# A Learning-Based Approach to Safety for Uncertain Robotic Systems

*Anayo Akametalu*
*Claire Tomlin, Ed.*
*Pieter Abbeel, Ed.*
*Lawrence Evans, Ed.*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2018-41

http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-41.html

May 10, 2018

## Acknowledgement

First, I would like to thank my committee members Professor Claire Tomlin, Professor Pieter Abbeel, and Professor Craig Evans for the technical guidance given to me along the way.

I also want to thank my parents Anayo and Josephine Akametalu. It's been 24 years since we left Nigeria to settle in Los Angeles, and everything that they have done for my siblings and I was to make achievements like this possible. Over the past six years I have had the privilege to struggle for a PhD, which was only possible because of their twenty plus years struggling to succeed in a foreign land. Watching these two immigrants build a good life for themselves and their children is the single most influential experience of my life, and I wish to build upon this legacy. This dissertation is a brick.

# A Learning-Based Approach to Safety for Uncertain Robotic Systems

by

Anayo K. Akametalu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Claire J. Tomlin, Chair
Professor Pieter Abbeel
Professor Lawrence C. Evans

Spring 2018

The dissertation of Anayo K. Akametalu, titled A Learning-Based Approach to Safety for Uncertain Robotic Systems, is approved:

Chair Date

Date

Date

University of California, Berkeley

A Learning-Based Approach to Safety for Uncertain Robotic Systems

# Abstract

A Learning-Based Approach to Safety for Uncertain Robotic Systems

by

Anayo K. Akametalu
Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley
Professor Claire J. Tomlin, Chair

Robotic systems are becoming more pervasive, and have the potential to significantly improve human lives. However, for these benefits to be realized it is critical that the safe operation of these systems be guaranteed. Reachability analysis has proven to be an effective tool for providing safety certificates for dynamical systems, given a model of the system. A major challenge in assuring safety, is that systems often have uncertainty due to the hard-to-model complex physical interactions, or lack of knowledge of the behavior of external agents, on which safety may depend.

This thesis uses Hamilton-Jacobi (HJ) reachability analysis to robustly guarantee safety for systems with uncertainty. In the presence of uncertainty there must be a balance between conservativeness as it pertains to safety and performance as it pertains to other system objectives, and we also account for this through reachability analysis. In addition, this thesis also explores methods for modifying the analysis as more data is collected from the robotics system, which ultimately allows for improved performance. This is referred to here as learning-based reachability analysis. The thesis concludes with a new HJ reachability formulation that enhances the learning-based analysis. The myriad of ideas presented throughout the thesis are demonstrated on various examples.

# Contents

# Acknowledgments

Just a week away from graduation, and it's surreal to know that I am almost at the finish line. I would be remiss if I did not take time to acknowledge the people that have helped and encouraged me along the way.

First, I would like to express my appreciation and gratitude to my committee members Professor Claire Tomlin, Professor Pieter Abbeel, and Professor Craig Evans for the technical guidance given to me along the way. Their expertise in optimal control, PDEs, and reinforcement learning have no doubt contributed to this thesis. I would like to especially acknowledge my advisor, Claire Tomlin, for always being a present, calm, kind, and patient force during my PhD journey. Many times I would go into our weekly one on one meetings ready to implode, and just as many times I would leave feeling strengthened and ready to attack my challenges, instead of attacking myself.

In addition to Claire, I want to thank the following individuals for conversations that got me through some rough patches: Chukwuebuka Nweke, Somil Bansal, Daniel Calderone, Oscar Dubon, and Christine Zhou.[1] [2] Thank you for being present when I felt at my lowest.

In spite of the lows, I still had many wonderful moments along this journey. Perhaps, the most meaningful was publishing my first paper, which would not have been possible without the help, guidance, and mentorship of the *Safe Learning Team*: Jaime Fisac, Melanie Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire. I would also like to acknowledge the other collaborators that I had the great pleasure of working with: Jerome Thai, Mo Chen, Shromona Ghosh, Forrest Laine, Frank Jiang, and Somil.

The special people that I met were not only limited to research. I have been privileged to be a part of many amazing groups during my time at Berkeley including: the Hybrid Systems Lab, semi-autonomous, 2013 linear systems prelim study group, BGESS (Black Graduate Engineering and Science Students), GSIs for inaugural offering of EE16A, OSMO (Oakland Science Math Outreach), 1715 (formerly 1943)[3], and my IM basketball team. I met many awesome people through these groups, some which I now consider good friends.

Next, I want to thank the wonderful staff at UC Berkeley that make it possible for us to (mostly) just focus on research and learning. In particular I want to recognize: Jessica Gamble, Shirley Salanio, Diana Lizarraga, Sheila Humphreys, Susanne Kauer,

[1] Fortunately, I met Oscar before he became the Vice Chancellor of Equity and Inclusion, a role that he is extremely deserving of, when he still had the time to meet and reply to my text messages. Now he is big time!

[2] Christine Zhou is a university student health services counselor for engineering students. Our conversations were very therapeutic, and I would like to encourage grad students to take advantage of such services if the journey gets too overwhelming. Self-care is important.

[3] This is a special shout out to my roommates Onye Okafor, Tayo Olukoya, and Casey Mackin for our countless adventures and making sure that I always had work life balance.

Meltem Erol, Tiffany Reardon, Mary Stewart, and Myra Rose. They made sure I was taken care of when it came to managing funding, scheduling classes, organizing conference travel, making reservations for important events, and so much more. Most importantly they just checked in on me.

Lastly, I want to express my greatest gratitude to my parents Anayo and Josephine Akametalu. It's been 24 years since we left Nigeria to settle in Los Angeles, and everything that they have done for my siblings and I was to make achievements like this possible. Over the past six years I have had the privilege to struggle for a PhD, which was only possible because of their twenty plus years struggling to succeed in a foreign land. I still remember my dad working night shifts for FedEx, where he loaded boxes.[4] During the day he would take classes at a trade school with aspirations of better employment. I also remember my mom juggling school with her job at a group home in a far away city. She had just learned to drive when we came to the states, and the long commute combined with LA traffic was stressful to say the least. Fortunately, those days are far behind us and we are in a much better place now. Watching these two immigrants build a good life for themselves and their children is the single most influential experience of my life, and I wish to build upon this legacy. This dissertation is a brick.

[4] I am not sure if he still has the navy blue jumpsuit.

# Chapter 1

# Introduction

From unmanned aerial vehicles (UAVs) to self-driving cars, robotic systems are becoming ever more pervasive in civilian lives. The benefits of these technologies are limitless, and have the potential to drastically change the world. In developing countries with poor road networks, there is the possibility for medicine delivery by UAVs. In more developed countries self-driving technology can lead to efficiency gains in the workforce by reducing traffic, and allow for more effective uses of real estate by eliminating the need for nearby parking.

In light of these benefits, there are also many challenges that come with integrating robots into everyday life. One particular challenge central to this thesis, is that of ensuring the safe operation of these systems. Safety here is best thought of as constraint satisfaction, e.g. avoiding collisions with obstacles and other agents or having an aircraft operate within its flight envelope to prevent damage. Analyzing safety typically requires a model of that system to aid in understanding how the system can potentially violate its constraints, and how to prevent such incidents from happening. Unfortunately, developing exact models can be difficult due to complex physical interactions, e.g. drag forces and friction, and unobservable states, e.g. if you do not know what a human driver wants to do, how can you ensure collision avoidance?

The things that are not modeled can be lumped together as uncertainty, which we assume here is bounded, e.g. we know upper and lower bounds for the drag force. The focus of this thesis is to ensure safety of robotic systems with uncertainty, and a few key insights that will be developed throughout are:

- Uncertainty leads to a trade-off between conservativeness and performance.
- A model and its uncertainty can be obtained by learning from observations of the system.
- The trade-off that we choose between conservativeness and performance should most accurately (and quickly) reflect the current model and uncertainty we have

of the system.

## 1.1 Reachability for Safety under Uncertainty

The main theoretical tool used and expanded upon in this thesis is reachability analysis. Reachability is extremely powerful for the formal verification of the safety of dynamical systems. The idea is quite simple: Imagine we have a vehicle that is constrained to drive along a straight line, and is approaching a yellow traffic light. Obviously, we do not want the vehicle caught in the intersection during a red light. We need to either accelerate at full throttle or slam on the brake. However, depending on the vehicle speed, distance to the intersection, and time the traffic violation might be inevitable. Reachability helps us determine the scenarios, or more technically states, from which we can remain safe, and it also provides us with the action that should be taken to do so. The more accurate the model of our vehicle, then the more precise the results obtained from reachability analysis. However, there might be things that are hard to model like the speed dependent drag forces or the friction forces, which depend on the road surface material.

We can choose to not model these effects explicitly and treat them as uncertainties. If the uncertainty is bounded, reachability can still be used to do a worst-case safety analysis in which it returns the states from which safety can be maintained in spite of any possible realization of the vehicle dynamics.

We focus primarily on Hamilton-Jacobi (HJ) reachability analysis, which is the most general formulation. It can handle a wide variety of constraint-satisfaction problems, and can also be applied to general controlled nonlinear systems with disturbances or adversarial behavior. In fact the worst-case analysis for systems with bounded uncertainty is accomplished by treating the uncertainty as an adversarial disturbance that aims to have the system violate the constraints. This approach returns a control law that is guaranteed to ensure safety as long as the true system is captured by the model and the disturbances being considered. Accounting for more disturbances leads to a more conservative control law that is more likely to ensure safety, however conservative behavior can degrade the system's performance with respect to other objectives. Sticking with the previous example, if we considered drag forces that were significantly greater than those experienced by the vehicle this might cause us to apply the brake even when it is not absolutely necessary, and thus increase the time it takes to get to the end destination. One of the challenges that we address in this thesis is managing this trade-off between conservativeness and performance in the face of uncertainty.

Before continuing, it would be important to mention the major limitation of HJ reachability, which is that the approach can become quickly prohibitive since the computational and storage complexity scale exponentially with the state space dimension. Addressing this problem is still an active research topic, and we refer the interested reader to [21] for some of the most recent approaches. Here, we will not address this problem and we will restrict our examples to systems with low dimensionality (two to four).

## 1.2 Learning about Safety

As we stated earlier uncertainty forces us to strike a balance between conservativeness and performance. However, as uncertainty (and the model) change so must our point of equilibrium. For example, think of what happens when someone is just learning to drive. In the beginning when the person is less certain of her abilities and what other drivers will do they drive more conservatively, opting for lower speeds and hitting the brake regularly. The focus is more on safety and less on other objectives like passenger comfort or minimizing time to reach the destination. However, over time as the person learns a better model of other drivers and her own abilities, she can ensure safety without behaving as conservatively thus making it possible to improve her performance with respect to other objectives.

Observations from the system can be used to constantly improve the model and reassess uncertainty. To get the most out of our robotic systems it's imperative that any safety analysis that is performed can quickly be updated to reflect the current system model and uncertainty. This is a major focus of this thesis. Furthermore, we will conclude this thesis with a novel formulation for reachability analysis that is more amenable to changes in the system model and uncertainty.

## 1.3 Outline

Our aim in this thesis is to ensure safety for uncertain robotic systems, while minimally degrading performance on other objectives. The thesis will be laid out as such:

- Chapter 2 contains the theoretical and implementation details for HJ reachability analysis as it pertains to constraint-satisfaction problems. Here we show that the problem can be solved by finding a solution to a related HJ equation, which is ultimately solved using dynamic programming.

- Chapter 3 introduces and focuses on the safe learning problem, which is learning-based control (e.g. adaptive control, reinforcement learning, etc.) with safety critical constraints. This particular problem is interesting because we are considering systems with uncertainty. In the presence of uncertainty, the system can asymptotically learn a control policy that achieves better performance than a handcrafted policy from the designer. In general this learning occurs through trial and error. We show how HJ reachability can be used in this setting to ensure safety throughout the learning process.

- Chapter 4 focuses on different learning-based (or data-driven) approaches to reachability analysis. As we alluded to earlier, as more observations are obtained from the system, the model and uncertainty will change, and it's important that these changes are reflected in the reachability analysis.

- Chapter 5 presents a new formulation of reachability analysis. In general the chapter focuses on developing the formulation and explaining its properties. We also show how the formulation can improve the learning-based reachability methods presented in Chapter 4.

This thesis is the compilation of results and examples of papers, both published [2], [4], [35] and submitted, from many co-authors. These will be cited as appropriate at the start of each chapter.

# Chapter 2

# Background

*This chapter is adapted from the material presented in [35].*

A key theme of this work is on safety in dynamical systems with uncertainty, which is studied here as a reachability problem. In a reachability problem, one is given a dynamical system described by an ordinary differential equation (ODE), a target set describing the final conditions under consideration, and a state constraint-set describing the conditions under consideration throughout operation of the dynamical system. Depending on the context one might be interested in a number of questions including: Which initial states can reach the target? Which initial states can avoid the target? Which initial states can reach the target while staying within the constraint set?, etc. In the context being considered, safety is simply just constraint satisfaction and the reachability problem we are interested in is finding the initial states from which the system can always stay within the constraint set.

Hamilton-Jacobi (HJ) reachability analysis is a theoretical tool from optimal control and differential games used to solve reachability problems. The main idea is to define a functional that scores trajectories (and control signals) based on how well they achieve a desired objective (reaching the target, staying within the constraint, etc.), and then to evaluate the states based on the score of the optimal trajectories starting from those states. In an optimal control problem the system under consideration has one set of inputs, and it is through these inputs that the trajectories are optimized. In a differential game we consider a system with two sets of inputs and a *zero sum outcome*, thus one set of inputs is optimizing the trajectories to achieve the largest payoff and the other is optimizing to minimize the payoff. The second set of inputs can be used to represent bounded uncertainties or disturbances, in which case the analysis boils down to a worst-case examination of the system.

HJ reachability analysis has proven to be extremely reliable in safety-critical applications because of the correctness guarantees that it can make about the underlying system, without needing to explicitly simulate from controllers from an infinite number of states. The tool is also very flexible and can be applied to a wide range of dynamical systems, and system constraints. Lastly, HJ reachability analysis produces a control strategy that guarantees the system constraints are satisfied during system operation. For these reasons the tool has been used in a number of safety-critical applications including emergency landing of unmanned aerial vehicles (UAVs) [30], vehicle platooning [24], safe learning [2], [42], collision-avoidance [46], [62], and many others [29], [48]. The need for tools that can provide correctness guarantees will only continue to increase, as we look towards a future that will see a greater integration of autonomous systems in everyday life due to increased interest in UAVs for civilian purposes [6], [10], [15], [37], [68], autonomous cars [19], [77], and domestic robots [88], [89].

There are many numerical tools available for solving the reachability problem under the HJ formulation, which boils down to solving an accompanying partial differential equation (PDE) or variational inequality (VI) [63], [69], [79]. However, these methods require approximating the solution to the PDE/VI on a grid, thus for a fixed accuracy the computation and storage of the approximation scales exponentially with the number of system states. Unfortunately, computational demands, in general, make these techniques impractical for systems with many states (above six). In some specific cases people have been able to leverage the structure of the dynamics to decompose a high-dimensional problem into lower dimensions [22], [23], [51]–[53], [61], [64]. There have also been more tractable techniques for solving the reachability problem. However, these methods are typically not as general and place restrictions on the dynamics model, e.g. linear systems. Some of the better known techniques attempt to approximate the solution with ellipsoids or hyperplanes [38], [44], [58].

In this chapter we will formalize the concepts above, and present the HJ reachability formulation as a tool for safety in systems with bounded uncertainty/disturbances.

## 2.1 System Model

Consider a fully observable system whose underlying dynamics may be non-deterministic, but bounded. We can formalize this as a dynamical system with state $x \in \mathbb{R}^n$, and two inputs, $u \in \mathcal{U} \subset \mathbb{R}^{n_u}, d \in \mathcal{D} \subset \mathbb{R}^{n_d}$ (with $\mathcal{U}$ and $\mathcal{D}$ compact) which we will refer to as the *controller* and the *disturbance*:

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

The flow field $f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{D} \to \mathbb{R}^n$ is assumed to be Lipschitz continuous and bounded. In the single-input case we drop the disturbance input, and just have $f(x, u) : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$.

Letting $\mathbb{U}$ and $\mathbb{D}$ denote the collections of measurable[1] functions $\boldsymbol{u} : [0, \infty) \to \mathcal{U}$ and $\boldsymbol{d} : [0, \infty) \to \mathcal{D}$ respectively, and allowing the controller and disturbance to choose

[1] A function $f : X \to Y$ between two measurable spaces $(X, \Sigma_X)$ and $(Y, \Sigma_Y)$ is said to be

any such signals, the evolution of the system from any initial state $x$ is determined (see for example [28], Ch. 2, Theorems 1.1, 2.1) by the unique continuous trajectory $\xi : [0, \infty) \to \mathbb{R}^n$ solving

$$\begin{aligned} \dot{\xi}(s) &= f(\xi(s), \boldsymbol{u}(s), \boldsymbol{d}(s)), \text{ a.e. } s \geq 0 \ , \\ \xi(0) &= x \ . \end{aligned} \tag{2.2}$$

Note that this is a solution in Carathéodory's *extended sense*, that is, it satisfies the differential equation *almost everywhere* (i.e. except on a subset of Lebesgue measure zero).

Throughout our analysis, we will use the notation $\xi_x^{\boldsymbol{u},\boldsymbol{d}}(\cdot)$ to denote the state trajectory $t \mapsto x$ corresponding to the initial condition $x \in \mathbb{R}^n$, the control signal $\boldsymbol{u} \in \mathbb{U}$ and the disturbance signal $\boldsymbol{d} \in \mathbb{D}$.

## 2.2 Safety via Hamilton-Jacobi Reachability Analysis

Safety can be posed as a constraint satisfaction problem. There are certain regions of the state space in which the system should only operate in, or equivalently there are certain regions that it should avoid at all cost. Clearly, a system designer should ensure that the system is never initialized in these undesirable states, but how can she ensure that the system will satisfy these constraints indefinitely? Is it even possible to do so? HJ reachability answers these questions and related questions.

### 2.2.1 State-Constraint

A central element in our problem is the *state-constraint set*, which defines a region $\mathcal{K} \subseteq \mathbb{R}^n$ of the state space, typically resulting from safety considerations, where the system is required to remain at all times. For technical purposes detailed below, we assume that this set is closed; no further assumptions (boundedness, connectedness, convexity, etc.) are made in the analysis. Our problem can thus be stated as finding the *safe set* $\Omega(\mathcal{K})$, the set of states $x$ from which our system can start and we can guarantee that a control signal exists to keep the state trajectory within $\mathcal{K}$ irrespective of the disturbance signal. This is also referred to as the discriminating kernel of $\mathcal{K}$ [9].

### 2.2.2 Minimum distance to target

Next we introduce the *target set* $\mathcal{T}$, which can in general represent a set of states that we want to drive the system to or that we want the system to avoid. In general

measurable if the preimage of a measurable set in $Y$ is a measurable set in $X$, that is: $\forall V \in \Sigma_Y, f^{-1}(V) \in \Sigma_X$, with $\Sigma_X, \Sigma_Y$ $\sigma$-algebras on $X$,$Y$.

the reachability problem is defined in terms of $\mathcal{T}$, and the task is to find the *reachable set* $\mathcal{R}(\mathcal{T})$, which is the set of states that will eventually enter $\mathcal{T}$. Note that we are considering this problem in the infinite-horizon setting, since we only care if a state can eventually reach the target.

Reachability analysis can be employed for constraint-satisfaction by selecting complement of the state-constraint set as the target $\mathcal{T} = \overline{\mathcal{K}}$, thus the safe set would be the complement of the reachable set $\Omega(\mathcal{K}) = \overline{\mathcal{R}(\mathcal{T})}$. In the context of constraint satisfaction the target set may also be referred to as the keep-out set or avoid set.

The target set $\mathcal{T}$ can be implicitly characterized as the sub-zero level set of a Lipschitz *surface function* $l : \mathbb{R}^n \to \mathbb{R}$:

$$x \in \mathcal{T} \iff l(x) < 0 \ . \tag{2.3}$$

This function always exists, since we can simply choose the *signed distance function* to $\mathcal{T}$, $s_{\mathcal{T}}(x)$, which is Lipschitz continuous by construction.[2] We use a clipped signed distance since the problem is solved over a fixed domain in practice, $l(x) = \min(\max(s_{\mathcal{T}}(x), -L), L)$ with $L > 0$, where $L$ is usually taken to be the largest value (in magnitude) on the domain.

To express whether a given trajectory *ever* violates the constraints, let the functional $\mathcal{V} : \mathbb{R}^n \times \mathbb{U} \times \mathbb{D} \to \mathbb{R}$ assign to each initial state $x$ and input signals $\boldsymbol{u}(\cdot)$, $\boldsymbol{d}(\cdot)$ the lowest value of $l(\cdot)$ achieved by trajectory $\xi_x^{\boldsymbol{u},\boldsymbol{d}}(\cdot)$ over all times $t \geq 0$:

$$\mathcal{V}\big(x, \boldsymbol{u}(\cdot), \boldsymbol{d}(\cdot)\big) := \inf_{t \geq 0} l\big(\xi_x^{\boldsymbol{u},\boldsymbol{d}}(t)\big). \tag{2.4}$$

This *outcome* $\mathcal{V}$ will be strictly smaller than zero if there exists any $t \in [0, \infty)$ at which the trajectory leaves the constraint set, and will be nonnegative if the system remains in the constraint set for all of $t \geq 0$. Denoting $\mathcal{V}^{\boldsymbol{u},\boldsymbol{d}}(x) = \mathcal{V}\big(x, \boldsymbol{u}(\cdot), \boldsymbol{d}(\cdot)\big)$, the following statement follows from (2.3) and (2.4) by construction.

**Proposition 1** *The set of points $x$ from which the system trajectory $\xi_x^{\boldsymbol{u},\boldsymbol{d}}(\cdot)$ under given inputs $\boldsymbol{u}(\cdot) \in \mathbb{U}, \boldsymbol{d}(\cdot) \in \mathbb{D}$ will remain in the constraint set $\mathcal{K}$ at all times $t \geq 0$ is equal to the zero superlevel set of $\mathcal{V}^{\boldsymbol{u},\boldsymbol{d}}(\cdot)$:*

$$\{x \in \mathbb{R}^n : \forall t \geq 0,\ \xi_x^{\boldsymbol{u},\boldsymbol{d}}(t) \in \mathcal{K}\} = \{x \in \mathbb{R}^n : \mathcal{V}^{\boldsymbol{u},\boldsymbol{d}}(\cdot) \geq 0\}.$$

Guaranteeing safe evolution from a given point $x \in \mathbb{R}^n$ requires determining whether there exists a control input $\boldsymbol{u}(\cdot) \in \mathbb{U}$ such that, for all disturbance inputs $\boldsymbol{d}(\cdot) \in \mathbb{D}$ satisfying $\boldsymbol{d}(t) \in \mathcal{D}$, the evolution of the system remains in $\mathcal{K}$, or equivalently $\mathcal{V}^{\boldsymbol{u},\boldsymbol{d}}(x) \geq 0$.

