTA*, pages 352–358. IEEE, 2023.

[49] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *NeurIPS*, pages 11423–11434, 2019.

[50] S. Floyd and M. K. Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Mach. Learn.*, 21(3):269–304, 1995.

[51] F. G. Foster. Dynamic Programming and Markov Processes. By R. A. Howard. Pp. 136. 46s. 1960. (John Wiley and Sons, N.Y.). *The Mathematical Gazette*, 46(358):340–341, 1962.

[52] P. Frihauf, M. Krstic, and T. Basar. Nash equilibrium seeking in noncooperative games. *IEEE Trans. Autom. Control.*, 57(5):1192–1207, 2012.

[53] S. Garatti and M. C. Campi. Risk and complexity in scenario optimization. *Math. Program.*, 191(1):243–279, 2022.

[54] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice - A survey. *Autom.*, 25(3):335–348, 1989.

[55] E. M. Hahn, T. Han, and L. Zhang. Synthesis for PCTL in Parametric Markov Decision Processes. In M. G. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, editors, *NASA Formal Methods - Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*, volume 6617 of *Lecture Notes in Computer Science*, pages 146–161. Springer, 2011.

[56] E. M. Hahn, H. Hermanns, and L. Zhang. Probabilistic reachability for parametric Markov models. *Int. J. Softw. Tools Technol. Transf.*, 13(1):3–19, 2011.

[57] L. P. Hansen, T. J. Sargent, G. Turmuhambetova, and N. Williams. Robust control and model misspecification. *J. Econ. Theory*, 128(1):45–90, 2006.

[58] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, Sept. 1994.

[59] J. Heinrich, M. Lanctot, and D. Silver. Fictitious Self-Play in Extensive-Form Games. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 805–813. JMLR.org, 2015.

[60] C. Hensel, S. Junges, J.-P. Katoen, T. Quatmann, and M. Volk. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4):589–610, 2022.

[61] M. W. Hirsch, S. Smale, and R. L. Devaney. Differential equations, dynamical systems, and an introduction to chaos. 2003.

[62] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

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

[64] G. N. Iyengar. Robust Dynamic Programming. *Math. Oper. Res.*, 30(2):257–280, 2005.

[65] P. Jagtap, S. Soudjani, and M. Zamani. Formal synthesis of stochastic systems via control barrier certificates. *IEEE Trans. Autom. Control.*, 66(7):3097–3110, 2021.

[66] W. Jin, Z. Wang, Z. Yang, and S. Mou. Neural certificates for safe control policies. *CoRR*, abs/2006.08465, 2020.

[67] S. Junges, E. Ábrahám, C. Hensel, N. Jansen, J.-P. Katoen, T. Quatmann, and M. Volk. Parameter Synthesis for Markov Models. *CoRR*, abs/1903.07993, 2019.

[68] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[69] K. C. Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Math. Program.*, 90(1):1–25, 2001.

[70] J. C. Knight. Safety critical systems: challenges and directions. In *ICSE*, pages 547–550. ACM, 2002.

[71] A. Kozarev, J. F. Quindlen, J. P. How, and U. Topcu. Case studies in data-driven verification of dynamical systems. In *HSCC*, pages 81–86. ACM, 2016.

[72] I. Kozine and L. V. Utkin. Interval-Valued Finite Markov Chains. *Reliab. Comput.*, 8(2):97–113, 2002.

[73] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In G. Gopalakrishnan and S. Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[74] A. Lavaei, S. Soudjani, E. Frazzoli, and M. Zamani. Constructing mdp abstractions using data with formal guarantees. *IEEE Control. Syst. Lett.*, 7:460–465, 2023.

[75] K. Lee and P. A. Fishwick. Dynamic model abstraction. In *WSC*, pages 764–771. IEEE Computer Society, 1996.

[76] L. Ljung. *System identification : theory for the user*. Prentice Hall information and system sciences series. Prentice Hall, 2nd ed. edition, 1999.

[77] A. M. Lyapunov. The general problem of the stability of motion. 1994.

[78] A. Majumdar, A. Farid, and A. Sonar. Pac-bayes control: learning policies that provably generalize to novel environments. *Int. J. Robotics Res.*, 40(2-3), 2021.