[2] For any nonempty set $\mathcal{M} \subset \mathbb{R}^m$, the signed distance function $s_{\mathcal{M}} : \mathbb{R}^m \to \mathbb{R}$ is defined as $\inf_{y \in \mathcal{M}} |z - y|$ for points outside of $\mathcal{M}$ and $-\inf_{y \in \mathbb{R}^m \setminus \mathcal{M}} |z - y|$ for points inside $\mathcal{M}$, where $|\cdot|$ denotes a norm on $\mathbb{R}^m$.

Our safe set can then be obtained by solving a differential game (or optimal control problem in the case where there is no disturbance). The game is played between the control and disturbance signal with the restriction that the disturbance signal can only use *nonanticipative strategies.* The set of nonanticipative strategies for the disturbance is $\mathcal{B} = \{\boldsymbol{\beta} : \mathbb{U} \to \mathbb{D} \mid \forall t \geq 0,\ \forall \boldsymbol{u}(\cdot), \hat{\boldsymbol{u}}(\cdot) \in \mathbb{U}, \big(\boldsymbol{u}(\tau) = \hat{\boldsymbol{u}}(\tau) \text{ a.e.} \tau \geq 0\big) \Rightarrow \big(\boldsymbol{\beta}[\boldsymbol{u}](\tau) = \boldsymbol{\beta}[\hat{\boldsymbol{u}}](\tau) \text{ a.e.} \tau \geq 0\big)\}$. The idea here is that disturbance can only make decisions based on the actions of the control up to present time. In this sense the disturbance has a slight advantage. See [62] for a detailed discussion on information pattern. The outcome of the game starting from state $x$ is represented by the *value function* $V(x)$,

$$V(x) := \inf_{\beta[\boldsymbol{u}](\cdot) \in \mathcal{B}} \sup_{\boldsymbol{u} \in \mathbb{U}} \mathcal{V}\big(x, \boldsymbol{u}(\cdot), \boldsymbol{d}(\cdot)\big). \tag{2.5}$$

We can now formally introduce the safe set $\Omega(\mathcal{K})$.

**Definition 1** *We say that a point $x$ is in the safe set $\Omega(\mathcal{K})$ of constraints $\mathcal{K}$ if and only if the system trajectory $\xi_x^{\boldsymbol{u},\boldsymbol{d}}$ starting at $x$, with both players acting optimally, remains in $\mathcal{K}$ for all time $t \geq 0$:*

$$\begin{aligned}\Omega(\mathcal{K}) := \{x \in \mathbb{R}^n : \forall \boldsymbol{\beta}(\cdot) \in \mathcal{B}, \exists \boldsymbol{u}(\cdot) \in \mathbb{U},\\ \forall t \geq 0, \xi_x^{\boldsymbol{u},\boldsymbol{\beta}[\boldsymbol{u}]}(t) \in \mathcal{K}\},\end{aligned}$$

The following classical result follows from Proposition 1.

**Proposition 2** *The safe set of the constraint set $\mathcal{K}$ is the zero superlevel set of the value function $V$:*

$$\Omega(\mathcal{K}) = \{x \in \mathbb{R}^n : V(x) \geq 0\}.$$

From the above proposition it follows that the reachable set can also be obtained from the value function

$$\mathcal{R}(\mathcal{T}) = \{x \in \mathbb{R}^n : V(x) < 0\}.$$

It has been shown that the value function for games with outcome given by equation (2.4) can be characterized as the unique viscosity solution to a variational inequality [13], [14] as shown in [36]. An alternative formulation involves a modified PDE[62]. In a finite-horizon setting when the game is played over a finite interval the $[0, T]$, the *finite-horizon value function* $V(x, t)$ can be computed by solving the following HJ equation:

$$0 = \min\left\{l(x) - V(x,t), \frac{\partial V}{\partial t}(x,t) + \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} \frac{\partial V}{\partial x}(x,t) f(x,u,d)\right\} \tag{2.6a}$$

$$V(x, T) = l(x). \tag{2.6b}$$

As $T \to \infty$, $V(x,t)$ becomes independent of $t$. We accordingly drop the dependence on $t$ and recover $V(x)$ as defined in (2.5), which we sometimes refer to as the *safety function*. Note that $V(x,t)$ can be used to characterize the safe set over a fixed horizon, but our focus is on safety in the infinite horizon setting.

For conciseness we introduce *control policies* $\pi(\cdot) : \mathbb{R}^n \to \mathcal{U}$ and *disturbance policies* $\rho(\cdot) : \mathbb{R}^n \to \mathcal{D}$, which map from state to control and disturbance, respectively. The optimal minimax action for both the control and disturbance is given by a state feedback policy.

**Definition 2** *The* optimal safe policy $\pi^*(\cdot)$ *is the solution to the optimization*[3]*:*

$$\pi^*(x) = \arg\max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} \frac{\partial V}{\partial x}(x) f(x,u,d).$$

Policy $\pi^*(\cdot)$ attempts to drive the system to the safest possible state.

### 2.2.3 Computing the Value Function

Several approximation schemes have been proposed for solving (2.6) and similar PDEs and variational equalities on a fixed grid $G$. The schemes can be broken into two approaches, Eulerian schemes [62], [69], [79] and semi-Lagrangian schemes [12], [34]. Both schemes approximate the PDE by characterizing the behavior of the PDE/VI at specific grid points in the state space. In the Eulerian framework the PDE is approximated directly, using finite-differences to approximate the partial derivatives. In the semi-Lagrangian framework the dynamics are discretized and the value function is characterized by how it changes along trajectories. Since all points along the trajectory are not on the grid, an interpolation scheme is also required to approximate those values.

Here we will use a semi-Lagrangian approximation based on a discrete time dynamic programming (DP) principle:

$$V^{k+1}_{\Delta t}(x) = \min\{l(x), \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} V^k_{\Delta t}(x + \Delta t f(x,u,d))\} \ , \tag{2.7a}$$

$$V^0_{\Delta t}(x) = l(x) \ , \tag{2.7b}$$

$$V_{\Delta t} = \lim_{k\to\infty} V^k_{\Delta t} \ , \tag{2.7c}$$

where $V_{\Delta t}(x)$ converges to $V(x)$ as the discrete time step $\Delta t \to 0$. With the semi-Lagrangian approximation, the value function is solved on a discrete grid. Representing the approximation in vectorized form, $\vec{V} \in \mathbb{R}^{n_G}$, the semi-Lagrangian approach

[3] Typically the optimal solution will be a singleton, although in general the arg max need not be unique, which leads to $\pi^* : \mathbb{R}^n \to 2^{\mathcal{U}}$. However, since we can always choose one element of the set arbitrarily, we will assume for simplicity a policy $\pi^* : \mathbb{R}^n \to \mathcal{U}$, that is, a unique mapping from states to control inputs.

yields

$$\vec{V}_i^0 = l(x_i) \ , \tag{2.8a}$$
$$\vec{V}_i^{k+1} = \min\{l(x_i), \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} I[\vec{V}^k](x_i + \Delta t f(x_i, u, d))\} \ , \tag{2.8b}$$
$$\vec{V} = \lim_{k\to\infty} \vec{V}^k \ , \tag{2.8c}$$

for $i = 1, ..., n_G$, where $\{x_i\}_{i=1}^{n_G}$ are the grid nodes, $n_G$ is the number of grid nodes, $\vec{V}_i^k$ is the approximate value for $V(x_i, k\Delta t)$ and $I[\vec{A}] : \mathbb{R}^n \to \mathbb{R}$ represents an interpolation operator defining, for every point $x$, the polynomial reconstruction based on the values $\vec{A}$. Unless stated otherwise $G$ is taken to be a regular equidistant array of points with mesh spacing $\Delta x_j$ along the $j$th axis, $j = 1, ..., n$.[4] We use a multilinear interpolator for the interpolation scheme, thus the interpolation function $I[\vec{A}](\cdot)$ is given by a convex combination over the elements of $\vec{A}$,

$$I[\vec{A}](x) = \phi(x) \cdot \vec{A} \ , \tag{2.9}$$

where $\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n_G}$, and $\forall x \in \mathcal{R}^n$ the elements of $\phi(x)$ are nonnegative and sum to 1.

The individual minimax games being played at each grid point can be collectively thought of as a game over policies $\pi(\cdot)$ and $\rho(\cdot)$. We also introduce the *backup operator* $B[\cdot] : \mathbb{R}^{n_G} \to \mathbb{R}^{n_G}$, which maps vectorized value functions onto themselves. Define the backup operator as

$$B[\vec{A}] := \min\{\vec{l}, \max_{\pi} \min_{\rho} \Phi_{\pi,\rho}\vec{A}\} \ , \tag{2.10}$$

where $\vec{l} \in \mathbb{R}^{n_G}$ with $\vec{l}_i = l(x_i)$ for $i = 1, ..., n_G$, $\Phi_{\pi,\rho} \in \mathbb{R}^{n_G \times n_G}$ is a policy-dependent stochastic matrix[5], and row $i$ of $\Phi_{\pi,\rho}$ is $\phi(x_i + \Delta t f(x, \pi(x_i), \rho(x_i)))$. For the one player setting this matrix becomes $\Phi_\pi$. We can now express (2.8) more concisely as

$$\vec{V}^0 = \vec{l} \ , \tag{2.11a}$$
$$\vec{V}_i^{k+1} = B[V^k] \ , \tag{2.11b}$$
$$\vec{V} = \lim_{k\to\infty} \vec{V}^k \ . \tag{2.11c}$$

This recursive procedure (2.11) is a dynamic programming algorithm referred to as *value iteration*, and here $\vec{V}$ is the *vectorized value function*, which is used to approximate the value function $V(x)$ as $I[\vec{V}](x)$. As we increase the fineness of the grid on a fixed domain, i.e. use more grid points, the accuracy of the approximation

[4] In the most general case the control and disturbance sets are also discretized, $\mathcal{U} = \{u_i\}_{i=1}^{n_\mathcal{U}}$ and $\mathcal{D} = \{d_i\}_{i=1}^{n_\mathcal{D}}$, and the minimax game is approximated.

[5] $\Phi_{\pi,\rho}$ is effectively the probability transition matrix for a Markov Decision Process (MDP) over a finite state space given by the grid nodes $\{x_i\}_{i=1}^{n_G}$.

improves. However, the computational effort also increases. Furthermore, for a desired level of accuracy the number of grid points, and thus computation, increases exponentially with the dimension of the system state, which is the well known *curse of dimensionality.*

The value iteration algorithm in (2.11) can be used to solve other optimal control problems, albeit with a different backup operator and initialization $\vec{V}^0$ [5], [12], [34]. Later, as we introduce other optimal control problems, including a novel formulation for reachability analysis, we will redefine the backup operator and specify the initialization required for the value iteration procedure.

### 2.2.4 Invariance Properties of Level Sets

Traditionally, the implicit hypothesis made to guarantee safety using a least-restrictive law in the form of (3.1) has been correctness of the disturbance bound $\mathcal{D}$ everywhere in the state space, (i.e. $d(x) \in \mathcal{D} \ \forall x \in \mathbb{R}^n$), or at least everywhere in the constraint set $\mathcal{K}$ [62], [43]. We will show that the necessary hypothesis for safety is in fact much less stringent, by proving an important result that we will use later to retain safety guarantees under partially incorrect models.

**Proposition 3** *Any nonnegative superlevel set of $V(x)$ is a robust controlled invariant set with respect to $d \in \mathcal{D}$.*

**Proof 1** *By Lipschitz continuity of $f$ and $l$, we have that $V$ is Lipschitz continuous [33] and hence, by Rademacher's theorem, almost everywhere differentiable. The convergence of $V(x,t)$ to $V(x)$ as $T \to \infty$ implies that at the limit $\frac{\partial V}{\partial t}(x,t) = 0$. Therefore, given any $\alpha \geq 0$, for any point $x \in \{x \mid V(x) \geq \alpha\}$ there must exist a control action $u^*$ such that $\forall d \in \mathcal{D}$, $\frac{\partial V}{\partial x}(x) f(x, u^*, d) \geq 0$; otherwise the right hand side of* (2.6a) *would be strictly negative for $T \to \infty$, contradicting convergence. Then, the value of $V$ from any such state $x$ can always be kept from decreasing, so $\{x | V(x) \geq \alpha\}$ is a robust controlled invariant set with respect to $d \in \mathcal{D}$.*

**Proposition 4** *Consider two disturbance sets $\mathcal{D}_1$ and $\mathcal{D}_2$, and a closed set $\mathcal{M} \subset \mathbb{R}^n$ that is robustly controlled invariant under $\mathcal{D}_1$. If $\mathcal{D}_2 \subseteq \mathcal{D}_1 \ \forall x \in \partial\mathcal{M}$, then $\mathcal{M}$ is robustly controlled invariant also under $\mathcal{D}_2$.*

**Proof 2** *Consider an arbitrary trajectory $\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}$ under the disturbance set $\mathcal{D}_2$, starting at $x_0 \in \mathcal{M}$, such that for some $\tau < \infty$, $\xi(\tau) \notin \mathcal{M}$. Since the trajectories are continuous, there must then exist $s \in [t_0, \tau]$ such that $\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}(s) \in \partial\mathcal{M}$. On the other hand, because $\mathcal{M}$ is robustly controlled invariant under $\mathcal{D}_1$, we know that $\exists \pi : \mathbb{R}^n \to \mathcal{U}$ such that no possible disturbance $d \in \mathcal{D}_1$ can drive the system out of $\mathcal{M}$. Since $\mathcal{D}_2 \subseteq \mathcal{D}_1 \ \forall x \in \partial\mathcal{M}$, the same control policy $\pi^*(x)$ on the boundary guarantees that no disturbance $d \in \mathcal{D}_2 \subseteq \mathcal{D}_1$ can drive the system out of $\mathcal{M}$. Hence, for $\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}$, switching to policy $\pi$ at time $s$ guarantees that the system will remain in $\mathcal{M}$. Therefore $\mathcal{M}$ is a robust controlled invariant set under $\mathcal{D}_2$.*

**Corollary 1** *Let* $\mathcal{Q}_\alpha = \{x \in \mathbb{R}^n : V(x) = \alpha\}$ *with* $\alpha \geq 0$ *be any nonnegative level set of the safety function* $V$*, computed for some disturbance set* $\hat{\mathcal{D}}$*. If* $d(x) \in \mathcal{D}\ \forall x \in \mathcal{Q}_\alpha$*, then the superlevel set* $\{x \in \mathbb{R}^n : V(x) \geq \alpha\}$ *is an invariant set under the computed safe control policy* $\pi^*(x)$*.*

# Chapter 3

# Safe Learning

*This chapter is adapted from the material presented in [2], [35].*

Learning-based methods in control have been around since at least the 1950s and 60s with the advent of adaptive control for the design of autopilots in high performance aircraft (see [32] for references). In the last decade, learning-based methods have seen a huge resurgence in interest and popularity due in particular to promising results of deep reinforcement learning schemes in virtual environments such as arcade videogames [66] and physics simulators [78]. These techniques show great promise for robotic applications, in which complex dynamics (e.g. contact forces) and hard-to-model environments (e.g. friction) limit the effectiveness of purely model-based approaches. Unfortunately, the inner-workings of many machine learning algorithms are difficult to interpret (notably in the case of deep neural networks), which makes it challenging to reach meaningful conclusions about the behavior of the system during the learning process, especially prior to convergence of the policy. This may not be a significant issue in simulation, but it poses real challenges when applying such techniques to physical systems in the real world. Lack of guarantees on the behavior of the underlying system may lead to failures, such as collisions or lack of stability, which in the best case hinder the learning process, and in the worst-case may result in material loss or injury. An example of such a failure occurred in 1967 with the NASA flight test X-15 Flight 191; an adaptive control scheme running on the aircraft began a limit-cycle oscillation, which resulted in high pitch gains and made it impossible for the pilot to pitch the aircraft out of a dive [50]. As the aircraft experienced rapidly increasing dynamic pressures it broke apart ending in the death of the pilot [50]. We refer to systems in which certain failure states are unacceptable as *safety-critical*, and it is on these systems that our work will focus.

In the last decade, learning-based control schemes have been successfully demonstrated in robotics applications in which the safety-critical aspects were effectively removed or mitigated, typically by providing a manual fallback mechanism or retrofitting the environment to allow safe failure. In [1], [27] a trained pilot was able to remotely take over control of the autonomous helicopter at any time; the power slide car maneuvers in [55] were performed on an empty test track; and the aerobatic quadrotor in [59] was enclosed in a safety net. While mostly effective, these *ad hoc* methods tend to come with their own issues (pilot handoffs, for instance, are notoriously prone to result in accidents [45]) and do not generalize well beyond the context of the particular demonstration. It is therefore necessary to develop principled and provably correct approaches to safety, attuned to the exploration-intense needs of learning-based algorithms, that can be built into the autonomous operation of learning robotic systems.

Current efforts in policy transfer learning propose training an initial control policy in simulation and then carrying it over to the physical system [26]. While progress in this direction is likely to reduce overall training time, it does not eliminate the risk of catastrophic system misbehavior. State-of-the art neural network policies have been shown to be vulnerable to small changes between training and testing conditions [49], which inevitably arise between simulated and real systems. Guaranteeing correct behavior of simulation-trained schemes in the real world thus remains an important unsolved problem.

Providing guarantees about a system's evolution inevitably requires some form of knowledge about the causal mechanisms that govern it. Fortunately, in practice it is never the case that the designer of a robotic system has no knowledge whatsoever of its dynamics: making use of approximate knowledge is both possible and, we argue, advantageous for safety.

## 3.1 Related Work

Early proposals of safe learning date back to the turn of the century. Lyapunov-based reinforcement learning [70] allowed a learning agent to switch between a number of pre-computed “base-level” controllers with desirable safety and performance properties; this enabled solid theoretical guarantees at the expense of substantially constraining the agent's behavior; in a similar spirit, later work has considered constraining policy search to the space of stabilizing controllers [74].

In risk-sensitive reinforcement learning [40], the expected return was heuristically weighted with the probability (risk) of reaching an “error state”; while this allowed for more general learning strategies, no guarantees could be derived from the heuristic effort. Nonetheless, the ideal problem formulation proposed in the paper, to maximize performance subject to some maximum allowable risk, inspired later work (see [39] for a survey) and is very much aligned with our own goals.

More recently, [67] proposed an ergodicity-based safe exploration policy for Markov decision processes (MDPs) with uncertain transition measures, which imposed a constraint on the probability, under the current belief, of being able to return to the starting state. While practical online methods for updating the system's belief on the transition dynamics are not discussed, and the toy grid-world demonstrations fall short of capturing the criticality of dynamics in many real-world safety problems, the probabilistic safety analysis is extremely powerful. Later safe exploration methods have assumed fully known deterministic dynamics with uncertain safety features [87], or allowed for uncertain dynamics but restricted the class of constraints to local stability (i.e. region of attraction) [17].

The robust model-predictive control approach in [8] learns about system dynamics for performance only, while enforcing constraints based on a robust nominal model. The method was successfully demonstrated on problems with nontrivial dynamics, including quadrotor flight. However, using an *a priori* model for safety at best constrains the system's ability to explore, and at worst may fail to keep the real system safe.

The model uncertainty can be treated explicitly as a disturbance, and the safety problem can be studied as a differential game, in which the controller must keep the system within the specified state constraints (i.e. away from failure states) in spite of the actions of an adversarial disturbance. Thus reachability analysis provides a suitable framework for learning in safety-critical systems. This robust, worst-case analysis determines a safe region in the state space and a control policy to remain inside it; a related approach involves ensuring invariance through "barrier functions" [7], [71], [81]. A key advantage is that in the interior of the safe region one can execute any desired action, as long as the safe control is applied at the boundary: in this sense, the technique yields a *least-restrictive* control law, which naturally lends itself to minimally constrained learning-based control. Initial work exploring this was presented in [42], [43], and this is the approach most similar to our own.

## 3.2 System Model Revisited

Moving forward the system dynamics are considered to be deterministic yet not completely modeled, though the unmodeled components are bounded. This constitutes the simplest class of systems for which learning is necessary. In this setting $d$ is a *deterministic* state-dependent disturbance capturing unmodeled dynamics, given by an unknown Lipschitz function $d : \mathbb{R}^n \to \mathcal{D}$. That is, we could in principle write the unknown dynamics as $F(x,u) = f\big(x,u,d(x)\big)$. Unlike $F$, $f$ is a known function, with all uncertainty captured by $d(\cdot)$.

## 3.3 Reachability-based Safe Learning

Learning-based control aims to achieve desirable system behavior by autonomously improving a policy $\pi_l : \mathbb{R}^n \to \mathcal{U}$, typically seeking to optimize an objective function. Safe learning additionally requires that certain constraints $\mathcal{K}$ remain satisfied while searching for such a policy. The main insight behind reachability-based safe learning is to separate the task of safety from the task of learning. Let us introduce an important notion from robust control theory.

**Definition 3** *A subset $\mathcal{M} \subset \mathbb{R}^n$ is a* robust controlled invariant set *under uncertain dynamics $\dot{x} = f(x, u, d)$, $d \in \mathcal{D}$, if for every point $x \in \mathcal{M}$ there exists a feedback control policy $\pi : \mathbb{R}^n \to \mathcal{U}$ such that, for all possible realizations, the closed-loop system trajectory satisfies $\xi(t) \in \mathcal{M}$ for all time $t \geq 0$.*

If we can find such a set and policy, then the safety problem is solved, and this is precisely what we get with the safe set $\Omega(\mathcal{K})$ and the optimal safe policy $\pi^*(\cdot)$.

Now to address the learning. Given that trajectories are continuous, the system state can only leave a controlled invariant set $\mathcal{M}$ by crossing its boundary $\partial\mathcal{M}$. Hence if $\mathcal{M}$ is closed, applying the feedback policy $\pi^*(x)$ for $x \in \partial\mathcal{M}$ is enough to render $\mathcal{M}$ robust controlled invariant, allowing an arbitrary control action to be applied in the interior of $\mathcal{M}$, which includes learning-based control actions. This leads to a least-restrictive control law that is guaranteed to satisfy the state constraints $\mathcal{K}$ while allowing for any action to be applied on the interior of the safe set. This least-restrictive control law can be used in conjunction with an arbitrary learning-based control policy $\pi_l$ (which may be repeatedly updated by the corresponding learning algorithm), to produce a *safe learning policy*:

$$\pi(x) = \begin{cases} \pi_l(x), & \text{if } V(x) > 0, \\ \pi^*(x), & \text{otherwise.} \end{cases} \tag{3.1}$$

### 3.3.1 Quadrotor Testbed

Throughout this thesis we will demonstrate various methods with an autonomous quadrotor helicopter learning a flight controller in different scenarios. To avoid redundancy all experiments utilizing this testbed will refer back to this section.

These methods are tested on the Stanford-Berkeley Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC), using Ascending Technologies Pelican (Fig. 3.1) and Hummingbird quadrotors (Fig. 4.4). The system receives full state feedback from a VICON motion capture system. The vehicle's dynamics are approximately decoupled through an on-board controller responsible for providing lateral stability around hover and vertical flight; our framework is then used to learn the feedback gains for a hybrid vertical flight controller. The learning and safety controllers were implemented and executed in MATLAB, on a Lenovo Thinkpad with an Intel i7 processor that communicated wirelessly with the vehicle's 1.99 GHz Quad-core Intel Atom processor. This was all done using the Indigo version of the Robot Operating System (ROS) framework.

Figure 3.1: A STARMAC quadrotor (Pelican) during the flight test.

Reachability computations are executed using the Level Set Toolbox [63], employing the Lax-Friedrich approximation for the numerical Hamiltonian; a weighted essentially nonoscillatory scheme for spatial derivatives; and a third-order total variation diminishing Runge-Kutta scheme for the time derivative [69], [80].

The purpose of the results presented on this testbed is not to advance the state of the art of quadrotor flight control or reinforcement learning techniques, but to illustrate how the proposed method can allow safe execution of an arbitrary learning-based controller without requiring any particular convergence rate guarantees.

We use an affine dynamical model of quadrotor vertical flight, with state equations:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= k_T u + g + k_0 + d(x) \end{aligned} \tag{3.2}$$

where $x_1$ is the vehicle's altitude, $x_2$ is its vertical velocity, and $u \in [0, 1]$ is the normalized motor thrust command. The gravitational acceleration is $g = -9.8 \text{ m/s}^2$. The parameters of the affine model $k_T$ and $k_0$ are determined for the Pelican and the Hummingbird vehicles through a simple on-the-ground experimental procedure—a scale is used to measure the normal force reduction for different values of $u$. Finally, $d$ is an unknown, state-dependent scalar disturbance term representing unmodeled forces in the system. The state constraint $\mathcal{K} = \{x : 0 \text{ m} \leq x_1 \leq 2.8 \text{ m}\}$ encodes the position of the floor and the ceiling, which must be avoided.

As the learning-based controller, we choose an easily implementable policy gradient reinforcement learning algorithm [56], which learns the weights for a linear

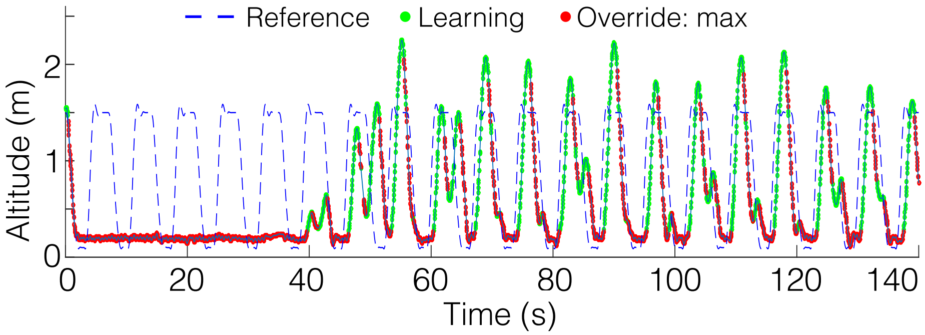


Figure 3.2: Vehicle altitude and reference trajectory over time. Initial feedback gains are set to zero. When the learning controller (green) lets the vehicle drop, the safety control (red) takes over preventing a collision. Within a few seconds, the learned feedback gains allow rough trajectory tracking and are subsequently tuned as the vehicle attempts to minimize error.

mapping from state features to control commands. Following [43], we define different features for positive and negative velocities and position errors, since the (unmodeled) rotor dynamics may be different in ascending and descending flight. This can be seen as the policy gradient algorithm learning the feedback gains for a hybrid proportional-integral-derivative (PID) controller.

### 3.3.2 Experiment: Learning to Fly from "Scratch"

*This experiment uses the quadrotor testbed described in Section 3.3.1.*

To demonstrate the strength of Hamilton-Jacobi-based guarantees for safely performing learning-based control on a physical system, we first require a Pelican quadrotor to learn an effective vertical trajectory tracking controller with an arbitrarily poor initialization. To do this, the policy gradient algorithm is initialized with all feature weights set to 0. The pre-computed safety controller (numerically obtained using [63]) is based on a conservative uncertainty bound of $\pm 1.5$ m/s$^2$ everywhere in the state space. The reference trajectory requires the quadrotor to aggressively alternate between hovering at two altitudes, one of them (1.5 m) near the center of the room, the other (0.1 m) close to the floor. This experiment illustrates the interplay between the *learning controller* and the *safety policy*, using the least-restrictive safe policy given by equation (3.1).

The experiment, shown in Fig. 3.2, is initialized with the vehicle in mid-air. Since all feature weights are initially set to zero, the vehicle's initial action is to enter free fall. However, as the quadrotor is accelerated by gravity towards the floor, the boundary of the computed safe set is reached, triggering the intervention of the safety controller, which automatically overrides the learning controller and commands the maximum available thrust to the motors ($u = 1$), causing the vehicle to decelerate and hover at a small distance from the ground. For the next few seconds, there is some chattering near the boundary of the safe set, and the policy gradient algorithm has some occasions to attempt to control the vehicle when it is momentarily pushed into the interior of the safe set. Initially it has little success, which leads the safety controller to continually intervene to prevent the quadrotor from colliding with the floor; this has the undesirable effect of slowing down the learning process, since observations under this interference are uninformative about the behavior of the vehicle when actually executing the commands produced by the learning controller (which is an "on-policy" algorithm). However, at approximately $t = 40$ s, the learning controller is able to make the vehicle ascend towards its tracking reference, retaining control of the vehicle for a longer span of time and accelerating the learning process. By $t = 60$ s, the quadrotor is approximately tracking the reference, with the safety controller only intervening during the aggressive descent phase of the repeated trajectory, to ensure (under its conservative model) that there is no risk of a ground collision. The controller continues to learn in subsequent iterations, overall improving its tracking accuracy.

The remarkable result in this experiment is not in the quality of the learned tracking controller after only a few seconds of active exploration (a merit that corresponds to the reinforcement learning method [56]), but the system's ability to achieve competent performance at its task from an extremely poor initial policy while remaining safe at all times.

## 3.4 Integrating Safety Cost

From the control law given by (3.1), the action applied in the interior of the safe set is specified by a learning algorithm that has some performance objective, which typically does include safety [42], [43]. This causes the reachability analysis and the learning algorithm to be disjoint with respect to one another, thus switching control from the learning algorithm can cause the system to chatter or stall (which is seen in the experiment from Section 3.3.2).

Furthermore, RL algorithms use the feedback obtained from the environment to specify the next action that should be taken, which seeks to converge to or execute an optimal policy [54]. This is particularly critical for on-policy algorithms, which require that the specified learning control be the one executed by the system [65], [75], [84]. Therefore, failing to incorporate safety metrics in the learning algorithm can degrade the convergence and performance. The objective here is to unify learning and safety by incorporating safety metrics in the learning.

One way to unify the two objectives is to create an algorithm that switches less frequently between the safe control and learning control, or equivalently minimize the number of times the system reaches the boundary of the safe set. On the interior of the safe set, the safety function $V(x)$ can be viewed as a measure of how far a state is from reaching the boundary of the set. The safety function can thus be used as a cost in the learning algorithm to discourage the system from reaching the boundary. This idea will be exemplified with the learning algorithm Policy Gradient via the Signed Derivative (PGSD) from [56].

### 3.4.1 PGSD with Safety Cost

PGSD is a model-free policy search algorithm. We present the algorithm here briefly and refer the reader to [56] for a more detailed description. In policy gradient learning algorithms, a parametrized control policy is updated in order to optimize a given cost function over the state-action pair, $C(x, u)$. As an example, we consider a quadratic cost and a control that is linear in the state features

$$C(x,u) = \frac{1}{2}(x - x^*)^\top Q(x - x^*) + \frac{1}{2}u^\top Ru, \qquad u = \Theta\phi(x). \tag{3.3}$$

where $x^*$ is the desired state, $Q$ and $R$ are diagonal positive semidefinite matrices penalizing deviation from $x^*$ and control input respectively, $\phi(x) \in \mathbb{R}^k$ is a vector of features, each of which maps the state to a scalar value, and $\Theta \in \mathbb{R}^{m\times k}$ is a matrix of weights that linearly map these features into controls. The controller's objective is to minimize the cost incurred over a horizon $H$ starting at $x_0$ while applying $\Theta$:

$$J(x_0, \Theta) = \sum_{t=1}^{H} C(x_t, u_t), \qquad u_t = \Theta\phi(x). \tag{3.4}$$

For simplicity of notation we omit the arguments of the cost in the following. PGSD updates the parameters as:

$$\Theta \leftarrow \Theta - \alpha\nabla_\Theta J \tag{3.5a}$$

$$\nabla_\Theta J = \frac{1}{H}\sum_{t=0}^{H-1}(\nabla_{u_t} J)\phi(x_t)^\top \tag{3.5b}$$

$$\nabla_{u_t} J \approx \sum_{t'=t+1}^{H} S^\top Q(x_{t'} - x^*_{t'}) + Ru_{t'}, \tag{3.5c}$$

where $\alpha > 0$ is the step size, and $S_{i,j}$ is the sign of $\frac{\partial(x_{t'})_i}{\partial(u_t)_j}$ with the additional restriction that only one element in each row of $S$ is nonzero corresponding to the control that has the largest effect on that given state. The safety metric can be included as follows:

$$C_S(x,u) = C(x,u) - \chi\log(V(x)), \tag{3.6}$$

where $\chi$ is a weighting factor. The log barrier function goes to infinity at the boundary of the critical level set, and is approximately constant in the interior. Equation (3.5c) then becomes

$$\nabla_{u_t} J = \sum_{t'=t+1}^{H} S^\top Q(x_{t'} - x^*_{t'}) - \frac{\chi S^\top \nabla_x V(x_{t'})}{V(x_{t'})} + R u_{t'}. \tag{3.7}$$

The gradient $\nabla_x V(x)$ is calculated from (2.6) during the reachability computations.

### 3.4.2 Experiment: Cart-Pole Swing Up with Safety Metric Intgration

In this simulation we highlight the benefits of including a safety metric in the learning algorithm. We implemented the PGSD algorithm with and without the safety metric on a simulated cart-pole system with one actuator controlling the thrust of the cart. The states of the system are $x = \begin{bmatrix} p & \dot{p} & \theta & \dot{\theta} \end{bmatrix}^\top$; position, velocity, clockwise angle (pendulum up is the origin), and angular velocity. The dynamics are

$$\ddot{p} = \frac{(u-d) + h_1 \text{cos}(\theta)\text{sin}(\theta) + h_2 \text{sin}(\theta)(\dot{\theta})^2}{h_3 + h_4 \text{cos}^2(\theta)}, \tag{3.8}$$

$$\ddot{\theta} = \frac{g_1 cos(\theta)(u-d) + g_2 \text{cos}(\theta)\text{sin}(\theta)(\dot{\theta})^2 + g_3 \text{sin}(\theta)}{g_4 + g_5 \text{cos}^2(\theta)}, \tag{3.9}$$

where $h_1, \ldots, h_4$ and $g_1, \ldots, g_5$ are physical constants, $\mathcal{U} = [-25 \text{ N}, 25 \text{ N}]$ is the set of valid thrust controls, and $\mathcal{D}(x)$ is uniformly bounded over the entire state space by $[-5 \text{ N}, 5 \text{ N}]$.

The task is for the pendulum to swing itself up and stabilize at a fixed point $p^* = 0.25$ meters, while never leaving the track, defined by $\mathcal{K} = \{x : -0.5 \text{ m} \leq p \leq 0.5 \text{ m}\}$. The goal is to track the reference $x^* = \begin{bmatrix} p^* & 0 & 0 & 0 \end{bmatrix}^\top$, starting at the initial condition $x_0 = \begin{bmatrix} 0 & 0 & \pi & 0 \end{bmatrix}^\top$.

The feature vector, $\phi(x)$, now contains two sets of features. One set of features is made inactive (set equal to zero) when the pole is above the horizon ($\frac{\pi}{2} < |\theta|$), and the other set becomes inactive when the pendulum is below the horizon. The first set of features includes the error in each state, as well as the absolute position (five features in total). The second set of features contains the error in the position and velocity, the absolute position, and two disjoint features for the angle error that are inactivated depending on whether the sign of the angular velocity is the same as the sign of the angle error (five features in total). The absolute position acts as a safety feature that moves the cart away from the ends of the tracks. As for the cost function $R = 0, Q_{1,1} = 1, Q_{3,3} = 2$, and $Q_{2,2} = Q_{4,4} = 0$. For PGSD with safety $\chi = 0.002$.

PGSD was run with and without the proposed safety metric in equation (3.6) for 100 seconds. The result can be seen in Fig. 3.3. Without the safety metric the controller switches more often, and it takes a longer time for the task to be completed.

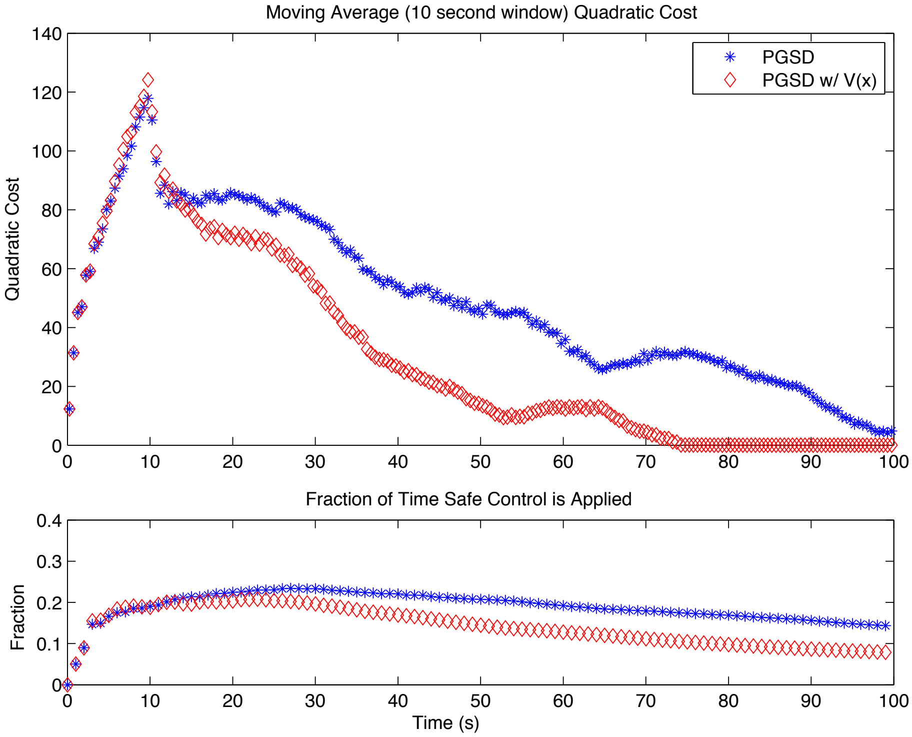


Figure 3.3: Top: The pendulum learns the swing-up task faster when the safety metric is included in the learning. Bottom: Incorporating the safety metric reduces the intervention of the safety controller.

## 3.5 Model Validation

The paradigm in reachability analysis is to assume that the system dynamics is captured by the model used in the differential game, in other words the assumption is that the unknown portion of the dynamics $d(x)$ belongs to $\mathcal{D}$ for all $x$. Under this assumption all the safety guarantees presented thus far hold true, and $\mathcal{D}$ is typically designed such that the assumption holds. However, once the least-restrictive control law is deployed on the system practitioners do not typically check to see whether the assumption does in fact hold. Given that we are merging model-based techniques (reachability) with data-driven methods (learning) two ideas come to mind. First, can the data being collected online be used to modify the model and ultimately the safe set and control law used to ensure safety. Second, can the data be used to validate whether the current model is sufficient to guarantee safety. In Chapter 4 we will consider the first idea. In this section we consider the second, and provide a modified control strategy to handle cases when the model may potentially be invalid.

Since the unknown part of the dynamics is represented by $d(x)$, $\mathcal{D}$ can effectively be thought of as representing the uncertainty that we have about the system. This uncertainty can vary across the state space, in which case it would be better to consider state dependent disturbance sets $\mathcal{D}(x)$, where $\hat{\mathcal{D}} : \mathbb{R}^n \to 2^{\mathcal{D}}$ is a set-valued map. As one might expect, in this setting safety is guaranteed if $d(x) \in \mathcal{D}(x)$ for all $x$. However, the reachability formulation must be tweaked slightly and the disturbance sets $\mathcal{D}(x)$ must satisfy certain regularity conditions. If $\mathcal{D}(x)$is compact and Lipschitz continuous in the Hausdorff metric[1] then the differential game is still well-defined and all of the propositions and corollaries in Section 2 hold true (replacing $\mathcal{D}$ with $\mathcal{D}(x)$) [35].

### 3.5.1 Model Reliability Margin

We further restrict our system dynamics to only have additive unknowns, which also includes the quadrotor model in Section 3.3.1.

$$f(x, u, d(x)) = g(x, u) + d(x), \tag{3.10}$$

where $g : \mathbb{R}^n \times \mathbb{R}^{n_u} \to \mathbb{R}^n$ is a locally Lipschitz continuous function representing the known part of the dynamics, and $d : \mathbb{R}^n \to \mathbb{R}^n$ is an unknown, but deterministic, state dependent disturbance capturing unmodeled dynamics.

Since the disturbance is additive, we can construct $d(x)$ for any state-input pair that has been visited as the residual between the observed dynamics and the model's prediction:

$$\hat{d}(x) = \hat{f}(x, u) - g(x, u), \tag{3.11}$$

[1] The Hausdorff metric (or Hausdorff distance) between any two sets $A$ and $B$ in a metric space $(M, d_M)$ is defined as $d_H(A, B) = \max\{\, \sup_{a\in A} \inf_{b\in B} d_M(a, b),\, \sup_{b\in B} \inf_{a\in A} d_M(a, b)\,\}$.

where $\hat{f}(x,u)$ is an approximation of the state derivative (for example through numerical differentiation). These online measurements of the system's evolution can be used to monitor the local value of the disturbance $d(x)$ in real time using (3.11). We define the *model reliability margin* $\lambda(x)$ as:

$$\lambda(x) = \frac{s_{\mathcal{D}(x)}\big(d(x)\big)}{\min\limits_{\delta \in \mathcal{D}(x)} s_{\mathcal{D}(x)}\big(\delta\big)}, \tag{3.12}$$

The model reliability margin is therefore a normalized signed distance function, with range [0,1] on the inside of $\mathcal{D}(x)$ and negative outside, which provides a metric for confidence in the model along the system's trajectory. Obtaining a measurement with small $\lambda(x) > 0$ indicates that the model is performing poorly, but its bound on the disturbance (on which reachability guarantees are based) is still correct locally. Conversely, if $\lambda(x) < 0$, then all theoretical safety guarantees are lost (the disturbance is playing an unexpected value for which the control action is not guaranteed to win the differential game).

We propose a new control strategy that takes into account the model reliability:

$$V_L = \begin{cases} \max(V(x), V_L) & \text{if } \lambda(x) \le \lambda_L \\ V_L, & \text{otherwise} \end{cases} \tag{3.13a}$$

$$\pi(x) = \begin{cases} \pi_l(x) & \text{if } V(x) - V_L > 0 \\ \pi^*(x), & \text{otherwise} \end{cases}, \tag{3.13b}$$

where the *critical safety level* $V_L$ is initialized to 0, and $\lambda_L \in (0,1)$ is a predefined threshold; the criteria to select its value will be discussed later in this section. We refer to the compact set $\{x : V(x) - V_L > 0\}$ as the *critical level set*; note that this set is initialized to $\Omega(\mathcal{K})$, and updated by the control strategy when the model reliability margin reaches $\lambda_L$. The purpose of the above control strategy is to keep the system within the critical level set. Effectively, the algorithm shrinks the allowed region of operation by pruning away those states potentially admitting disturbances that are not captured by the model. The principle by which the new allowed operating region is chosen to be the critical level set is based on the following result.

**Proposition 5** *If a state $z$ is reached such that $\lambda_L \geq \lambda(z) > 0$ and $\exists V_S \in [0, V(z)]$ such that $d(x) \in \mathcal{D}(x)$ $\forall x \in \{\, x : V(x) = V_S \,\}$, then the strategy given by* (4.11) *is guaranteed to keep the system safe.*

**Proof 3** *Since $\forall x \in \{\, x : V(x) = V_S \,\}$, $d(x) \in \mathcal{D}(x)$, by Corollary 1 the level set associated with $V_S$ is indeed a controlled invariant set of the true system under control policy $u^*(x)$. Since the control strategy given by* (3.13) *will apply the control $u^*(x)$ for all $x$ such that $V(x) \leq V_L$, then the system cannot leave the level set $\{x : V(x) \geq V_S\}$, and therefore safety is guaranteed.*

Note that the conditions for guaranteeing safety under the control strategy in (3.13) are much less stringent than those required by current reachability-based safety algorithms. In order to preserve safety, these frameworks require that the estimated disturbance set capture the true disturbance at least on the boundary of the computed discriminating kernel. Conversely, our algorithm guarantees safety as long as there exists *some* super-zero level set $\{x : V(x) = V_S\}$, $0 \leq V_S \leq V_L$ of the computed safety function $V(x)$ such that the disturbance is captured by the computed disturbance set on the boundary of the said set. In a continuous state space, there is an infinite number of candidate level set boundaries and it suffices that one of them satisfies the condition for safety guarantees to hold.

It is important to note that the choice of $\lambda_L$ determines the conservativeness of the control strategy: a larger value of $\lambda_L$ leads the algorithm to start applying the safe control policy $u^*(x)$ for smaller deviations in the measured disturbance with respect to its expected value as predicted by the GP. The likelihood of there existing a candidate level set boundary where the disturbance lies within the specified bounds is therefore larger for larger values of $\lambda_L$, but the algorithm will also be more sensitive to modeling error and become more restrictive for smaller inconsistencies with observations.

Moreover, Proposition 5 provides a *sufficient* condition for safety when implementing this proposed control strategy. Safety may still be provided even when its premises do not hold.