[79] K. Margellos, M. Prandini, and J. Lygeros. On the connection between compression learning and scenario based single-stage and cascading optimization problems. *IEEE Trans. Autom. Control.*, 60(10):2716–2721, 2015.

[80] I. Meedeniya, I. Moser, A. Aleti, and L. Grunske. Evaluating probabilistic models with uncertain model parameters. *Softw. Syst. Model.*, 13(4):1395–1415, 2014.

[81] Y. Meng and J. Liu. Robustly complete finite-state abstractions for verification of stochastic systems. In *FORMATS*, volume 13465 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2022.

[82] P. Mohajerin Esfahani, T. Sutter, and J. Lygeros. Performance bounds for the scenario approach and an extension to a class of non-convex programs. *IEEE Transactions on Automatic Control*, 60(1):46–58, 2015.

[83] J. Nash. Non-cooperative games. *Cournot Oligopoly*, pages 82–94, Jan. 1989.

[84] M. Nazeri, T. Badings, A.-K. Schmuck, S. Soudjani, and A. Abate. Data-driven abstraction and synthesis for stochastic systems with unknown dynamics. In *CDC*, pages 6754–6759. IEEE, 2025.

[85] A. Nejati, A. Lavaei, P. Jagtap, S. Soudjani, and M. Zamani. Formal verification of unknown discrete- and continuous-time systems: A data-driven approach. *IEEE Trans. Autom. Control.*, 68(5):3011–3024, 2023.

[86] A. Nilim and L. E. Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Oper. Res.*, 53(5):780–798, 2005.

[87] I. Osband, B. V. Roy, D. J. Russo, and Z. Wen. Deep exploration via randomized value functions. *J. Mach. Learn. Res.*, 20:124:1–124:62, 2019.

[88] D. Paccagnan, M. C. Campi, and S. Garatti. The pick-to-learn algorithm: Empowering compression for tight generalization bounds and improved post-training performance. In *NeurIPS*, 2023.

[89] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In *CDC*, pages 3482–3487. IEEE, 2002.

[90] S. Park, E. Serpedin, and K. A. Qaraqe. Gaussian Assumption: The Least Favorable but the Most Useful [Lecture Notes]. *IEEE Signal Process. Mag.*, 30(3):183–186, 2013.

[91] A. Platzer. Logics of Dynamical Systems. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 13–24. IEEE Computer Society, 2012.

[92] A. Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.

[93] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. *Games Econ. Behav.*, 63(2):642–662, 2008.

[94] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *HSCC*, volume 2993 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2004.

[95] S. Prajna, A. Jadbabaie, and G. J. Pappas. Stochastic safety verification using barrier certificates. In *CDC*, pages 929–934. IEEE, 2004.

[96] S. Prajna, A. Jadbabaie, and G. J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. Autom. Control.*, 52(8):1415–1428, 2007.

[97] A. Puggelli, W. Li, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. Polynomial-Time Verification of PCTL Properties of MDPs with Convex Uncertainties. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 527–542. Springer, 2013.

[98] T. Quatmann, C. Dehnert, N. Jansen, S. Junges, and J. Katoen. Parameter synthesis for markov models: Faster than ever. In *ATVA*, volume 9938 of *Lecture Notes in Computer Science*, pages 50–67, 2016.

[99] J.-F. Raskin and O. Sankur. Multiple-Environment Markov Decision Processes. In *FSTTCS*, volume 29 of *LIPIcs*, pages 531–543. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

[100] D. Ren, W. Lu, J. Lv, L. Zhang, and B. Xue. Model predictive control with reach-avoid analysis. In *IJCAI*, pages 5437–5445. ijcai.org, 2023.

[101] L. Rickard, A. Abate, and K. Margellos. Learning robust policies for uncertain parametric Markov decision processes. In *L4DC*, volume 242 of *Proceedings of Machine Learning Research*, pages 876–889. PMLR, 2024.

[102] L. Rickard, A. Abate, and K. Margellos. Continuous-time data-driven barrier certificate synthesis. In *CDC*, pages 5794–5799. IEEE, 2025.

[103] L. Rickard, A. Abate, and K. Margellos. Data-driven certificate synthesis. *Autom.*, 185:112798, 2026.