### 3.5.2 Experiment: Online Model Validation for Safety

*This experiment uses the quadrotor testbed described in Section 3.3.1.*

To illustrate the strength of the new control strategy (3.13), we consider an experiment, in which we compare its behavior to that of the standard reachability-based safety framework with no online validation when presented with an initial model that fails to account for the true disturbance. The model reliability threshold $\lambda_L$ is set to 0.1. For the sake of fairness, we define a trajectory that does not explicitly attempt to drive the system out of the state constraints (which in this case are re-defined to be $\mathcal{K} = \{x : 0.2 \text{ m} \leq x_1 \leq 2.8 \text{ m}\}$, 0.2 m corresponding to the actual position of the body-frame origin when the base of the vehicle comes to physical contact with the ground). The initial model in this case assumes a prior bound on $d(x)$ which is 10 times smaller than in the previous experiment. Both algorithms begin with the same initial safe set $\Omega_0$. Once the test begins, the standard algorithm breaches the computed safe set on several occasions and violates the constraints incurring two consecutive ground collisions, marked in Fig. 3.4. Conversely, the proposed framework immediately detects a persistent unmodeled disturbance (in this case, $\lambda(x) < 0$) and consequently reduces the safe set to the current safety value, immediately applying the control action $\pi^*$: this constitutes the system's best effort for safety given its current knowledge of the system, even when the sufficient conditions from Proposition

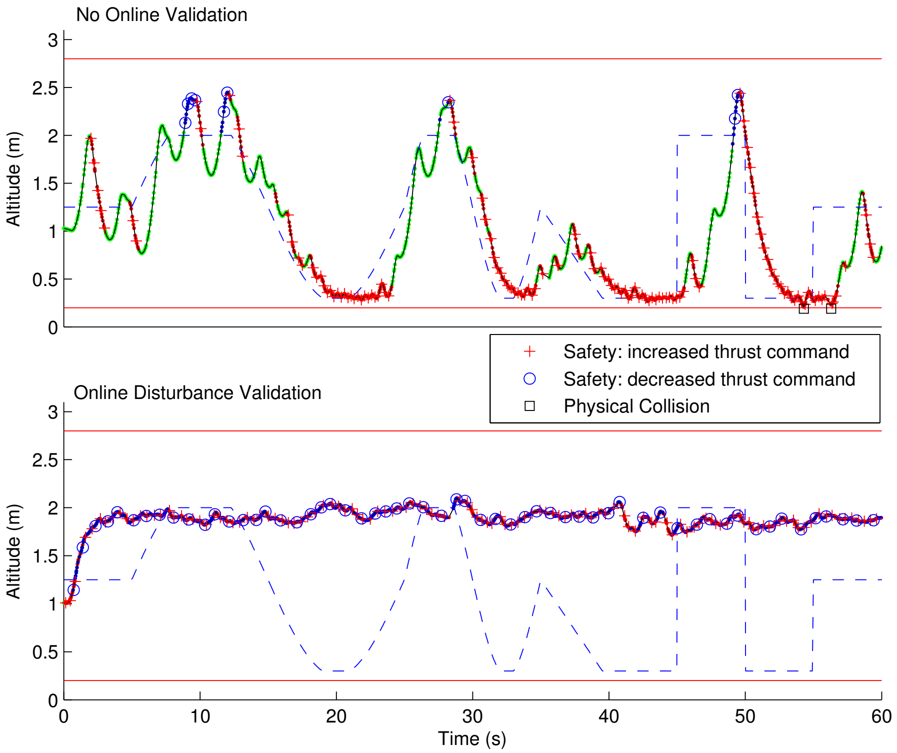


Figure 3.4: Altitude trajectory of the quadrotor with an incorrect disturbance model. The standard reachability-based safety algorithm attempts to follow the trajectory, incurring several safety breaches and two physical collisions. The proposed algorithm rapidly detects the inconsistency between model and observations and retracts to the region with the highest computed safety value.

5 do not hold (since the model reliability margin $\lambda(x)$ is already negative to begin with). Since the measured disturbance keeps breaking the model's assumptions the system keeps contracting its safe set to higher level curves until it reaches what is, to the best of its knowledge, the safest grid cell in the state space. While the results in Fig. 3.4 may look quite restrictive, it should be noted that this is an extreme case where the model of the system is wrong everywhere. Under these conditions, temporarily restricting the vehicle's motion is generally preferable to a crash.

# Chapter 4

# Learning-Based Reachability

*This chapter is adapted from the material presented in [4], [35].*

So far we have proposed a framework for robustly guaranteeing safety for systems with state-dependent bounded uncertainty. A critical requirement for guaranteeing safety is the correctness of the disturbance bounds $\hat{\mathcal{D}}(x)$. These bounds are typically chosen conservatively resulting in smaller safe sets. Given the least restrictive control law (3.1) this can greatly reduce the performance of the system with respect to other objectives, e.g. learning a control law for tracking, since the system has less of the state space at its disposal.

In order to avoid excessive conservativeness and keep theoretical guarantees, it is essential to have a principled method to refine the system model based on acquired measurements, and a reliable mechanism to detect and react to model discrepancies with the real systems behavior; both of these components are inherently data-driven. We thus arrive at a very important insight: *the relation between safety and learning is reciprocal.* Not only is safety a key requirement for learning in autonomous systems, but learning about the real systems behavior is itself indispensable to provide practical safety guarantees. In particular, we are interested in how the system's behavior impacts the safe set.

The objective of this chapter is to develop data-driven methods for computing the safe set, or more accurately for computing the value function. There are two ways to do this: *model-based* and *model-free.*[1] Model-based techniques first fit a model using the data, and then compute the value function using the model; this is the approach taken by [2], [35], [43]. Model-free techniques modify the value function directly using the data, without explicitly computing a model, which is the approach taken by [4].

We present a model-based technique using Gaussian Processes[73] in Section 4.1,

[1] This distinction between model-based and model-free is taken from Reinforcement Learning, which also attempts to compute the value function (for a different optimal control problem) using data from the system.

and a model-free technique using temporal differencing[83] in Section 4.2.

## 4.1 Gaussian Processes for Modeling Additive Disturbances

*This section is an adaptation of Section IV.B of [35].*

Similar to Section 3.5.1 this section assumes a system with additive disturbance, so the dynamics are given by (3.10). Recall, the disturbance is used to capture the uncertainty in the system due to the fact that $d(x)$ is unknown. It is reasonable then to construct an approximation of the state-dependent disturbance set $\hat{\mathcal{D}}(x)$ so that it reflects this uncertainty. Here we do so using a Gaussian Process.

### 4.1.1 Gaussian Process

To estimate the disturbance function $d(x)$ over the state space, we model it as being drawn from a Gaussian process (GP). GPs are a powerful abstraction that extends multivariate Gaussian regression to the infinite-dimensional space of functions, allowing Bayesian inference based on (possibly noisy) observations of a function's value at finitely many points. We give here an overview of Gaussian process regression and direct the interested reader to [73] for a more comprehensive introduction.

A GP is a random process or field defined by a mean function $\mu : \mathbb{R}^n \to \mathbb{R}$ and a positive semidefinite covariance kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Each component, $d^j, j \in \{1, ..., n_d\}$, of the disturbance function is treated as an independent GP:

$$d^j(x) \sim \mathcal{GP}(\mu^j(x), k^j(x, x')). \tag{4.1}$$

A defining characteristic of a GP is that the marginal probability distribution of the function value at any finite number of points is a multivariate Gaussian. This will allow us to obtain the disturbance bound $\hat{\mathcal{D}}(x)$ as a Cartesian product of confidence intervals for the components of $d(x)$ at each state $x$, choosing the bound to capture a desired degree of confidence.[2]

GPs allow incorporating new observations in a nonparametric Bayesian setting. First, assume a prior GP distribution over the $j$-th component of $d(\cdot)$, with mean $\mu^j(\cdot)$ and covariance kernel $k^j(\cdot, \cdot)$. The class of the prior mean function and covariance kernel function is chosen to capture the characteristics of the model (linearity, periodicity, etc), and is associated to a set of hyperparameters $\theta_p$. These are typically set to maximize the marginal likelihood of an available set of training data, or possibly to reflect some prior belief about the system.

[2] By assuming independence of disturbance components we are effectively over-approximating the confidence ellipsoid in $\mathbb{R}^{n_d}$ by its minimal containing box; a less conservative analysis could compute $\hat{\mathcal{D}}(x)$ using a vector-valued GP model, at the expense of heavier computation.

Next, consider $N$ measurements $\hat{\mathbf{d}}^j = [\hat{d}^j_1, \ldots, \hat{d}^j_N]$, observed with independent Gaussian noise $\epsilon^j_i \sim \mathcal{N}(0, (\sigma^j_n)^2)$ at the points $X = [x_1, \ldots, x_N]$, i.e. $\hat{d}^j_i = d^j(x_i) + \epsilon^j_i$. Combined with the prior distribution (4.1), this new evidence induces a GP posterior; in particular, the value of $d^j$ at finitely many points $X_*$ is distributed as a multivariate normal:

$$\mathbb{E}[d^j(X_*) \mid \hat{\mathbf{d}}^j, X] = \tag{4.2a}$$

$$\mu^j(X_*) + K^j(X_*, X)(K^j(X, X) + (\sigma^j_n)^2 I)^{-1}(\hat{\mathbf{d}}^j - \mu^j(X)),$$

$$\mathrm{cov}[d^j(X_*) \mid X] = \tag{4.2b}$$

$$K^j(X_*, X_*) - K^j(X_*, X)(K^j(X, X) + (\sigma^j_n)^2 I)^{-1} K^j(X, X_*),$$

where $d^j_i(X) = d^j(x_i)$, $\mu^j_i(X) = \mu^j(x_i)$, and for any $X, X'$ the matrix $K^j(X, X')$ is defined component-wise as $K^j_{ik}(X, X') = k^j(x_i, x'_k)$. The hyperparameters of the kernel function are refitted whenever a new batch of data $X$ is obtained, so the variance implicitly depends on the measurements $d^j$.

Considering a single query point, i.e. $X_* = \{x_*\}$, the marginalized GP posterior becomes a univariate normal distribution quantifying both the expected value of the disturbance function, $\bar{d}^j(x_*)$, and the uncertainty of this estimate, $\big(\sigma^j(x_*)\big)^2$,

$$\bar{d}^j(x_*) = \mathbb{E}[d^j(x_*) | \hat{\mathbf{d}}^j, X] \tag{4.3a}$$

$$\big(\sigma^j(x_*)\big)^2 = \mathrm{cov}[d^j(x_*) | X] \ . \tag{4.3b}$$

We can use the Bayesian machinery of GP regression to compute a *likely* bound $\hat{\mathcal{D}}(x)$ on the disturbance function $d(x)$ based on the history of commanded inputs $u_i$ and state measurements $x_i$, $i \in \{1, ..., N\}$. Using (3.11), we can obtain measurements of $d(x_i)$ from the residuals between the observed dynamics and the model's prediction.

The residuals $\hat{\mathbf{d}} = [\hat{d}(x_1), \ldots, \hat{d}(x_N)]$ are processed through (4.2) to infer the marginal distribution of $d(x_*)$ for an arbitrary point $x_*$, specified by the expected value $\bar{d}^j(x_*)$ and the standard deviation $\sigma^j(x_*)$ of each component of the disturbance. This distribution can be used to construct a disturbance set $\hat{\mathcal{D}}(x_*) \subseteq \mathcal{D}$ at any point $x_*$; in practice, this will be done at finitely many points $x_{\mathbf{i}}$ on a grid, and used in the numerical reachability computation to obtain the safety function and the safe control policy.

We now introduce the design parameter $p$ as the desired marginal probability that the disturbance function $d(x)$ will belong to the bound $\hat{\mathcal{D}}(x)$ at each point $x$; typically, $p$ should be chosen to be close to 1. The set $\hat{\mathcal{D}}(x)$ is then chosen for each $x$ as follows. Let $z = \sqrt{2}\,\mathrm{erf}^{-1}(p^{1/n_d})$, where $\mathrm{erf}(\cdot)$ denotes the Gauss error function; that is, define $z$ so that the probability that a sample from a standard normal distribution $\mathcal{N}(0, 1)$ lies within $[-z, z]$ is $p^{1/n_d}$. We construct $\hat{\mathcal{D}}(x)$ by taking a Cartesian product of confidence intervals:

$$\hat{\mathcal{D}}(x) = \prod_{j=1}^{n_d} [\bar{d}^j(x) - z\sigma^j(x),\ \bar{d}^j(x) + z\sigma^j(x)]. \tag{4.4}$$

Since each component $d^j(x)$ is given by an independent Gaussian $\mathcal{N}\left(\bar{d}^j(x), \sigma^j(x)\right)$, the probability of $d(x)$ lying within the above hyperrectangle is by construction $\left(p^{1/n_d}\right)^{n_d} = p$.

**Remark 1** *It is commonplace to use Gaussian distributions to capture beliefs on variables that are otherwise known to be bounded. While one might object that the unbounded support of* (4.3) *contradicts our problem formulation (in which the disturbance $d$ took values from some compact set $\mathcal{D} \subset \mathbb{R}^{n_d}$), the hyperrectangle $\hat{\mathcal{D}}(x)$ in* (4.4) *is always a compact set. Note that the theoretical input set $\mathcal{D}$ is never needed in practice, so it can always be assumed to contain $\hat{\mathcal{D}}(x)$ for all $x$.*

Under Lipschitz continuous prior means $\mu^j$ and covariance kernels $k^j$, the disturbance bound (4.4) varies (Hausdorff) Lipschitz-continuously in $x$ [35]. The safety analysis described in Section 2.2 can be carried out by solving the Hamilton-Jacobi equation (2.6) replacing $\mathcal{D}$ with $\hat{\mathcal{D}}(x)$ given by (4.4), which will be a correct disturbance bound at any single state $x$ with probability $p$ based on the information available at the time of computation. As the system gathers new information, the posterior probability of $d(x) \in \hat{\mathcal{D}}(x)$ will change for each $x$ (and will typically no longer equal $p$). More generally, we have the following result.

**Proposition 6** *Let $q$ be the probability that $d(x) \in \hat{\mathcal{D}}(x)$ for some state $x$ with $V(x) \geq 0$. Then the probability that $L_f^* V(x) := D_x V(x) \cdot f\left(x, \pi^*(x), d(x)\right) \geq 0$ is at least $q$.*

***Proof 4*** *Omitting $x$ for conciseness, we have: $P(L_f^* V \geq 0) = P\big(L_f^* V \geq 0 | d \in \hat{\mathcal{D}}\big) P\big(d \in \hat{\mathcal{D}}\big) + P\big(L_f^* V \geq 0 | d \notin \hat{\mathcal{D}}\big) P\big(d \notin \hat{\mathcal{D}}\big)$.*

*By Corollary 1, the first term evaluates to $1 \cdot q$; the second term is nonnegative (and will typically be positive, since not all values of $d \notin \hat{\mathcal{D}}$ will be unfavorable for safety, and there may be some for which the input $\pi^*(x)$ leads the system to locally increase $V$).*

Based on this result, we can begin to reason about the guarantees of the reachability analysis applied to the real system in a Bayesian framework, inherited from the GP model. We examine this next.

### 4.1.2 Model Validation with Gaussian Processes

*This section is an adaptation of Section IV.C of [35].*

So far we have used the disturbance measurements for two purposes, in the previous subsection we use them in batches to construct the model, and in Section 3.5.1 individual measurements are used to validate the model. Let us consider how construction and validation come together in the larger context of the safe learning algorithm. Given a batch of data, e.g. $N$ measurements, a model can be constructed to perform the safety analysis. After the system is deployed the incoming stream of data is used to validate the current model. Once we have enough data to make up another batch (another $N$ measurements in this example), then a new model is constructed, and the process is repeated.

With the GP we can unify the model construction and model validation under a Bayesian framework. First, we describe the process at a high level. The GP uses a batch of data to create a posterior distribution on the disturbance function $d(x)$. The disturbance set $\hat{\mathcal{D}}(x)$ is then constructed to ensure that with probability $p$ it contains the disturbance function. Due to the Bayesian framework provided by the GP, as new measurements are acquired they can be incorporated into the posterior (which at this point can be thought of as the new prior) to produce a new posterior. This will also change the probability that $\hat{\mathcal{D}}(x)$ contains $d(x)$, and this new probability can be used to validate the model. In the remainder of this section, we will discuss how to update the belief on the disturbance function, and then provide two different theoretical criteria for safety intervention. The first criterion provides global probabilistic guarantees, but has computational challenges associated to its practical implementation. The alternative method only provides a local guarantee, but can more easily be applied in real time.

Let us denote $X_{\text{old}}$ and $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{old}}$ as the evidence used in computing the disturbance set $\hat{\mathcal{D}}(x)$, and $X_{\text{new}}$ and $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{new}}$ as the evidence acquired online after the disturbance set is computed. Conditioned on the old evidence, the function $d^j(x)$ is normally distributed with mean and variance given by (4.2) with $X = X_{\text{old}}$ and $\hat{\mathbf{d}}^{\mathbf{j}} = \hat{\mathbf{d}}^{\mathbf{j}}_{\text{old}}$, and the disturbance set is given by (4.4). If we also condition on the new evidence and keep the hyperparameters fixed, then the mean and variance are updated by modifying (4.2) with $X = [X_{\text{old}}, X_{\text{new}}]$ and $\hat{\mathbf{d}}^{\mathbf{j}} = [\hat{\mathbf{d}}^{\mathbf{j}}_{\text{old}}, \hat{\mathbf{d}}^{\mathbf{j}}_{\text{new}}]$.

**Remark 2** *Performing the update requires inverting $K^j([X_{old}, X_{new}], [X_{old}, X_{new}])$. This can be done efficiently employing standard techniques: since $K^j(X_{old}, X_{old})$ has already been inverted (in order to compute the disturbance bound $\hat{\mathcal{D}}$), all that is needed is inverting the Schur Complement of $K^j(X_{old}, X_{old})$ in $K^j([X_{old}, X_{new}], [X_{old}, X_{new}])$, which has the same size as $K^j(X_{new}, X_{new})$.*

Ideally we would incorporate $X_{\text{new}}$ and $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{new}}$ to relearn the GP hyperparameters as quickly as new measurements come in: otherwise new measured disturbance values $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{new}}$ will only affect the posterior mean, with the variance depending exclusively on where the measurements were made ($X_{\text{new}}$). However, performing this update online is computationally prohibitive. Instead, we update the hyperparameters every time a new estimated bound $\hat{\mathcal{D}}$ is produced for safety analysis, keeping them fixed in between. In practice the set $X_{\text{old}}$ will be much larger than $X_{\text{new}}$, so the estimated hyperparameters would not be expected to change significantly.

**Remark 3** *In settings where conditions are slowly time-varying, it may be desirable to give recently observed data more weight than older observations. This can naturally* *be encoded by the Gaussian process by appending time as an additional dimension in $X$: points that are distant in time would then be more weakly correlated, analogous to space.*

Based on the new Gaussian distribution, we can reason about the *posterior* confidence in the safety guarantees produced by our original safety analysis, which relied on the *prior* Gaussian distribution resulting from measurements $\hat{\mathbf{d}}^{\mathbf{j}}_{\text{old}}$ at states $X_{\text{old}}$.

#### 4.1.2.1 Global Bayesian safety analysis

The strongest result available for guaranteeing safety under the present framework is Corollary 1, which allows the system to exploit any superzero level set $\mathcal{Q}_\alpha$ ($\alpha \geq 0$) of the safety function $V$ throughout which the model is locally correct; all that is needed is for such a $\mathcal{Q}_\alpha$ to exist for $\alpha \in [0, V(x)]$ given the current state $x$.

It is possible to devise a safety policy to fully exploit the sufficient condition in Corollary 1 in a Bayesian setting: if the posterior probability that the corollary's hypotheses will hold drops to some arbitrary *global confidence threshold* $\gamma_0$, the safe controller can override the learning agent. With probability $\gamma_0$, the corollary will still apply, in which case the system is guaranteed to remain safe for all time; even if Corollary 1 does not apply at this time (which could happen with probability $1-\gamma_0$), it is still possible that the disturbance $d(x)$ will not consistently take adversarial values that force the computed safety function $V(x)$ to decrease, in which case the system may still evolve safely. Therefore, this policy guarantees a lower bound on the probability of maintaining safety for all time.

In order to apply this safety criterion, the system needs to maintain a Bayesian posterior of the sufficient condition in Corollary 1. We refer to this posterior probability as the *global safety confidence* $\gamma(x; X, \hat{\mathbf{d}}^{\mathbf{j}})$, or $\gamma(x)$ for conciseness:

$$\gamma(x; X, \hat{\mathbf{d}}^{\mathbf{j}}) := P\big(\exists \alpha \in [0, V(x)], \forall x \in \mathcal{Q}_\alpha : d(x) \in \hat{\mathcal{D}}(x) | X, \hat{\mathbf{d}}^{\mathbf{j}}\big). \tag{4.5}$$

Based on this, we propose the following least-restrictive control law:

$$\pi(x) = \begin{cases} \pi_l(x), & \text{if } \big(\gamma(x) > \gamma_0\big) \wedge \big(V(x) > 0\big), \\ \pi^*(x), & \text{otherwise,} \end{cases} \tag{4.6}$$

so the system applies any action it desires if the global safety confidence is above the threshold, but applies the safe controller once this is no longer the case.

Note that if confidence in the safety guarantees is restored after applying the safety action the learning algorithm will be allowed to resume control of the system. This can happen by multiple mechanisms: moving to a region with higher $V(x)$ will tend to increase the probability that *some* lower level set may satisfy the hypotheses of Corollary 1; moving to a region with less inconsistency between expected and observed dynamics will typically lead to higher posterior belief that *nearby* level sets will satisfy the hypotheses of Corollary 1; and generally acquiring new data may, in some cases, increase the posterior confidence that Corollary 1 may apply.

Computing the joint probability that the bound $\hat{\mathcal{D}}(x)$ captures the Gaussian process $d(x)$ *everywhere* on a level set $\mathcal{Q}_\alpha$ is not possible, since the set of functions $d(x)$ satisfying this condition is bounded on uncountably many dimensions, and thus not measurable in function space. Similarly, evaluating the joint probability for a continuum of level sets $\mathcal{Q}_\alpha$ for $\alpha \in [0, V(x)]$ is not feasible. Instead, exploiting the Lipschitz assumption on $d(x)$, we can obtain the sought probability $\gamma(x)$ from a marginal distribution over a sufficiently dense set of sample points on each $\mathcal{Q}_\alpha$ and a sufficiently dense collection of level sets between 0 and $V(x)$.

We can then use numerical methods [41] to compute the multivariate normal cumulative distribution function and estimate the marginal probability (using compact logic notation):

$$\gamma(x) \approx P\Bigg(\bigvee_{s=1}^{S} \bigwedge_{i=1}^{I} d(x_{s,i}) \in \hat{\mathcal{D}}(x_{s,i})\Bigg), \tag{4.7}$$

over $S$ level sets $0 = \alpha_0 < ... < \alpha_S = V(x)$ and $I$ sample points from each level set $Q_{\alpha_s}$. As the density of samples increases with larger $S$ and $I$, the marginal probability (4.7) asymptotically approaches the Gaussian process probability (4.5). Unfortunately, however, current numerical methods can only efficiently approximate these probabilities for multivariate Gaussians of about 25 dimensions [41], which drastically limits the number of sample points ($S \times I \approx 25$) that the marginal probability can be evaluated over, making it difficult to obtain a useful estimate. In view of this, a viable approach may be to bound (4.5) below as follows:

$$\gamma(x) \geq \underline{\gamma}(x) := \max_{\alpha \in [0, V(x)]} P\big(\forall x \in \mathcal{Q}_\alpha : d(x) \in \hat{\mathcal{D}}(x)\big), \tag{4.8}$$

and approximately compute this as

$$\underline{\gamma}(x) \approx \max_{s \in \{1,\ldots,S\}} P\Bigg(\bigwedge_{i=1}^{I} d(x_{s,i}) \in \hat{\mathcal{D}}(x_{s,i})\Bigg), \tag{4.9}$$

with the advantage that a separate multivariate Gaussian evaluation can be done now for each level set ($I \approx 25$). Computing this approximate probability as the system explores its state space provides a decision mechanism to guarantee safe operation of the system with a desired degree of confidence, which the system designer or operator can adjust through the $\gamma_0$ parameter.

#### 4.1.2.2 Local Bayesian safety analysis

Evaluating the expression in (4.9) is still computationally intensive, which can limit the practicality of this method for real-time validation of safety guarantees in some applications, such as mobile robots relying on on-board processing. An alternative is to replace the global safety analysis with a local criterion that offers much faster computation traded off with a weaker safety guarantee.

Instead of relying on Corollary 1, this lighter method exploits Proposition 6 and the fact that the disturbance model is locally correct, which we formalize now.

**Proposition 7** *If $d(x) \in \operatorname{int}\hat{\mathcal{D}}(x)$, then there exists $\Delta t > 0$ such that all possible trajectories followed by the system starting at $x$ will satisfy $d\big(\xi(\tau)\big) \in \hat{\mathcal{D}}\big(\xi(\tau)\big)$ for all $\tau \in [t, t + \Delta t]$.*

***Proof 5*** *Let $L_{\hat{\mathcal{D}}}$ be the Lipschitz (Hausdorff) constant of $\hat{\mathcal{D}}$, $L_d$ the Lipschitz constant of $d$, and $C_f$ a norm bound on the dynamics $f$. We then have that over an arbitrary time interval $[t, t + \Delta t]$, regardless of the control and disturbance signals $\boldsymbol{u}(\cdot)$, $\boldsymbol{d}(\cdot)$, any system trajectory starting at $\xi(t) = x$ satisfies $|\xi(\tau) - x| \leq C_f \Delta t, \forall \tau \in [t, t + \Delta t]$. This implies both $|d(\xi(\tau)) - d(x)| \leq L_d C_f \Delta t$ and $d_H\big(\hat{\mathcal{D}}(\xi(\tau)), \hat{\mathcal{D}}(x)\big) \leq L_{\hat{\mathcal{D}}}\mathcal{C}_f \Delta t$. Requiring that the open ball $B\big(d(x), (L_d + L_{\hat{\mathcal{D}}})C_f \Delta t\big)$ be contained in $\hat{\mathcal{D}}(x)$ ensures $d(\xi(\tau)) \in \hat{\mathcal{D}}(\xi(\tau))$. Since $d(x) \in \operatorname{int}\hat{\mathcal{D}}(x)$, there must exist a small enough $\Delta t > 0$ for which this condition is met.*

The system is allowed to explore the computed safe set freely as long as the probability of the estimated model $\hat{\mathcal{D}}$ being *locally reliable* remains above a certain threshold $\lambda_0$; if this threshold is reached, the safe controller intervenes, and the system is guaranteed to locally maintain or increase the computed safety value $V(x)$ with probability no less than $\lambda_0$. While this local guarantee does not ensure safety globally, it does constitute a useful heuristic effort to prevent the system from entering unexplored and potentially unsafe regions of the state space. Further, although the method is not explicitly tracking the hypotheses of Corollary 1, the local result becomes a global guarantee if these hypotheses do indeed hold.

We define the *local safety confidence* $\lambda(x; X, \hat{\mathbf{d}}^{\mathbf{j}})$, more concisely $\lambda(x)$[3] , as the posterior probability that $d(x)$ will be contained in $\hat{\mathcal{D}}(x)$ at the current state $x$, given all observations made until now:

$$\lambda(x; X, \hat{\mathbf{d}}^{\mathbf{j}}) := P\big(d(x) \in \hat{\mathcal{D}}(x) \mid X, \hat{\mathbf{d}}^{j}\big). \tag{4.10}$$

We then have the following local safety certificate.

**Proposition 8** *Let the disturbance $d(\cdot)$ be distributed component-wise as $n_d$ independent Gaussian processes* (4.1). *The safety policy $\pi^*(\cdot)$ is guaranteed to locally maintain or increase the system's computed safety $V(\cdot)$ with probability greater than or equal to the local safety confidence $\lambda(x)$.*

[3] Note that $\lambda$ was also used in Section 3.5.1 to represent the model reliability margin. We choose to reuse the variable here because it still serves as a measure of the local correctness of the model. Moving forward $\lambda$ will refer to the local safety confidence unless stated otherwise.

***Proof 6*** *The proof follows directly from Propositions 7 and 6, and the definition of $\lambda(x)$, noting that the boundary of $\hat{\mathcal{D}}(x)$ has zero Lebesgue measure and thus under any Gaussian distribution $P(d \in \operatorname{int} \hat{\mathcal{D}}(x) | d \in \hat{\mathcal{D}}(x)) = 1$.*

A *local confidence threshold* $\lambda_0 \in (0, p)$ can be established such that whenever $\lambda(x) < \lambda_0$ the model is considered insufficiently reliable (reachability guarantees may fail locally with probability greater than $1 - \lambda_0$), and the safety control is applied. The proposed safety control strategy is therefore as follows:

$$\pi(x) = \begin{cases} \pi_l(x), & \text{if } \big(V(x) > 0\big) \wedge \big(\lambda(x) > \lambda_0\big), \\ \pi^*(x), & \text{otherwise} \end{cases}. \tag{4.11}$$

Similarly to (4.6), under this control law, if confidence on the local reliability of the model is restored after applying the safe action and making new observations, the system will be allowed to resume its learning process, as long as it is in the interior of the computed safe set.

After generating a new GP model and defining $\hat{\mathcal{D}}(x)$ as described in Section 4.1.1, the prior probability with which the disturbance function $d(x)$ belongs to the set $\hat{\mathcal{D}}(x)$ is by design $p$ everywhere in the state space. As the system evolves, more evidence is gathered in the form of measurements of the disturbance along the system trajectory, so that the belief that $d(x) \in \hat{\mathcal{D}}(x)$ is updated for each $x$. In particular, in the Gaussian process model, this additional evidence amounts to augmenting the covariance matrix $K^j$ in (4.2) with additional data points and reevaluating the mean and variance of the posterior distribution of $d(x)$. Based on the new Gaussian distribution, $\lambda(x; X, \hat{\mathbf{d}}^{\mathbf{j}})$ can readily be evaluated for each $x$ as

$$\lambda(x) = \prod_{j=1}^{n_d} \frac{1}{2} \left[ \operatorname{erf}\left( \frac{d_+^j(x) - m^j(x)}{s^j(x)\sqrt{2}} \right) - \operatorname{erf}\left( \frac{d_-^j(x) - m^j(x)}{s^j(x)\sqrt{2}} \right) \right], \tag{4.12}$$

with $d_+^j(x) = \bar{d}^j(x) + z\sigma^j(x)$, $d_-^j(x) = \bar{d}^j(x) - z\sigma^j(x)$, $m^j(x) = \mathbb{E}[d^j(x)|X, \hat{\mathbf{d}}^j]$, $s^j(x) = \sqrt{\operatorname{var}(d^j(x)|X)}$; recall that $z$ was defined to yield the desired probability mass $p$ in $\hat{\mathcal{D}}(x)$ at the time of safety computation, as per (4.4). Also recall that $\bar{d}^j(x)$ and $\sigma^j(x)$ are determined from the previous batch of data and used to define the boundaries of $\hat{\mathcal{D}}(x)$. Thus (4.12) is integrating the new Gaussian distribution (which incorporates the new data points) over the set $\hat{\mathcal{D}}(x)$.

Parameters $p$ and $\lambda_0$ (or, in its case, $\gamma_0$) allow the system designer to choose the degree of conservativeness in the system: while $p$ regulates the amount of uncertainty accounted for by the robust model-based safety computation, $\lambda_0$ ($\gamma_0$) determines the acceptable degradation in the resulting certificate's posterior confidence before a safety intervention is initiated. A value of $p$ close to 1 will lead to a large, high-confidence $\hat{\mathcal{D}}(x)$ throughout the state space, but this analysis may result in a small or even empty safe set; on the other hand, if $p$ is low, $\hat{\mathcal{D}}(x)$ will be smaller and

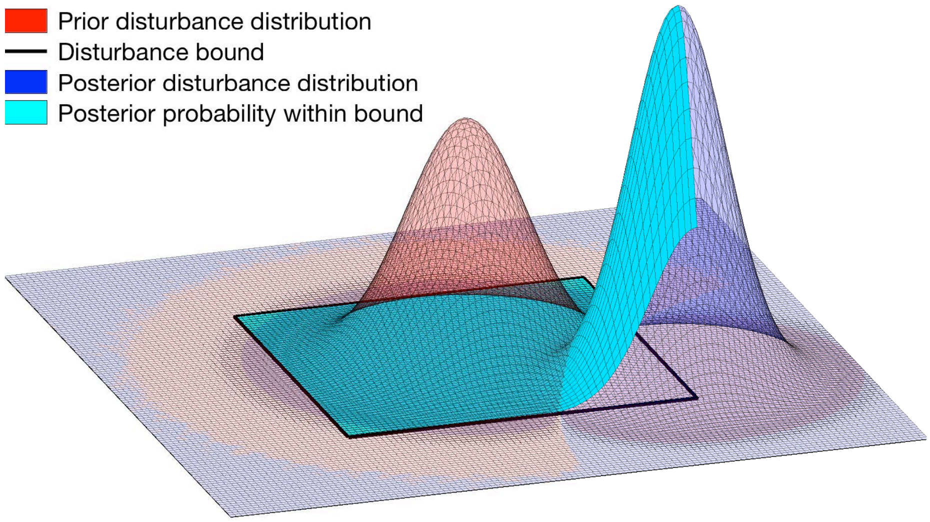


Figure 4.1: Evolution of the probability distribution of the disturbance $d(x)$ at a particular state $x$. The prior distribution is used to compute the bound $\hat{\mathcal{D}}(x)$ using confidence intervals, such that it contains a specified probability mass $p$. As more data are obtained, the distribution may shift, leading to a different posterior probability mass contained within $\hat{\mathcal{D}}(x)$.

the computed safe set will be larger, but guarantees are more likely to be deemed unreliable (as per $\lambda_0$ or $\gamma_0$) in light of later observations.

In the case of local safety analysis, immediately after computing a new model $\hat{\mathcal{D}}$, $\lambda(x)$ is by construction equal to $p$ everywhere in the state space. As more measurements are obtained, the posterior distribution over the disturbance changes, as illustrated in Fig. 4.1, which can result in $\lambda(x)$ locally increasing or decreasing. If $\lambda_0$ is chosen to be close to $p$, it is likely that the safety override will take place under minor deviations with respect to the model's prediction; as $\lambda_0$ becomes lower, however, the probability that the disturbance will violate the modeling assumptions before the safety controller intervenes increases. This reflects the fundamental tradeoff between risk and conservativeness in safety-critical decision making under uncertainty. The proposed framework therefore allows the system designer to adjust the degree of conservativeness according to the needs and characteristics of the system at hand.

### 4.1.3 Experiment: Safe Learning with GP

*This experiment uses the quadrotor testbed described in Section 3.3.1.*

In this experiment, we demonstrate the iterative updating of the safe set and safety policy using observations of the system dynamics gathered over time, as well as the online validation of the resulting guarantees. All components of the framework are active during the test, namely *learning controller*, *safety policy*, *iterative safety re-computation*, and *Bayesian guarantee validation*, with the main focus being on the latter two.

Here, the Pelican quadrotor attempts to safely track the same reference trajectory, while using the gathered information about the system's evolution to refine its notion of safety. In this case, the policy gradient learning algorithm is initialized to a hand-tuned set of parameter values. The initial dynamic model available to the safety algorithm is identical to the one used in the previous experiment, with a uniform uncertainty bound of $\pm 1.5\text{m/s}^2$. However, the system is now allowed to update this bound, throughout the state space, based on the disturbance posterior computed by a Gaussian process model.

To learn the disturbance function, the system starts with a Gaussian process prior over $d(\cdot)$ defined by a zero mean function and a squared exponential covariance function:

$$k(x, x') = \sigma_f^2 \exp\left(\frac{(x - x')^T \mathcal{L}^{-1}(x - x')}{2}\right), \tag{4.13}$$

where $\mathcal{L}$ is a diagonal matrix, with $\mathcal{L}_i$ as the $i$th diagonal element, and $\theta_p = \left[\sigma_f^2, \sigma_n^2, \mathcal{L}_1, \mathcal{L}_2\right]$ are the hyperparameters, $\sigma_f^2$ being the signal variance, $\sigma_n^2$ the measurement noise variance, and the $\mathcal{L}_i$ the squared exponential's characteristic length for position and velocity respectively. The hyperparameters are chosen to maximize the marginal likelihood of the training data set, and are recomputed for each new batch of data when a new disturbance model $\hat{\mathcal{D}}(x)$ is generated for safety analysis. Finally, the chosen prior mean and covariance kernel classes are both Lipschitz continuous, ensuring that all required technical conditions for the theoretical results hold.

The expressions (4.2), (4.3) give the marginal Gaussian process posterior on $d(x^*)$ for a query point $x^*$. To numerically compute the safe set, the system first evaluates (4.4) to obtain the disturbance bound $\hat{\mathcal{D}}(\mathbf{x_i})$ at every point $\mathbf{x_i}$ on a state-space grid, as the 95% confidence interval ($p = 0.95$) of the Gaussian process posterior over $d(x)$; next, it performs the robust safety analysis by numerically solving the HJI equation (2.6) on this grid (using [63]) and obtaining the safety function $V(x)$.

The trajectory followed by the quadrotor in this experiment is shown in Fig. 4.2. The vehicle starts off with an *a priori* conservative global bound on $d(x)$ and computes an initial conservative safe set $\Omega_1$ (Fig. 4.3). It then attempts to track the reference trajectory avoiding the unsafe regions by transitioning to the safe control $u^*(x)$ on $\partial\Omega_1$. The disturbance is measured and monitored online during this test, under the local safety confidence criterion, and found to be locally consistent with the initial conservative bound. After collecting 10 s of data, a new disturbance bound $\hat{\mathcal{D}}(x)$ is constructed using the corresponding Gaussian process posterior, from which a second safety function $V_2(x)$ and safe set $\Omega_2$ are computed via Hamilton-Jacobi reachability analysis. This process takes roughly 2 seconds, and at approximately

$t = 12$ s the new safety guarantees and policy are substituted in.

The Pelican continues its flight under the results of this new safety analysis: however, shortly after, the vehicle measures values of $d$ that consistently approach the boundary of $\hat{\mathcal{D}}(x)$, and reacts by applying the safe control policy and locally climbing the computed safety function. This confidence-based intervention takes place several times during the test run, as the vehicle measures disturbances that lower its confidence in the local model bounds, effectively preventing the vehicle from approaching the ground.

After a few seconds, a new Gaussian process posterior is computed based on the first 20 s of flight data, resulting in an estimated safe set $\Omega_3$, an intermediate result between the initial conservative $\Omega_1$ and the overly permissive $\Omega_2$ (Fig. 4.3). The learning algorithm is then allowed to resume tracking under this new safety analysis, and no further safety overrides take place due to loss of safety confidence.

This experiment demonstrates the algorithm's ability to safely refine its notion of safety as more data become available, without requiring the process to consist in a series of strictly conservative under-approximations.

### 4.1.4 Experiment: Gone with the Wind

*This experiment uses the quadrotor testbed described in Section 3.3.1.*

In this experiment, we display the efficacy of online safety guarantee validation in handling alterations in operating conditions unforeseen by the system designer. All components of the framework are active, except for the *iterative safety re-computation*, which is not used in this case.

This experiment is performed using the lighter Hummingbird quadrotor, which is more agile than the Pelican but also more susceptible to wind. We initialize the disturbance set to a conservative range of ±2 $\text{m/s}^2$, which amply captures the error in the double-integrator model for vertical flight. The vehicle tracks a slow sinusoidal trajectory using policy gradient [56] to improve upon the manually initialized controller parameters. At approximately $t = 45$ s an unmodeled disturbance is introduced by activating a fan aimed laterally at the quadrotor. The fan is positioned on the ground and angled slightly upward, so that its effect increases as the quadrotor flies closer to the ground. The presence of the airflow causes the attitude and lateral position controllers to use additional control authority to stabilize the quadrotor, which couples into the vertical dynamics as an unmodeled force.

The experiment is performed with and without the *Bayesian guarantee validation* component, with resulting trajectories shown in Fig. 4.5. Without validation, the quadrotor violates the constraints, repeatedly striking the ground. With validation, the fan's airflow is quickly detected as a discrepancy with the model near the floor, and the safety controller override is triggered. The vehicle avoids entering the affected region for the remainder of the flight. Although only the local confi-

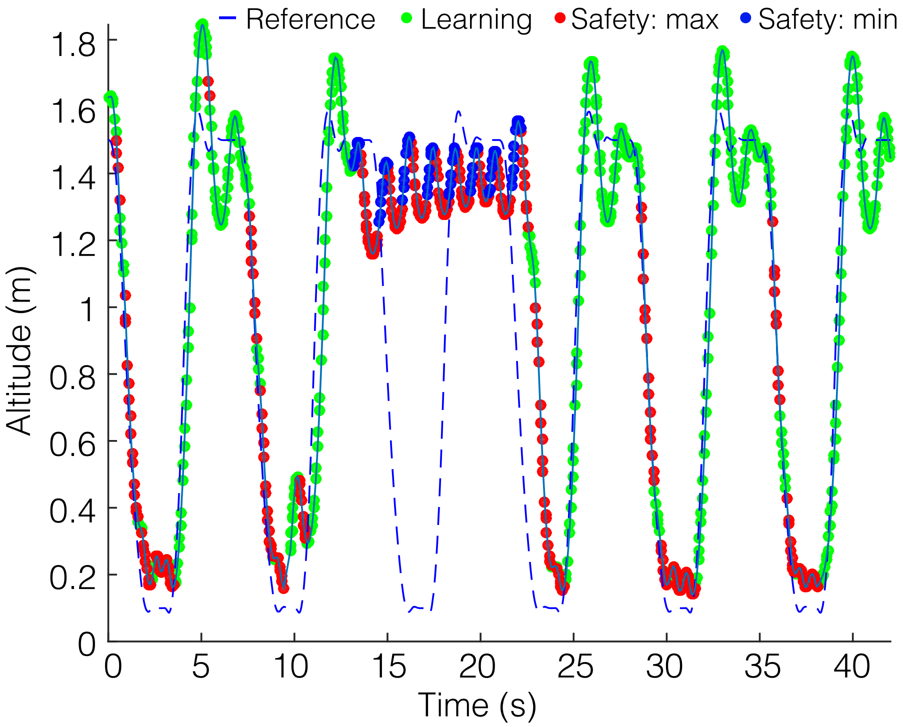


Figure 4.2: Vehicle altitude and reference trajectory over time. After flying with an initial conservative model, the vehicle computes a first Gaussian process model of the disturbance with only a few data points, resulting in an insufficiently accurate bound. The safety policy detects the low confidence and refuses to follow the reference to low altitudes. Once a more accurate disturbance bound is computed, tracking is resumed, with a less restrictive safe set than the original one.

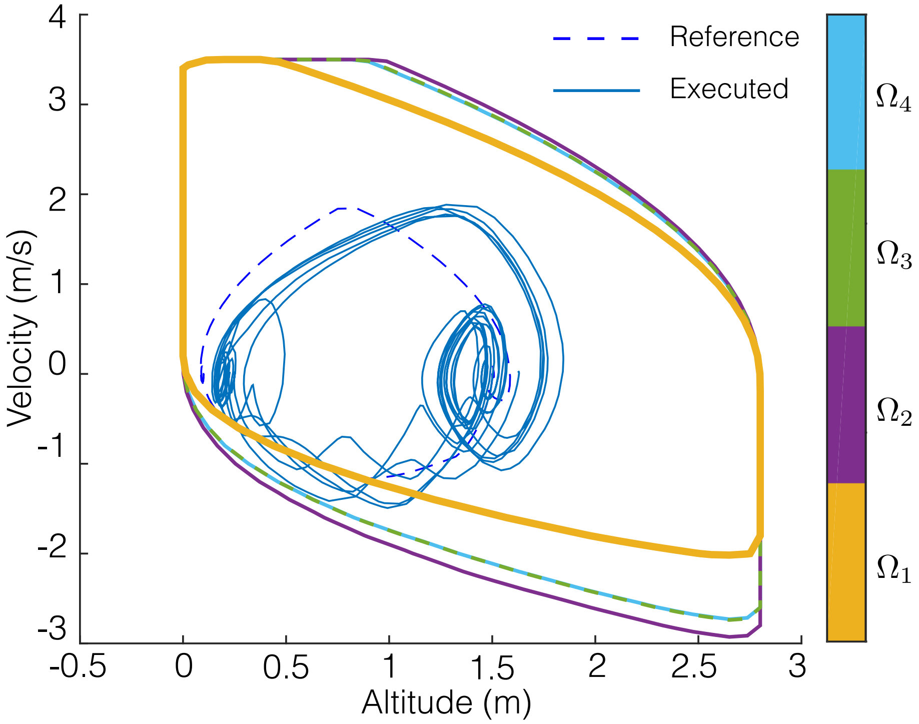
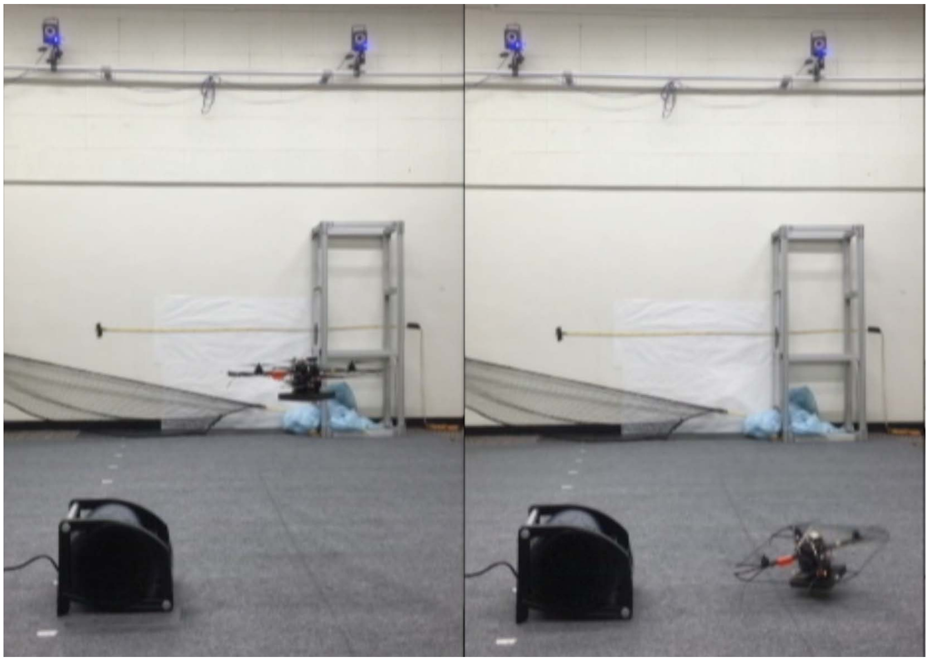


Figure 4.3: Safe sets computed online by the safety algorithm as it gathers data and successively updates its Gaussian process disturbance model. The vehicle's trajectory eventually leaves the initial, conservative $\Omega_1$, but remains in the converged safe set ($\Omega_4$) at all times, even *before* this set is computed. While the intermediate set $\Omega_2$ would have been overly permissive, this is remedied by the intervention of the safety controller as soon as the model is observed to behave poorly.



(a) With online guarantee validation (b) Without online guarantee validation

Figure 4.4: Hummingbird quadrotor learning a vertical flight policy under the requirement of not colliding. When the fan is turned on, the system experiences an unmodeled disturbance that it has not previously encountered. This can lead to a ground collision even under robust safety policies (right). The proposed Bayesian validation method detects the inconsistency and prevents the vehicle from entering the uncertain region (left).