[104] L. Rickard, T. S. Badings, L. Romao, and A. Abate. Formal controller synthesis for markov jump linear systems with uncertain dynamics. In *QEST*, volume 14287 of *Lecture Notes in Computer Science*, pages 10–29. Springer, 2023.

[105] L. Romao, A. Papachristodoulou, and K. Margellos. On the exact feasibility of convex scenario programs with discarded constraints. *IEEE Trans. Autom. Control.*, 68(4):1986–2001, 2023.

[106] A. Salamati, A. Lavaei, S. Soudjani, and M. Zamani. Data-driven verification and synthesis of stochastic systems via barrier certificates. *Autom.*, 159:111323, 2024.

[107] D. Scheftelowitsch, P. Buchholz, V. Hashemi, and H. Hermanns. Multi-Objective Approaches to Markov Decision Processes with Uncertain Transition Parameters. In *VALUETOOLS*, pages 44–51. ACM, 2017.

[108] X. Shang, J. Cortés, and Y. Zheng. Willems' fundamental lemma for nonlinear systems with koopman linear embedding. *IEEE Control. Syst. Lett.*, 8:3135–3140, 2024.

[109] O. Smouni, M. L. Nachidi, A. Rabhi, and H. Midavaine. Enhanced h∞ control for DC-DC converters under parametric uncertainty: Experimental verification. In *MED*, pages 328–333. IEEE, 2025.

[110] P. Solanki, N. Vertovec, Y. Schnitzer, J. V. Beers, C. de Visser, and A. Abate. Certified approximate reachability (care): Formal error bounds on deep learning of reachable sets, 2025.

[111] S. E. Z. Soudjani and A. Abate. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *Siam Journal On Applied Dynamical Systems*, 12(2):921–956, 2013.

[112] H. Von. Stackelberg. The Theory of Market Economy. Translated from the German and with an Introduction by A.T. Peacock. London, Edinburgh, Glasgow, W. Hodge & Co, Ltd., 1952, xxiii p. 328 p., 25/-. *Recherches Économiques de Louvain/ Louvain Economic Review*, 18(5):543–543, 1952.

[113] D. Sun, S. Jha, and C. Fan. Learning certified control using contraction metric. In *CoRL*, volume 155 of *Proceedings of Machine Learning Research*, pages 1519–1539. PMLR, 2020.

[114] J. L. Svensen, X. Cheng, S. Boersma, and C. Sun. Chance-constrained stochastic MPC of greenhouse production systems with parametric uncertainty. *Comput. Electron. Agric.*, 217:108578, 2024.

[115] M. van der Vegt, N. Jansen, and S. Junges. Robust Almost-Sure Reachability in Multi-Environment MDPs. In *TACAS (1)*, volume 13993 of *Lecture Notes in Computer Science*, pages 508–526. Springer, 2023.

[116] A. Virmaux and K. Scaman. Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *NeurIPS*, pages 3839–3848, 2018.

[117] A. Wachi and Y. Sui. Safe reinforcement learning in constrained markov decision processes. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9797–9806. PMLR, 2020.

[118] W. Wiesemann, D. Kuhn, and B. Rustem. Robust Markov Decision Processes. *Math. Oper. Res.*, 38(1):153–183, 2013.

[119] J. C. Willems, P. Rapisarda, I. Markovsky, and B. D. Moor. A note on persistency of excitation. *Syst. Control. Lett.*, 54(4):325–329, 2005.

[120] G. R. Wood and B. P. Zhang. Estimation of the Lipschitz constant of a function. 8(1):91–103, 1996.

[121] Y. Yang, H. Hu, T. Wei, S. E. Li, and C. Liu. Scalable synthesis of formally verified neural value function for hamilton-jacobi reachability analysis. *CoRR*, abs/2407.20532, 2024.

[122] N. Yousefimanesh, I. Bos, and D. Vermeulen. Strategic rationing in Stackelberg games. *Games Econ. Behav.*, 140:529–555, 2023.

[123] P. Yu and D. V. Dimarogonas. Distributed Motion Coordination for Multirobot Systems Under LTL Specifications. *IEEE Trans. Robotics*, 38(2):1047–1062, 2022.