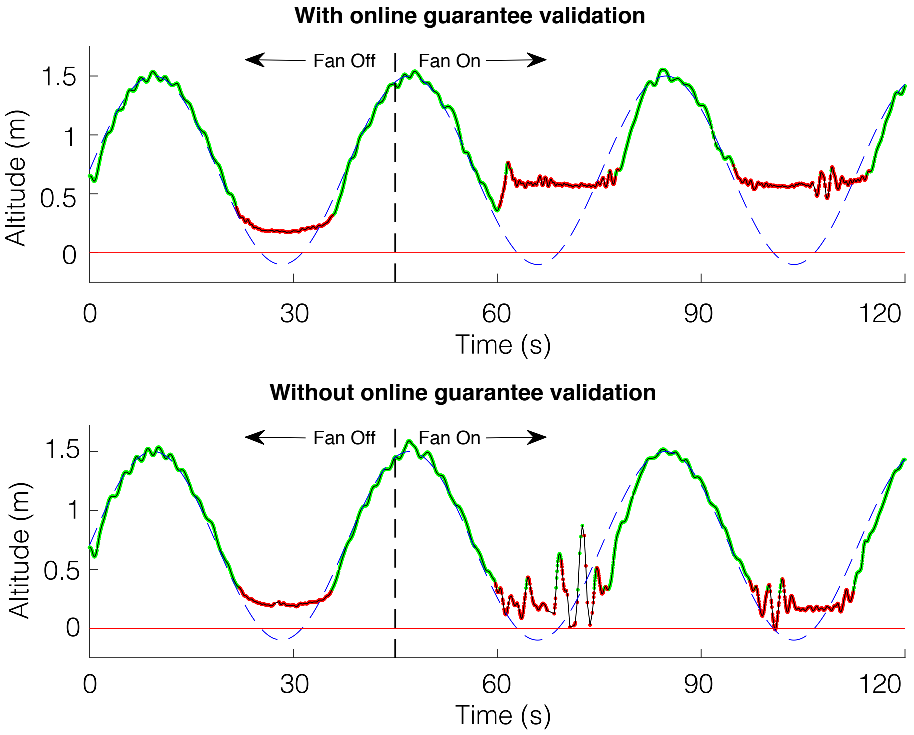


Figure 4.5: Vehicle altitude and reference trajectory over time, shown with and without online model validation. After the fan is turned on, the vehicle checking local model reliability detects the inconsistency and overrides the learning controller, avoiding the region with unmodeled dynamics; the vehicle without model validation enters this region and collides with the ground multiple times. The behavior is repeated when the reference trajectory enters the perturbed region a second time.

dence method is used, providing a strictly local safety guarantee, the safe controller succeeds in maintaining safety throughout the experiment. This provides strong evidence suggesting that, beyond its theoretical guarantees, the local Bayesian analysis also constitutes an effective best-effort approach to safety in more general conditions, given limited computational resources and available knowledge about the system.

## 4.2 Local-updates via Temporal Differencing

We now present a model-free technique for learning the safe set, which as the name implies, modifies the value function directly without constructing a model. In particular, we will focus on *temporal differencing* (TD), which comes from the field of Reinforcement Learning (RL), where goal is to solve a discrete time sum of rewards optimal control problem for systems with unknown dynamics.

With TD the value function is represented with a function approximator[4], and experiences from the real system are used to directly update the parameters of the function approximator. A key benefit of model-free techniques is that the experiences (which amount to individual data points) can be quickly incorporated into the value function, which allows these methods to be run online. For model-based approaches one needs to run reachability analysis to update the value function, and this computation can be quite expensive. In the model-free setting, we give up global accuracy, but the value function can be updated quickly in the regions of the state space that the system actually visits.

We will briefly introduce the RL problem and the TD method for solving this problem. We then show how TD can be used within the safe learning context to more accurately approximate the value function (safe set) locally.

### 4.2.1 Reinforcement Learning and Temporal Differencing

*Note that in this section we will solve a different type of optimal control problem, so we briefly redefine the terms value function and optimal policy for this section.*

RL seeks to generate optimal policies for solving sequential decision-making problems through observations of interactions between an agent and its environment. These problems consider stochastic systems with discrete dynamics. In order to keep the notation consistent we will consider a discrete time approximation of our dynamical system $x_{k+1}{=}f_{\Delta t}(x_k, u_k, d(x_k))$, where $\Delta t$ is the sample time, $k$ is the time step, and

$$f_{\Delta t}(x, u, d(x)) = x + f(x, u, d(x)) \cdot \Delta t. \tag{4.14}$$

The RL problem is modeled as a Markov Decision Process (MDP): given an initial state $x_0$, reward $R(x, u)$, a discount factor $\gamma \in [0, 1]$, the system dynamics $f_{\Delta t}(x, u, d(x))$, the objective is to find a policy $\pi : \mathbb{R}^n \to \mathcal{U}$ that maximizes the expected sum of future discounted rewards [5]

$$\max_{\pi} E_{\pi, x_0}\left[\sum_{i=0}^{\infty} \gamma^i R(x_i, \pi(x_i))\right]. \tag{4.15}$$

[4] A simple function approximator can be linear interpolation on a grid. In that sense the value function is always represented through a function approximator since, in general, we can only approximate the solution to the HJ equation on a grid.

[5] An expectation is included because in general the system dynamics can be stochastic. In particular, this stochasticity can be captured in our model by considering $d(x)$ as a state dependent random variable with support $\mathcal{D}$, which in turn makes $f$ a random variable. For now we still maintain that our system is deterministic.

For a policy $\pi$ we can define the *evaluation function* $V^{\pi}(x) = E_{\pi,x}\left[\sum_{i=0}^{\infty} \gamma^i R(x_i, \pi(x_i))\right]$. The *optimal policy* $\pi^*$, which maximizes (4.15), will yield the value function $V$, which in turn satisfies *Bellman's Optimality Condition*, namely:

$$V(x) = \max_{u \in \mathcal{U}} R(x,u) + \gamma E_x[V\left(f_{\Delta t}(x,u,d(x))\right)]. \tag{4.16}$$

The evaluation function for a given policy must satisfy the following condition

$$V^{\pi}(x) = R(x,\pi(x)) + \gamma E_x[V^{\pi}\left(f_{\Delta t}(x,\pi(x),d(x))\right)], \tag{4.17}$$

which for the optimal policy is equal to the value function, $V^{\pi^*} = V$.

Once $V$ is known, determining the optimal policy is straightforward,

$$\pi^*(x) = \arg\max_{u \in \mathcal{U}} R(x,u) + \gamma E_x[V\left(f_{\Delta t}(x,u,d(x))\right)]. \tag{4.18}$$

Most RL methods iterate back and forth between approximating $V$ and determining $\pi^*$ [16], [90]. As stated earlier one of the methods to solve this problem is TD, which was famously applied to create TD Gammon, a program that learned to play backgammon at an expert level [85]. TD is an online algorithm that obtains new estimates of the value function at every time step by taking differences of current estimates of the value function at successive states.

When the state space is discrete and finite the value function is given as a table, and TD simply updates the values of the table. For a continuous state space, we first approximate the value function with a parameterized function $V(x) \approx V_\theta(x)$ with parameter vector $\theta \in \mathbb{R}^{n_\theta}$, and then update the parameter vector. Similarly the evaluation function can be parameterized $V^{\pi}(x) \approx V^{\pi}_{\theta}(x)$. A common class of approximators are linear parameterizations, $V_\theta(x) = v(x) \cdot \theta$. This class also includes interpolation over a grid, $v(x) \cdot \theta = \phi(x) \cdot \vec{V}$, where $\phi$ is the interpolation function and the parameter vector $\vec{V}$ contains the values at the grid nodes.

Given a policy, TD seeks to find the evaluation function by performing stochastic gradient descent on the following loss function

$$\mathcal{L}(\theta) = \frac{1}{2}\left(R(x,\pi(x)) + \gamma V^{\pi}_{\theta_k}(x^+) - V^{\pi}_{\theta_k}(x)\right)^2, \tag{4.19}$$

where $x^+ = f(x,\pi(x),d(x))$. Note $x^+$ can be obtained from the environment by applying $\pi(x)$ at state $x$, so $f$ does not need to be known explicitly. Also note that the loss function is defined such that it is minimized in expectation when the evaluation function equation (4.17) is satisfied.

Stochastic gradient descent is run using the samples $\{(x_k, \pi(x_k), x_k^+)\}_k$ obtained from the environment (physical system or simulation), and at every time step the parameter vector is updated

$$\theta^{k+1} \leftarrow \theta^k + \alpha \nabla_\theta V^{\pi}_{\theta_k}(x_k)\left(R(x_k,\pi(x_k)) + \gamma V^{\pi}_{\theta_k}(x_k^+) - V^{\pi}_{\theta_k}(x_k)\right), \tag{4.20}$$

where $\alpha \in [0,1]$ is the learning rate. This is the simplest TD update rule, and we refer the interested reader to [82] and [86] for more sophisticated variants of TD learning.

To get the value function, TD must be incorporated into a policy iteration algorithm that iteratively improves the policy [5], [72]. If the policy converges to the optimal policy then the TD algorithm (if successful) yields the value function, since the evaluation function of $\pi^*$is precisely $V$.

### 4.2.2 Temporal Differencing for Reachability Analysis

We now show how temporal differencing can be used to learn safe sets. Under a worst-case treatment of the system uncertainty the safe set is obtained as the solution to (2.6). If the model were known perfectly equation (2.6) becomes

$$0 = \min\left\{ l(x) - V(x,t), \frac{\partial V}{\partial t}(x,t) + \max_{u \in \mathcal{U}} \frac{\partial V}{\partial x}(x,t) f(x,u,d(x)) \right\} \tag{4.21a}$$

$$V(x,T) = l(x)., \tag{4.21b}$$

and, again, the value function $V(x)$ could be obtained as $T \to \infty$.

The discrete time approximation of the value function satisfies

$$V(x) = \min\{l(x), \max_{u \in \mathcal{U}} V(x + \Delta t f(x,u,d(x)))\}, \tag{4.22}$$

with the optimal policy given by

$$\pi^*(x) = \arg\max_{u \in \mathcal{U}} \min\{l(x), V(x + \Delta t f(x,u,d(x)))\}, \tag{4.23}$$

Given the optimal policy $\pi^*$, (4.22) can be written as

$$V(x) = \min\{l(x), V(x + \Delta t f(x, \pi^*(x), d(x)))\}.^6 \tag{4.24}$$

In general (4.21) is solved on a grid, so the value function is represented by a linear function approximator $V(x) = V_\theta(x) = \phi(x) \cdot \vec{V}$, where $\phi(x)$ is the interpolation function over the grid and $\theta = \vec{V}$ contains the grid node values.

Equation (4.25) allows us to characterize the value function, and it can be used to define a loss function that is nonnegative, and only zero when (4.25) is satisfied

$$\mathcal{L}(\theta) = \left(\min\{l(x), V_\theta(x^+)\} - V_\theta(x)\right), \tag{4.25}$$

[6] Note here a slight abuse of notation: following the convention of Section 2.2.3 the discrete time approximation should be represented by $V_{\Delta t}$, but to maintain consistency with Section 4.2.1 and to minimize notation we use $V$ for both the continuous and discrete time value function in this section.

where $x^+ = f(x, \pi^*(x), d(x))$. Once again, the dynamics $f$ need not be known explicitly. Using the samples $\{(x_k, \pi(x_k), x_k^+)\}_k$, we perform the following TD update at every time step:

$$\theta^{k+1} \leftarrow \theta^k + \alpha \nabla_\theta V_{\theta_k}(x_k) \left( \min\{l(x_k), V_{\theta_k}(x_k^+)\} - V_{\theta_k}(x_k) \right), \tag{4.26}$$

or in the specific case when the function approximation is a grid interpolation

$$\vec{V}^{k+1} \leftarrow \vec{V}^k + \alpha \phi(x_k) \left( \min\{l(x_k), \phi(x_k^+) \cdot \vec{V}^k\} - \phi(x_k) \cdot \vec{V}^k \right), \tag{4.27}$$

The presentation thus far assumes that the optimal policy is known. We only make this assumption for ease of presentation. In general we can define a policy evaluation function and policy-dependent loss function like we did in the previous section. The TD algorithm would not be any different, but it would need to be combined with a policy iteration algorithm to ultimately yield the value function.

### 4.2.3 Safe Learning with Temporal Differencing

Leveraging TD in the context of safe learning presents some unique challenges over the traditional RL setting. First, if we are dealing with a safety critical system, safety should not be compromised while learning the safe set. Second, in RL the priority is to learn the value function, so almost all the experiences that result from the actions can be used for learning. On the other hand, in safe learning, learning the safe set (value function) is an auxiliary task, so we need to determine which experiences are actually eligible for learning since many of the actions are not taken with this objective in mind.

To address the first challenge, we recall that the model is only partially unknown and is in fact bounded in its uncertainty. By performing reachability analysis with the dynamics $f$ and disturbance set $\mathcal{D}$ we have a value function and optimal safe policy (under the disturbance $\mathcal{D}$) that ensures safe operation of the system as long as $d(x) \in \mathcal{D}$. In learning the value function for the perfectly known system we can use the policy from the robust reachability analysis as the optimal policy. Even if the two policies are different, this choice will only lead to a conservative under-approximation of the value function and safe set. We can also use the value function from the robust analysis to initialize the TD algorithm.

The robust reachability analysis also leads to a solution for the second challenge. With the resulting value function and optimal policy a least-restrictive control law can be applied. Since the least restrictive control law only takes the optimal safe action on the boundary of the safe set, i.e. $V(x) <= 0$, the value function can only be updated from experiences near the boundary. At first this may seem like a shortcoming, but it has some practical benefits. First, it ensures that the safe set can only be modified gradually, which is important in the context of a safety critical system. Second, learning is concentrated in areas that the system is visiting due to its other objectives (e.g. tracking a reference), thus the safe set is only modified in areas that could potentially help the system have better overall performance.

The full TD reachability algorithm is given by Algorithm 1.

```
Algorithm 1: Online Reachability through TD
Input: $f$, $\mathcal{U}$, $\mathcal{D}$, $\mathcal{K}$, $l$, $\alpha$, $\Delta t$, $\mathcal{O}$, $j$
Initialization:
$\pi^*, \vec{V}^0 \leftarrow$ HJIReachability($f$, $\mathcal{U}$, $\mathcal{D}$, $\mathcal{K}$, $l$)
$k = 0$
Repeat:
  if $\phi(x_k) \cdot \vec{V}^k \leq 0$ then
    $u_k = \pi^*(x_k)$
    $\vec{V}^{k+1} \leftarrow \vec{V}^k + \alpha\phi(x_k)\Big(\min\{l(x_k), \phi(x_{k+1}) \cdot \vec{V}^k\} - \phi(x_k) \cdot \vec{V}^k\Big)$
  else
    $u_k \in \mathcal{U}$
    $\vec{V}^{k+1} = \vec{V}^k$
  end if
  $k = k + 1$
```

### 4.2.4 Model-based Temporal Differencing

A key benefit of TD with a grid interpolation function approximator is that updates to the value function can be made quickly and cheaply, since only a few parameters change at every iteration. However, model-free methods are not data-efficient. Model-free methods essentially use one sample to make one update. Model-based methods on the other hand aggregate all the data to make many updates since ultimately reachability analysis modifies all the grid node values.

To get the best of both worlds, TD can be performed at multiple grid nodes simultaneously at each iteration. Define the *neighborhood* of state $x$, $N_j(x)$, as the $j$ nearest grid nodes (typically in the 1 norm), if the state $x$ is visited we can perform TD at each state in $N_j(x)$. Since the points in the neighborhood are not actually visited, we need to infer the dynamics at these points. We assume access to a time-varying oracle $\mathcal{O}^k : \mathbb{R}^n \to \mathbb{R}^n$ that takes a state $x$, and predicts the subsequent state under the optimal policy. The oracle does not know the subsequent state but rather estimates it given data from the system. Effectively, the oracle is a model, but a key distinction is that it should be able to update quickly as new data is presented (e.g. iterative least squares) since TD is run online. Once we have an oracle we can update the value function over a neighborhood according to Algorithm 2.

**Algorithm 2:** Model-based Online Reachability through TD

```
Input: $f$, $\mathcal{U}$, $\mathcal{D}$, $\mathcal{K}$, $l$, $\alpha$, $\Delta t$, $\mathcal{O}$, $j$
Initialization:
$\pi^*(x), \vec{V}^0(x) \leftarrow$ HJIReachability($f$, $\mathcal{U}$, $\mathcal{D}$, $\mathcal{K}$, $l$)
$k = 0$
Repeat:
  if $\phi(x_k) \cdot \vec{V}^k \leq 0$ then
    $u_k = \pi^*(x_k)$
    for $y$ in $N_j(x_k)$ do
      $y^+ = \mathcal{O}^k(y)$
      $\vec{V}^{k+1} \leftarrow \vec{V}^k + \alpha\phi(y) \Big(\min\{l(y), \phi(y^+) \cdot \vec{V}^k\} - \phi(y) \cdot \vec{V}^k\Big)$
    end for
  else
    $u_k \in \mathcal{U}$
    $\vec{V}^{k+1} = \vec{V}^k$
  end if
  $k = k + 1$
```

Assuming Lipschitz continuous dynamics and a fine grid, the vector field over $N_j(x)$ is well approximated by the vector field at $x$, thus we choose the following oracle

$$\mathcal{O}^k(y) = y + x_k^+ - x_k, \quad y \in N_j(x_k), \tag{4.28}$$

where $(x_k, x_k^+)$ is the transition experienced at time step $k$ under policy $\pi^*$.

### 4.2.5 Experiment Setup

In the next two sections,4.2.6 and 4.2.7, we demonstrate this algorithm on some simulations. In both experiments the system has some performance objective in addition to a safety constraint, and TD is employed to help achieve the objective by modifying the safe set. We choose learning rate $\alpha = 1$ to make the safe set grow at the fastest rate. We select the number of nearest neighbors $j = 3^n$, where $n$ is the dimension of the state vector. We find the nearest grid point to the state where we want to update the safe set, and take the smallest $n-$dimensional grid centered around that grid point to be our neighborhood. The idea is to update the safe set uniformly along all dimensions and to keep the inference region small since our oracle will be more accurate locally.

### 4.2.6 Experiment: Learning to Track a Trajectory

Consider a simplified affine model for the vertical dynamics of a quadrotor with an unknown payload

$$\ddot{z} = g + du, \tag{4.29}$$

where the state $x = [\dot{z} \quad z]$ is the vertical position and velocity, $g = -9.8\text{m/s}^2$ is the gravitational constant, the control $u$ is the motor thrust, and the disturbance $d$ accounts for the scaling of the motor thrust, which depends on the mass of the quadrotor (and its payload). For this example $\mathcal{U} = [0, g]$ and $\mathcal{D} = [1.1, 1.5]$. These values are chosen heuristically to represent a range of systems that have enough thrust to overcome gravity.

The objective is to learn to track a trajectory, while staying in a bounded set $\mathcal{K} = \{x : 0\text{m} \ \leq z \leq 3\text{m} \ \}$. Our motivation comes from learning landing controllers for delivery drones, where the mass of the package may vary and we want to avoid collisions with the ground. For learning we use Policy Gradient via the Signed Derivative (PGSD) [55]. PGSD is an on-policy reinforcement learning technique that learns better controllers by observing the outcomes of the current controller. We chose this algorithm primarily for its ease of implementation. For brevity we omit the details of the algorithm, and refer the interested reader to [55].

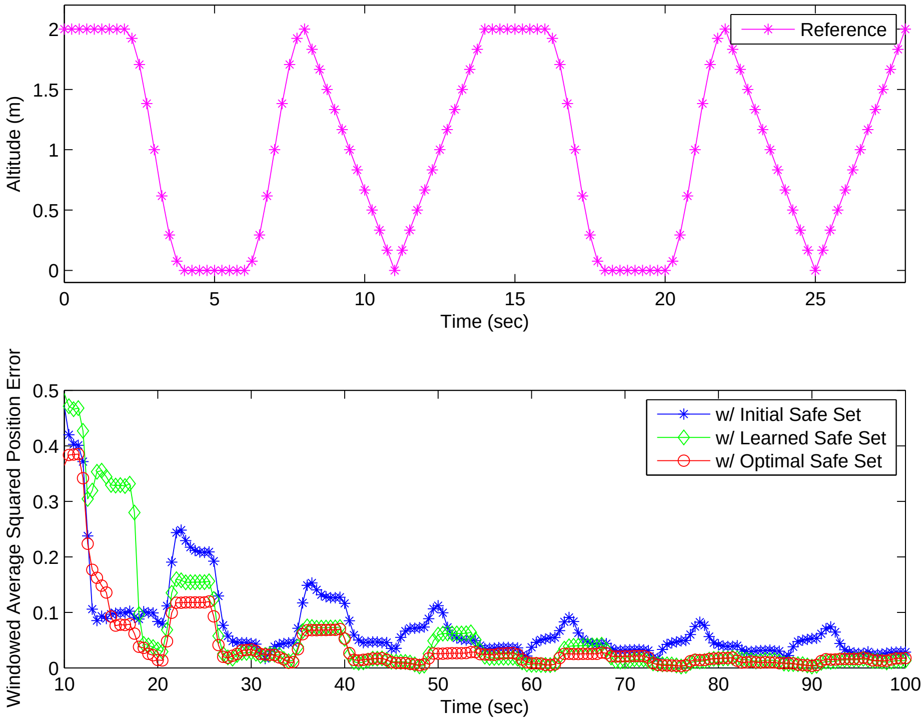


Figure 4.6: The reference trajectory being tracked (top). The squared position tracking error averaged over a five second moving window (bottom).

In the simulation we choose the disturbance function $d = 1.4$, while the initial safe set is computed with $\mathcal{D}$ (thus the initial control strategy will be conservative). In all the trials, PGSD is active when the system is in the safe set, otherwise the optimal safe action is applied. In the first simulation the safe set is not updated. In the second simulation the safe set is updated according to Algorithm 1. A third simulation is also done using the safe set for the perfectly known deterministic model. The comparison between the simulations can be seen in Fig. 4.6.

Though the reference trajectory remains inside $\mathcal{K}$, it exits the initial safe set as seen in Fig. 4.7. It is actually impossible to follow this trajectory using the least restrictive control strategy (3.1). However, the trajectory is completely contained in the final learned safe set. The performance of the learning algorithm is enhanced when the safe set is updated. With the updated safe set we are eventually able to converge to the same tracking performance obtained using the optimal safe set. It is also important to note that the safe set is mainly updated in regions that are relevant to achieving the task (i.e. where the reference trajectory visits).

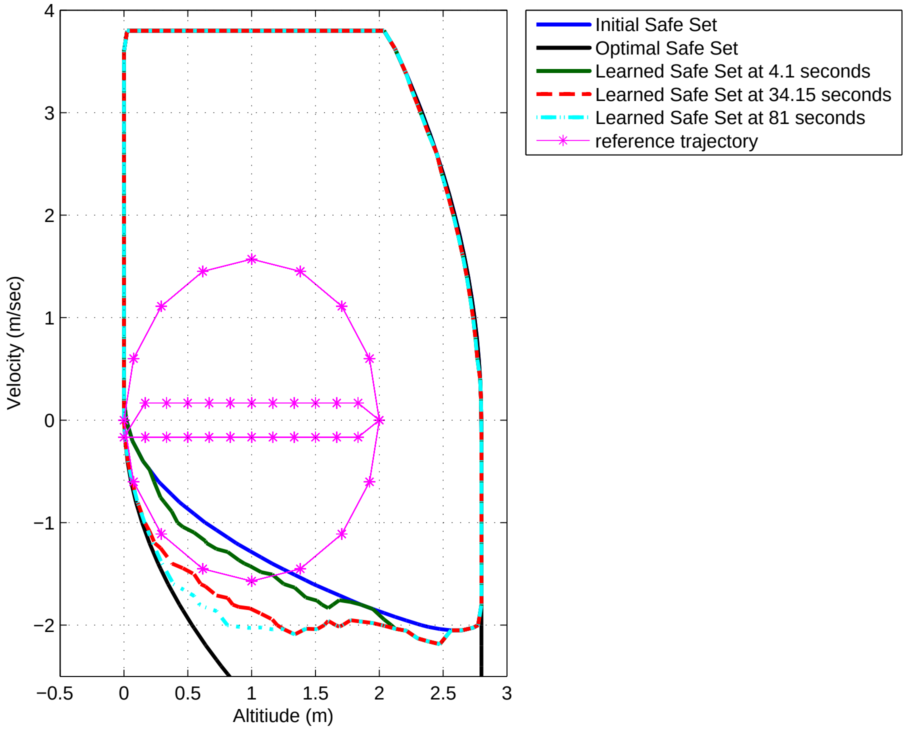


Figure 4.7: Snapshots of the safe set at different times during the simulation. The reference trajectory exits the initial safe set, but stays inside the learned safe set.

### 4.2.7 Experiment: Catch Me if You Can

HJI reachability has been extensively applied to zero-sum differential games, particularly for solving reach-avoid [60] and pursuit-evasion games [62]. A reach-avoid game involves two players. Player $a$ attempts to reach a goal $\mathcal{G}$, while player $b$ tries to prevent her from reaching $\mathcal{G}$. In the pursuit-evasion setting, player $b$ attempts to capture player $a$, while player $a$ tries to avoid capture. In both games, the two players must also avoid obstacles $\Psi$. Each game can be solved using a modification of (2.6) [36], [48].

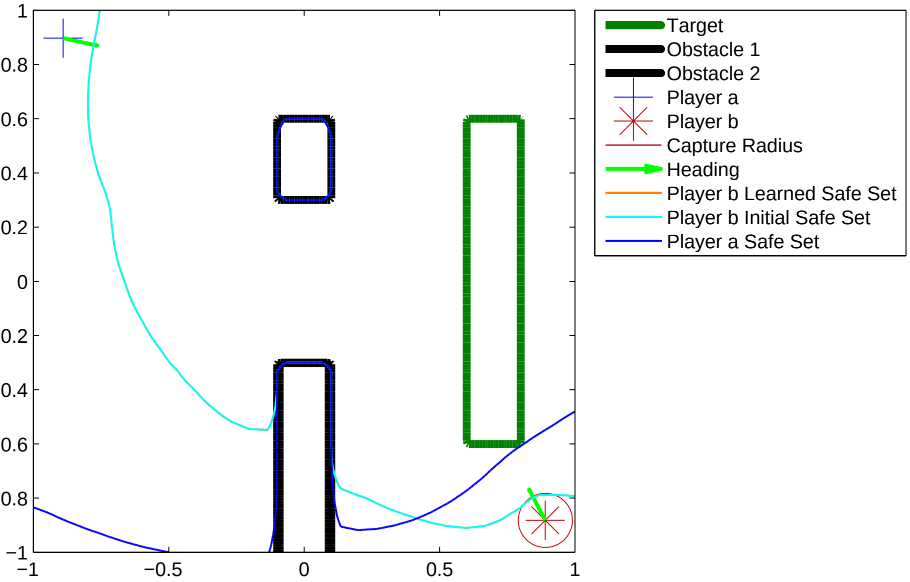


Figure 4.8: The game is initialized so that both players are safe assuming equal speeds. If player $a$ is inside the player $a$ safe set then she can always avoid capture. If player $a$ is inside the player $b$ safe set then player $b$ can defend the goal. Note that the goal is referred to here as the target.

Take the state $x = \begin{bmatrix} x_a & y_a & x_b & y_b \end{bmatrix}^T$ as the joint position of both players in $\mathbb{R}^2$. The dynamics are specified as

$$\dot{x} = \begin{bmatrix} d_1 \cos(d_2) \\ d_1 \sin(d_2) \\ u_1 \cos(u_2) \\ u_1 \sin(u_2) \end{bmatrix}, \tag{4.30}$$

where $d = \begin{bmatrix} d_1 & d_2 \end{bmatrix}$ is the speed and heading angle of player $a$, and $u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}$ is the speed and heading angle of player $b$.

Define the sets $\Psi_a$ and $\Psi_b$ as the set of states for which player $a$ is inside the obstacle, and the set of states for which player $b$ is inside the obstacle, respectively. Player $b$ captures player $a$ if $\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \leq R_C$, where $R_C$ is the capture radius. The capture set $\mathcal{C}$ is the set of states for which player $a$ has been captured. Lastly, define a bounded Lipschitz continuous function $\omega_a : \mathbb{R}^n \to \mathbb{R}$, such that $\mathcal{C} \cup \Psi_a = \{x \mid \omega_a(x) < 0\}$. A typical choice for $\omega_a$ is the signed distance function $s_{\mathcal{C} \cup \Psi_a}$.

For the reach-avoid game player $b$ has a safety specification, namely to defend the goal, so the target set (which is the complement to the constraint set $\mathcal{K}$) is given by $\mathcal{T} = \Psi_b \cup \mathcal{G}_a$, where $\mathcal{G}_a$ is the set of states corresponding to player $a$ reaching the goal. The solution for this game solves

$$\begin{aligned}\max\Big\{ &\min\{l(x) - V_{RA}(x,t), \frac{\partial V_{RA}(x,t)}{\partial t} + \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} \frac{\partial V_{RA}(x,t)}{\partial x} \cdot f(x,u,d)\},\\ &- \omega_a(x) - V_{RA}(x,t)\Big\} = 0, \quad \forall t \in [0,T]\end{aligned} \tag{4.31}$$

with $V_{RA}(x,T) = l(x)$, where we recall $l(x)$ is the signed distance to $\mathcal{T}$. The safe set for the reach avoid game $\Omega_{RA}$ is the set of states for which player $b$ can win the game, i.e. player $a$ does not reach the goal and player $b$ does not hit the obstacle.

In the pursuit-evader game player $a$ has the safety specification, namely to not get captured or hit an obstacle. Therefore, the target set is $\mathcal{T} = \mathcal{C} \cup \Psi_a$, and the solution to the game solves

$$\begin{aligned}\max\Big\{ &\min\{l(x) - V_{PE}(x,t), \frac{\partial V_{PE}(x,t)}{\partial t} + \max_{d\in\mathcal{D}} \min_{u\in\mathcal{U}} \frac{\partial V_{PE}(x,t)}{\partial x} \cdot f(x,u,d)\},\\ &- \omega_b(x) - V_{PE}(x,t)\Big\} = 0, \quad \forall t \in [0,T]\end{aligned} \tag{4.32}$$

with $V_{PE}(x,T) = l(x)$, and where $\omega_b : \mathbb{R}^n \to \mathbb{R}$ is a Lipschitz continuous function such that $\Psi_b = \{x \mid \omega_b(x) < 0\}$. A typical choice for $\omega_b$ is the signed distance function $s_{\Psi_b}$. Note that now the max is being taken over $\mathcal{D}$ since player $a$ is trying to stay in the constraint set $\mathcal{K}$. The safe set for the pursuit-evader game $\Omega_{PE}$ is the set of initial states for which player $a$ can avoid capture and not hit the obstacle.

Let us consider a mixed reach-avoid pursuit-evader game. Player $a$ is allowed to attack the goal, as long as she can avoid capture, while player $b$ is allowed to pursue player $a$, as long as she can guard the goal. For player $a$ we have the following strategy

$$d = \begin{cases} d^*_{RA}(x), & \text{if } V_{PE}(x) > 0, \\ d^*_{PE}(x), & \text{otherwise,} \end{cases} \tag{4.33}$$

where $d^*_{PE}(x)$ is the *optimal evade action* and $d^*_{RA}(x)$ is the *optimal attack action* to the goal, which are both obtained as the maximizers to their respective games. Similarly for player $b$

$$u = \begin{cases} u^*_{PE}(x), & \text{if } V_{RA}(x) > 0, \\ u^*_{RA}(x), & \text{otherwise,} \end{cases} \tag{4.34}$$

where $u^*_{PE}(x)$ is the *optimal pursuit action* and $u^*_{RA}(x)$ is the *optimal defend action*, which are both obtained as the maximizers to their respective games.

The setup of the game is shown in Fig. 4.8. We first solve (4.31) and (4.32) with $\mathcal{U} = \{u \mid 0 \text{ m/s} \leq u_1 \leq 2 \text{ m/s}, -\pi \leq u_2 \leq \pi\}$ and $\mathcal{D} = \{d \mid 0 \text{ m/s} \leq d_1 \leq$

2 m/s, , $-\pi \leq d_2 \leq \pi\}$. In both games it is always optimal for the two players to apply their maximum velocities. In our simulations the maximum velocity of player $a$ is capped to 1 m/s, therefore, player $b$ has a conservative control law (4.34) since the reachability analysis assumes player $a$ has a maximum velocity of 2 m/s. In the first simulation we allow the game to play out without modification to the safe set. Within five seconds it is evident that the game will end in a draw (player $a$ does not reach the goal, nor is she captured). Though player $a$ is relatively much slower, player $b$ still "believes" player $a$ may be just as fast. Player $a$ has a fixed feedback strategy (4.33) since $V_{PE}$ is not changed. From the perspective of player $b$, player $a$ is a deterministic disturbance in the dynamics. However, this disturbance is not as adversarial as the range of disturbances considered during the reachability analysis. Thus $V_{RA}$ characterizes an under-approximation of the safe set for the deterministic system. In the second simulation we employ our algorithm to come up with better estimates of the optimal safety value function by updating $V_{RA}$ given the observations from the game. The update rule (4.27) is slightly modified,

$$\vec{V}_{RA}^{k+1} \leftarrow \vec{V}_{RA}^{k} + \alpha\phi(x_k)\left(\max\left\{\,\min\{l(x_k), \phi(x_k^+)\cdot\vec{V}_{RA}^{k}\}, -\omega_a(x_k)\right\} - \phi(x_k)\cdot\vec{V}_{RA}^{k}\right), \tag{4.35}$$

where $\vec{V}_{RA}$ contains the grid node values for the reach-avoid game.

The updates enlarge the safe set of player $b$ during the simulation. In the second column of Fig. 4.9 we can see that the orange curve (player $b$'s learned safe set) has receded away from the cyan curve (player $b$'s initial safe set). The two curves are different because of the updates that were done during prior visits to that region of the state space. In the particular sequence shown in the second column player $a$ is on the boundary of the cyan curve, if the control law was determined by the initial safe set player $b$ would opt to defend the goal, just as what is shown in the first column. However, by using the orange curve (learned safe set) player $b$ has more freedom to pursue. After visiting that region of the state space a few times and updating the safe set, player $b$ eventually captures player $a$. In total it takes about 3.4 seconds for the capture.

It should be noted that the disturbance is not actually being learned. Rather, we are learning the safety value of a given state directly from our observations. In the unvisited regions of the state space player $b$ still "believes" player $a$ has a maximum velocity of 2 m/s, thus the decision to capture player $a$ is made only based on the local information acquired from the data. One could also use the data to infer the dynamics of both players and then recompute the safe set entirely. However, with 31 grid points in each dimension, computing the safe set using the Level Set Toolbox takes 26 minutes on a MacBook Pro with a Core i7 processor. It takes about 0.17 seconds to do one local update (over a neighborhood) in Matlab. Note that relative to these durations, player $a$ is captured in seconds.

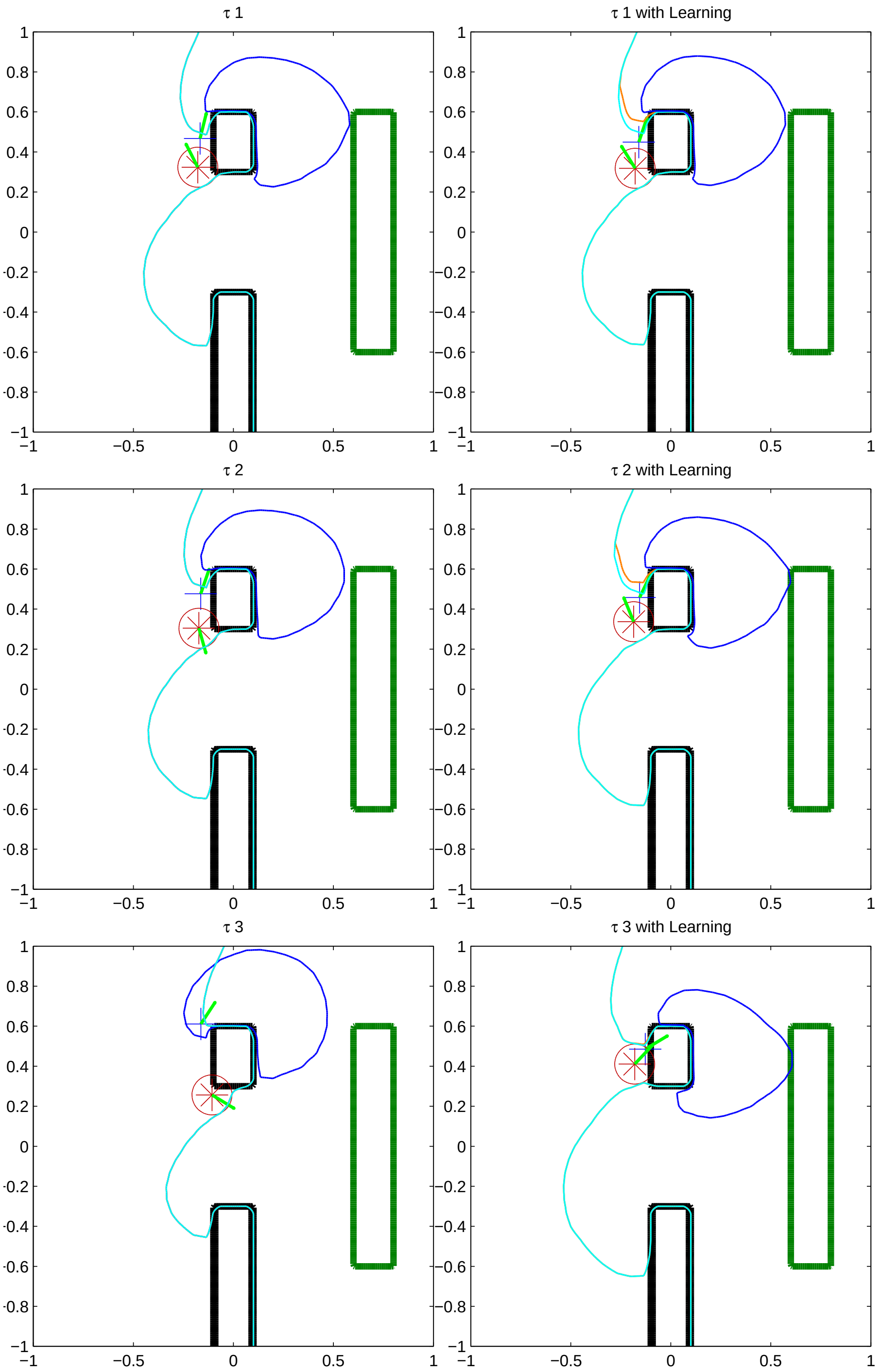


Figure 4.9: These snapshots show player $b$'s sequence of actions in a similar scenario with (2nd column) and without (1st column) updating its safe set. In the first column player $b$ chooses to defend the goal because player $a$ is on the boundary of player $b$'s initial safe set (given by the cyan curve). In the first two images of the 2nd column player $a$ is on the boundary of the cyan curve, which would have resulted in player $b$ defending the goal. However, player $a$ has not reached the orange curve (player $b$'s learned safe set), which has receded from the cyan curve due to the learning algorithm. Thus player $b$ continues to pursue and eventually captures player $a$.

# Chapter 5

# Reachability Analysis with Discounting

*This chapter is adapted from the material presented in [3].*

Thus far we have focused on applying reachability analysis, first for computing safe sets in the face of model uncertainty and then for learning safe sets from system observations. In this section we will turn our attention to the actual analysis itself and develop a novel formulation for conducting reachability analysis. The following discussion will apply more broadly than the safe learning context considered thus far, however some of the motivation comes from the work presented thus far, in particular the work presented on learning safe sets. We will look at the connection between the new formulation and learning safe sets in Section 5.5, but for now we begin with a broad perspective. Instead of referring to the safe set, we will primarily make reference to the more general reachable set. We also place emphasis here on the target set $\mathcal{T}$ over the state-constraint set $\mathcal{K}$.

It should be noted that this chapter is somewhat disjoint from the previous chapters so some of the notation will be reused including: $\lambda, \gamma,$ and $\mu$.

## 5.1 On Hamilton-Hacobi Reachability and Contraction Mappings

As shown in Section 2.2.2, the standard reachability formulation solves for the minimum distance to the target set $\mathcal{T}$, by computing the viscosity solution to the HJ equation in (2.6). This can be thought of more generally as a *minimum reward* optimal control problem or differential game. Like most optimal control problems, the solution is solved approximately on a grid via value iteration, which recursively applies a backup operator until convergence. An example of this procedure was highlighted in Section 2.2.3, and we encourage the reader to review that section before continuing to the next section.

One downside of MR optimal control problems is that the backup operator in this setting is not a contraction mapping, thus a specific initialization of value iteration is required for convergence to the correct solution. A contraction mapping allows for arbitrary initialization, and with a good initialization convergence can be accelerated. This would be particularly useful when computing reachable sets for multiple system models with similar dynamics allowing for the solution of one problem to be used as an initialization for the other. This is particularly the case when learning reachable (safe) sets via model-based techniques, in which we generate a sequence of models from system observations.

On the other hand, infinite horizon *sum of discounted reward* (SDR) optimal control problems yield backup operators that are contraction mappings [18]. This allows for more efficient solution methods, like policy iteration [47], [72] and multigrid approaches [5], [25].

Drawing inspiration from SDR problems, we propose a *minimum discounted reward* (MDR) formulation for computing tight over- and under-approximations of infinite horizon reachable sets. This new formulation yields a contraction mapping, making it possible to extend policy iteration and multigrid approaches to this setting. In addition, it also provides a way forward for learning reachable sets when dynamics are unknown or difficult to model, a topic that has garnered some attention recently [4], [31]. This draws greatly from Reinforcement Learning (RL), which can be thought of as solving the infinite horizon SDR problem when the system model is unknown. Many RL algorithms boil down to finding the fixed-point of the backup operator associated with SDR. The techniques developed in RL, like temporal differencing [83] and Q-learning [90], can naturally be extended to do the same in the MDR setting, thus facilitating research in learning reachable sets.

The rest of the chapter is organized as follows. In Section 5.2 we briefly go over SDR problems. In Section 5.3 we describe the MDR formulation for HJ reachability, and prove some relevant results. In Section 5.4 we explore policy iteration and multigrid approaches within the context of MDR problems. Section 5.5 draws inspiration from RL and presents some preliminary ideas on how the formulation can be used for learning reachable sets. Section 5.6 contains examples demonstrating the ideas developed throughout the paper. Again, the reader is encouraged to revisit Section 2.2.3 to recap the MR HJ reachability formulation.

Here, both the one-input and adversarial two-input settings are considered, and we opt to use the terminology from optimal control for consistency. For example *optimal control* will be used to refer to both settings, and we use terms like *reward* instead of *payoff* throughout.[1]

[1] Typically the terms *optimal control* and *reward* are reserved for the one player setting and *zero sum differential game* and *payoff* are the analogous terms in the adversarial two-player setting.

## 5.2 On Sum of Discounted Rewards and Contraction Mappings

Value iteration converges to a unique solution, independent of the initialization, when the backup operator is a *contraction mapping.*

**Definition 4** *A mapping $M(\cdot) : \mathbb{R}^{n_G} \to \mathbb{R}^{n_G}$, is said to be a contraction mapping in the norm $||\cdot||$ over the space $\mathbb{R}^{n_G}$ if there exists a Lipschitz constant $0 \leq \kappa < 1$ such that for any $\vec{A}_1, \vec{A}_2 \in \mathbb{R}^{n_G}$, $||M(\vec{A}_1) - M(\vec{A}_2)|| \leq \kappa||\vec{A}_1 - \vec{A}_2||$.*

Any contraction mapping has a unique fixed point. The operator given by (2.10) is not a contraction mapping. To see this note that any vector $\alpha\vec{1} \in \mathbb{R}^{n_G}$ is a fixed point for $\alpha < -L$.

The SDR problem does yield a contraction mapping, and we explore it briefly for the one player case. The objective is to maximize the integral (sum) of exponentially discounted rewards. In an abuse of notation, the associated value function $V(x)$ for this problem is given by

$$V(x) := \sup_{\boldsymbol{u} \in \mathbb{U}} \int_0^\infty r\big(\xi_x^{\boldsymbol{u}}(t)\big) \exp(-\lambda t) dt, \quad \lambda > 0, \tag{5.1}$$

where $r(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is a state-dependent reward function, and $\lambda$ is a *discount rate.*

It is known that this value function is the solution to the time-independent Hamilton-Jacobi equation [11],

$$\lambda V(x) = \max_{u \in \mathcal{U}} \frac{\partial V}{\partial x}(x) f(x,u) + g(x). \tag{5.2}$$

Using the same semi-Lagrangian scheme as in the MR setting (see Section 2.2.3), equation (5.2) can be approximated as

$$V_{\Delta t}(x) = \max_{u \in \mathcal{U}} \gamma V_{\Delta t}(x + \Delta t \cdot f(x,u)) + \Delta t \cdot r(x), \tag{5.3}$$

where $\gamma = \exp(-\lambda \Delta t)$ is the *discount factor.*

The approximation is solved for using value iteration with the following backup operator:

$$B[\vec{A}] := \vec{r} + \max_{\pi} \gamma \Phi_\pi \vec{A}, \tag{5.4}$$

where $\vec{r}_i = \Delta t \cdot r(x_i)$. The initialization of the value iteration procedure is arbitrary $\vec{V}^0 \in \mathbb{R}^{n_G}$ because the backup operator in (5.4) is a contraction mapping in the infinity norm, $||\cdot||_\infty$ [18].

Discounting reduces the impact of future rewards on the outcome of the trajectory, and ultimately yields a contraction mapping. In fact, the discount factor $\gamma$ is the Lipschitz constant of the contraction mapping.

## 5.3 Minimum of Discounted Rewards Hamilton-Jacobi Reachability Analysis

Inspired by the SDR setting, we now present the MDR optimal control problem. Moving forward any mention of value function $V(x)$ refers to the MR setting, and is defined by (2.4), (2.5), and (2.6).

### 5.3.1 What to discount?

A natural proposal for the outcome of the MDR problem would be

$$\inf_{t\geq 0} l\big(\xi_x^{\boldsymbol{u},\boldsymbol{d}}(t)\big)\exp(-\lambda t). \tag{5.5}$$

However, there is an issue with defining this outcome. Recall that discounting makes rewards contribute less to the outcome the further they occur in the future. Since we take an infimum, the discounted reward should become more positive the further it occurs in the future making it less likely to be selected by the infimum. This only works if the reward is nonpositive everywhere, which is not the case for $l$, since it is a clipped signed distance. This is easily fixed with the following outcome for the MDR problem

$$\mathcal{Z}\big(x,\boldsymbol{u}(\cdot),\boldsymbol{d}(\cdot)\big) := L + \inf_{t\geq 0}(l\big(\xi_x^{\boldsymbol{u},\boldsymbol{d}}(t)\big) - L)\exp(-\lambda t). \tag{5.6}$$

The quantity in the infimum is always nonpositive since $L$ upperbounds $l(x)$ by construction. Note that if $\lambda = 0$, i.e. no discounting, then we have the minimum reward outcome (2.4).

We define the MDR value function as

$$Z(x) := \inf_{\beta[\boldsymbol{u}](\cdot)\in\mathcal{B}} \sup_{\boldsymbol{u}\in\mathbb{U}} \mathcal{Z}\big(x,\boldsymbol{u}(\cdot),\beta[\boldsymbol{u}](\cdot)\big). \tag{5.7}$$

For convenience we define two functions $U(x) := Z(x) - L$ and $h(x) := l(x) - L$. We will show that $U(x)$ is the viscosity solution to a particular time-independent HJ equation

$$0 = \min\left\{h(x) - U(x), \max_{u\in\mathcal{U}}\min_{d\in\mathcal{D}}\frac{\partial U}{\partial x}(x)f(x,u,d) - \lambda U(x)\right\}. \tag{5.8}$$

We begin by presenting some Lemmas to facilitate the proof.

**Lemma 1** *The function $U(x)$ is well defined.*

**Proof 7** *Define the sequence $\{U(x,k)\}_k$, where*

$$U(x,k) = \inf_{\beta[\boldsymbol{u}](\cdot)\in\mathcal{B}} \sup_{\boldsymbol{u}\in\mathbb{U}} \inf_{t\in[0,k\Delta t]}(h\big(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(t)\big))\exp(-\lambda t), \tag{5.9}$$

*and $\Delta t > 0$. The sequence is nonincreasing, since $U(x, k+1) \leq U(x, k)$, and is lower bounded by $-2L$, so it converges. Clearly in the limit this sequence also equals $U(x)$.*

**Lemma 2** Dynamic programming principle. *For $\delta > 0$,*

$$U(x) = \inf_{\beta[\boldsymbol{u}](\cdot)\in\mathcal{B}} \sup_{\boldsymbol{u}\in\mathbb{U}_\delta} \big[\min\{ \inf_{t\in[0,\delta]} h\big(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(t)\big)\exp(-\lambda t), \atop \exp(-\lambda\delta)U(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta))\}\big] , \tag{5.10}$$

*where $\mathbb{U}_\delta$ consists of measurable functions on the interval $[0, \delta]$.*

**Proof 8** *Splitting the time interval of the infimum in (5.6) into $[0, \delta]$ and $t > \delta$, $U(x)$ can be expressed as*

$$U(x) = \inf_{\beta[\boldsymbol{u}](\cdot)\in\mathcal{B}} \sup_{\boldsymbol{u}\in\mathbb{U}} \big[\min\{ \inf_{t\in[0,\delta]} h\big(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(t)\big)\exp(-\lambda t), \atop \inf_{t>\delta} h\big(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(t)\big)\exp(-\lambda t)\}\big] . \tag{5.11}$$

*Due to time-invariance of the dynamics, if we define $s = t - \delta$, $u_\delta(s) = u(t + \delta)$ and $y = \xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta)$,*

$$U(x) = \inf_{\beta[\boldsymbol{u}](\cdot)\in\mathcal{B}} \sup_{\boldsymbol{u}\in\mathbb{U}} \big[\min\{ \inf_{t\in[0,\delta]} h\big(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(t)\big)\exp(-\lambda t), \atop \exp(-\lambda\delta)\inf_{s>0} h\big(\xi_y^{\boldsymbol{u}_\delta,\beta[\boldsymbol{u}_\delta]}(s)\big)\exp(-\lambda s)\}\big] . \tag{5.12}$$

*The game over the second interval can be optimized independently of the first interval once $y = \xi_x^{\boldsymbol{u},\boldsymbol{d}}(\delta)$ is specified thus it can be replaced with $\exp(-\lambda\delta)U(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta))$. Furthermore $\mathbb{U}$ is replaced with $\mathbb{U}_\delta$ since the game is only played explicitly on $[0, \delta]$.*

Before presenting the proof of the viscosity solution, it will be necessary to introduce the concepts of *viscosity subsolution* and *viscosity supersolution.* To simplify notation, we first define the *Hamiltonian* $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$,

$$H(x, p) = \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} f(x, u, d) \cdot p. \tag{5.13}$$

**Definition 5** *A function $\phi$ (in this case $U$) on $\mathbb{R}^n$ is a viscosity subsolution of (5.8), if for all $\psi \in C^1(\mathbb{R}^n)$ and $x_0$ such that $\phi(x_0) = \psi(x_0)$ and $x_0$ attains a local maximum on $\phi - \psi$, then*

$$\min\left\{h(x_0) - \psi(x_0), H(x_0, \frac{\partial\psi}{\partial x}) - \lambda\psi(x_0)\right\} \geq 0. \tag{5.14}$$

**Definition 6** *A function $\phi$ (in this case $U$) on $\mathbb{R}^n$ is a supersolution of (5.8), if for all $\psi \in C^1(\mathbb{R}^n)$ and $x_0$ such that $\phi(x_0) = \psi(x_0)$ and $x_0$ attains a local minimum on $\phi - \psi$, then*

$$\min\left\{h(x_0) - \psi(x_0), H(x_0, \frac{\partial\psi}{\partial x}) - \lambda\psi(x_0)\right\} \leq 0. \tag{5.15}$$

Now we present the major theoretical result for this section.

**Theorem 1** *The function $U(x)$ is the unique viscosity solution to the time-independent HJ equation given by* (5.8)*.*

**Proof 9** *The structure of the proof follows the classical approach in [33], analogously to [36], and draws from viscosity solution theory. We start by assuming that $U$ is not a viscosity solution and then derive a contradiction to Lemma 2.*

*A continuous function is a viscosity solution if it is both a* subsolution *and* supersolution*. Note that $U$ is uniformly continuous due to the continuity assumptions on $f$ and $l$. We first show $U$ is a subsolution of* (5.8)*.*

*From the local maximum condition in Definition 5 and continuity of system trajectories, there exists a sufficiently small $\delta > 0$, such that for $\tau \in [0, \delta]$*

$$U(\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}(\tau)) \leq \psi(\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}(\tau))$$

*for all $u(\cdot) \in \mathbb{U}$ and $d(\cdot) \in \mathbb{D}$.*

*For sake of contradiction, assume* (5.14) *is false, then one of the following must be true*

$$h(x_0) = \psi(x_0) - \epsilon_1 \tag{5.16a}$$

$$H(x_0, \frac{\partial \psi}{\partial x}) - \lambda \psi(x_0) = -\epsilon_2, \tag{5.16b}$$

*for some $\epsilon_1, \epsilon_2 > 0$. If* (5.16a) *is true, then*

$$h(\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}(\tau)) \exp(-\lambda \tau) \leq \psi(x_0) - \frac{\epsilon_1}{2} = U(x_0) - \frac{\epsilon_1}{2} \tag{5.17}$$

*Incorporating this into the dynamic programming principle (Lemma 2), we have*

$$\begin{aligned} U(x_0) &\leq \inf_{\beta[\boldsymbol{u}](\cdot) \in \mathcal{B}} \sup_{\boldsymbol{u} \in \mathbb{U}_\delta} \big\{ \inf_{t \in [0,\delta]} h\big(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(t)\big) \exp(-\lambda t) \big\} \\ &\leq U(x_0) - \frac{\epsilon_1}{2}, \end{aligned} \tag{5.18}$$

*which is a contradiction since $\epsilon_1 > 0$. Similarly, if* (5.16b)*, then for a small enough $\delta > 0$ and some nonanticipative strategy $\beta[\cdot]$,*

$$H(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau), \frac{\partial \psi}{\partial x}) - \lambda \psi(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau)) \leq \frac{\epsilon_2}{2} \tag{5.19}$$

*for all $\tau \in [0, \delta]$ and all inputs $\boldsymbol{u}(\cdot) \in \mathbb{U}$. Due to the* max *in* (5.13)*, for $\tau \in [0, \delta]$*

$$\begin{aligned} &f(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau), \boldsymbol{u}(\tau), \beta[\boldsymbol{u}](\tau)) \cdot \frac{\partial \psi}{\partial x} - \lambda \psi(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau)) \leq \\ &H(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau), \frac{\partial \psi}{\partial x}) - \lambda \psi(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau)). \end{aligned} \tag{5.20}$$

*Combining the two previous inequalities and integrating over the interval* $[0, \delta]$ *we have*

$$\exp(-\lambda\delta)\psi(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta)) - \psi(x_0) \leq \frac{\epsilon_2}{2}\delta \tag{5.21}$$

*Recalling that* $U(x_0) = \psi(x_0)$

$$\exp(-\lambda\delta)U(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta)) \leq \frac{\epsilon_2}{2}\delta + U(x_0) \tag{5.22}$$

*Incorporating this into the dynamic programming principle, we have*

$$U(x_0) \leq \exp(-\lambda\delta)U(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta)) \leq \frac{\epsilon_2}{2}\delta + U(x_0), \tag{5.23}$$

*which is a contradiction, thus we conclude that* $U$ *is a subsolution.*

*Next we show that* $U$ *is a supersolution. From the local minimum condition in Definition 6 and continuity of system trajectories, there exists a sufficiently small* $\delta > 0$*, such that for* $\tau \in [0, \delta]$

$$U(\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}(\tau)) \geq \psi(\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}(\tau))$$

*for all* $u(\cdot) \in \mathbb{U}$ *and* $d(\cdot) \in \mathbb{D}$*.*

*If we suppose* (5.15) *is false, then both of the following must hold*

$$h(x_0) = \psi(x_0) + \epsilon_1 \tag{5.24a}$$

$$H(x_0, \frac{\partial\psi}{\partial x}) - \lambda\psi(x_0) = \epsilon_2, \tag{5.24b}$$

*for some small* $\epsilon_1, \epsilon_2 > 0$*. If* (5.24a) *is true, then*

$$\begin{aligned} h(\xi_{x_0}^{\boldsymbol{u},\boldsymbol{d}}(\tau))\exp(-\lambda\tau) &\geq \psi(x_0) + \frac{\epsilon_1}{2} \\ &= U(x_0) + \frac{\epsilon_1}{2} \end{aligned} \tag{5.25}$$

*Similarly, if* (5.16b)*, then for small enough* $\delta > 0$ *and some input* $\boldsymbol{u}(\cdot) \in \mathbb{U}$

$$H(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau), \frac{\partial\psi}{\partial x}) - \lambda\psi(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau)) \geq \frac{\epsilon_2}{2} \tag{5.26}$$

*for all* $\tau \in [0, \delta]$ *and all nonanticipative strategies* $\beta[\cdot]$*.*

*Due to the* min *in* (5.13)*, for* $\tau \in [0, \delta]$

$$\begin{aligned} &f(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau), \boldsymbol{u}(\tau), \beta[\boldsymbol{u}](\tau)) \cdot \frac{\partial\psi}{\partial x} - \lambda\psi(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau)) \geq \\ &H(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau), \frac{\partial\psi}{\partial x}) - \lambda\psi(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\tau)). \end{aligned} \tag{5.27}$$

*Combining the two previous inequalities and integrating over the interval* $[0,\delta]$ *we have*

$$\exp(-\lambda\delta)\psi(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta)) - \psi(x_0) \geq \frac{\epsilon_2}{2}\delta \tag{5.28}$$

*Recalling that* $U(x_0) = \psi(x_0)$

$$\exp(-\lambda\delta)U(\xi_{x_0}^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta)) \geq \frac{\epsilon_2}{2}\delta + U(x_0). \tag{5.29}$$

*Incorporating this into the dynamic programming principle, we have*

$$\begin{aligned} U(x) = \inf_{\beta[\boldsymbol{u}](\cdot)\in\mathcal{B}} \sup_{\boldsymbol{u}\in\mathbb{U}_\delta} \big[ \min\{ &\inf_{t\in[0,\delta]} h\big(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(t)\big)\big) \exp(-\lambda t), \\ &\exp(-\lambda\delta)U(\xi_x^{\boldsymbol{u},\beta[\boldsymbol{u}]}(\delta))\}\big] \geq U(x) + \min\{\frac{\epsilon_1}{2}, \frac{\epsilon_2}{2}\delta\}, \end{aligned} \tag{5.30}$$

*which is a contradiction, thus* $U$ *is also a supersolution.*

*Since we have shown that* $U$ *is both a viscosity subsolution and viscosity supersolution of the HJ equation, this completes the proof that* $U$ *is a viscosity solution of* (5.8)*. Uniqueness follows from the classical comparison and uniqueness theorems for viscosity solutions (see Theorem 4.2 in [14]).*

### 5.3.2 Computing the Discounted Value Function

The discrete approximation of (5.8) is given by

$$U_{\Delta t}(x) = \min\left\{ h(x), \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} \gamma U_{\Delta t}(x + \Delta t f(x,u,d)) \right\}, \tag{5.31}$$

which can be solved on a grid $G$ via value iteration

$$\vec{U}^0 \in \mathbb{R}^{n_G}, \tag{5.32a}$$

$$\vec{U}^{k+1} = B[U^k], \tag{5.32b}$$

$$\vec{U} = \lim_{k\to\infty} \vec{U}^k, \tag{5.32c}$$

with the backup operator defined as

$$B[\vec{A}] := \min\left\{ \vec{h}, \max_{\pi} \min_{\rho} \gamma \Phi_{\pi,\rho} \vec{A} \right\}, \tag{5.33}$$

where $\vec{h}_i = h(x_i)$. The MDR value function $Z(x)$ is then approximated by $I[\vec{U}](x)+L$, where again $I[\vec{U}](\cdot)$ is the interpolation operator. We now prove that (5.33) is a contraction.

**Lemma 3** *For any two functions $q, g : A \times B \to \mathbb{R}$,*

$$|\max_a \min_b q(a,b) - \max_a \min_b g(a,b)| \leq \max_a \max_b |q(a,b) - g(a,b)|. \tag{5.34}$$

**Proof 10** *Define the minimax optimizers for $q$ as the pair $(a_q, b_q)$, and minimax optimizers of $g$ as the pair $(a_g, b_g)$. Without loss of generality we assume that $q(a_q, b_q) \geq g(a_g, b_g)$. We then have the following inequalities:*

$$\begin{aligned} &|\max_a \min_b q(a,b) - \max_a \min_b g(a,b)| \leq |q(a_q, b_q) - \min_b g(a_q, b)| \\ &\leq |q(a_q, b_{gg}) - g(a_q, b_{gg})| \leq \max_a \max_b |q(a,b) - g(a,b)|, \end{aligned}$$

*with $b_{gg} := \arg\min_b g(a_q, b)$.*

**Theorem 2** *The operator given by (5.33) is a contraction mapping in the infinity norm $|| \cdot ||_\infty$ on the space $\mathbb{R}^{n_G}$.*

**Proof 11** *Defining $B[\cdot]$ as in (5.33), take $A_1, A_2 \in \mathbb{R}^{n_G}$:*

$$\begin{aligned} &||B[\vec{A}_1] - B[\vec{A}_2]||_\infty = \\ &||\min\left\{\vec{h}, \max_\pi \min_\rho \gamma \Phi_{\pi,\rho} \vec{A}_1\right\} - \min\left\{\vec{h}, \max_\pi \min_\rho \gamma \Phi_{\pi,\rho} \vec{A}_2\right\}||_\infty. \end{aligned}$$

*Leveraging the identity $\min\{a,b\} = \frac{1}{2}((a+b) - |a-b|)$ and using the shorthand $\Pi[\vec{A}] = \max_\pi \min_\rho \gamma \Phi_{\pi,\rho} \vec{A}$ , the above is equal to*

$$\frac{1}{2}||(\Pi[\vec{A}_1] - \Pi[\vec{A}_2]) - (|\Pi[\vec{A}_1] - \vec{h}| - |\Pi[\vec{A}_2] - \vec{h}|)||_\infty,$$

*which by the triangle inequality, is upper bounded by*

$$\frac{1}{2}||(\Pi[\vec{A}_1] - \Pi[\vec{A}_2])||_\infty + \frac{1}{2}||(|\Pi[\vec{A}_1] - \vec{h}| - |\Pi[\vec{A}_2] - \vec{h}|)||_\infty.$$

*Given the inequality $|a - b| > |(|a| - |b|)|$, this has upper bound*

$$\begin{aligned} &||(\Pi[\vec{A}_1] - \Pi[\vec{A}_2])||_\infty = \\ &||\max_\pi \min_\rho \gamma \Phi_{\pi,\rho} \vec{A}_1 - \max_\pi \min_\rho \gamma \Phi_{\pi,\rho} \vec{A}_2||_\infty. \end{aligned}$$

*Finally from Lemma 3, the last upper bound is*

$$\max_\pi \max_\rho ||\gamma \Phi_{\pi,\rho}(\vec{A}_1 - \vec{A}_2)||_\infty \leq \gamma ||\vec{A}_1 - \vec{A}_2||_\infty,$$

*where the last inequality comes from the fact that $\Phi_{\pi,\rho}$ is a stochastic matrix for all policies, thus $||\Phi_{\pi,\rho}||_\infty = 1$.*

### 5.3.3 Under- and Over-Approximating the Reachable Set

With MDR formulation there is no particular level curve of the value function that characterizes the reachable set. However, it is possible to find level curves that correspond to over and under approximations of the reachable set.

We have the inequality $Z(x) \geq V(x)$ because the terms being discounted in the outcome are nonpositive. It immediately follows that

$$\{x \mid V_\lambda(x) \leq 0\} \subseteq \mathcal{R}(\mathcal{T}), \tag{5.35}$$

For an over-approximation we first need to characterize the error between $Z(x)$ and $V(x)$. The difference between the two functions can be bounded. Define $\tau(x)$ as the time when the minimum distance to the target is achieved for a trajectory starting at state $x$ under the optimal control and disturbance signals. Then we have the following bound

$$Z(x) - V(x) \leq (L - l(\xi_x^{\boldsymbol{u},\boldsymbol{d}}(\tau(x))))(1 - \exp(-\lambda\tau(x))). \tag{5.36}$$

Noting that $V(x) = l(\xi_x^{\boldsymbol{u},\boldsymbol{d}}(\tau(x)))$, we get the resulting inequality

$$Z(x) - V(x)\exp(-\lambda\tau(x)) \leq L(1 - \exp(-\lambda\tau(x))), \tag{5.37}$$

Furthermore, outside the reachable set $V(x) > 0$ leading to

$$Z(x) - V(x) \leq L(1 - \exp(-\lambda\tau(x))) \quad \forall x \notin \mathcal{R}(\mathcal{T}). \tag{5.38}$$

Assuming an upper bound $\bar{\tau} \geq \tau(x)$, we have the following over-approximation for the reachable set

$$\mathcal{R}(\mathcal{T}) \subseteq \{x \mid Z(x) \leq L(1 - \exp(-\lambda\bar{\tau}))\}. \tag{5.39}$$

It is clear from (5.37) that the tightness of the approximations can be tuned via the discount rate $\lambda$.

## 5.4 Improving Convergence

In this section we present methods that may yield much faster convergence than value iteration. These approaches have been extensively applied to the SDR setting, and we now apply them to the MDR problem.

### 5.4.1 Policy Iteration

In the one player setting (control only), if the backup operator is a contraction mapping the solution can also be obtained via *policy iteration.* First define the *policy backup operator* $B^\pi[\cdot] : \mathbb{R}^{n_G} \to \mathbb{R}^{n_G}$,

$$B^\pi[\vec{A}] = \min\left\{\vec{h}, \gamma\Phi_\pi\vec{A}\right\}. \tag{5.40}$$

This operator is a contraction mapping. The policy iteration algorithm generates the sequence $\{\vec{U}^{\pi^k}\}_k$ according to

$$\vec{U}^{\pi^k} = B^{\pi^k}[\vec{U}^{\pi^k}], \tag{5.41a}$$

$$\pi^{k+1} = \arg\max_{\pi} B^{\pi}[\vec{U}^{\pi^k}], \tag{5.41b}$$

$$\vec{U} = \lim_{k\to\infty} \vec{U}^{\pi^k}. \tag{5.41c}$$

Note that $\vec{U}^{\pi^k}$, the fixed point of (5.41a), is obtained through value iteration with the policy backup. Finding the fixed point of the policy backup is computationally less intensive than the other backup operators presented thus far, since no optimization is performed over policies. The policy iteration algorithm only optimizes over policies when switching policies.

In practice policy iteration is typically recommended over value iteration because policies can converge faster than values resulting in faster convergence of the algorithm [76]. A more detailed analysis of policy iteration (as it pertains to SDR) can be found in [20], [47], [72].

We now prove that policy iteration converges for the MDR problem.

**Proposition 9** *Assuming a finite control set $\mathcal{U} = \{u_i\}_{i=1}^{n_\mathcal{U}}$, the policy iteration algorithm converges to the vectorized value function obtained from* (5.32) *without the disturbance.*

**Proof 12** *It's sufficient to show that sequence $\{\vec{U}^{\pi^k}\}_k$ is nondecreasing, i.e. $\vec{U}^{\pi^{k+1}} \geq \vec{U}^{\pi^k}$ $\forall k$. Since the number of policies is finite the nondecreasing criterion implies that the sequence of vectors will converge. Also note that $\max_{\pi} B^{\pi}[\cdot] = B[\cdot]$ as defined in* (5.32) *without the disturbance, so the sequence converges to the vectorized value function.*

*Consider two sequences $\vec{X}^{i+1} = \min\left\{\vec{U}^{\pi^k}, \gamma\Phi_{\pi^{k+1}}\vec{X}^i\right\}$ and $\vec{Y}^{i+1} = \min\left\{\vec{h}, \gamma\Phi_{\pi^{k+1}}\vec{Y}^i\right\}$, with $\vec{X}^0 = \vec{Y}^0 = \vec{U}^{\pi^k}$. Since $\vec{h} \geq \vec{U}^{\pi^k}$, by* (5.41a)*, we have $\vec{Y}^i \geq \vec{X}^i, \forall i \geq 0$. Next, we note that $\vec{X}^1 = \min\left\{\vec{U}^{\pi^k}, \gamma\Phi_{\pi^{k+1}}\vec{U}^{\pi^k}\right\} = \min\left\{\vec{h}, \gamma\Phi_{\pi^k}\vec{U}^{\pi^k}, \gamma\Phi_{\pi^{k+1}}\vec{U}^{\pi^k}\right\}$. Furthermore, the third term in the $\min\{\,\}$ is greater than the second, so $\vec{X}^1 = \min\left\{\vec{h}, \gamma\Phi_{\pi^k}\vec{U}^{\pi^k}\right\} = \vec{U}^{\pi^k}$, thus $\vec{X}^i = \vec{U}^{\pi^k}, \forall i \geq 0$. Lastly, $\lim_{i\to\infty}\vec{Y}^i = \vec{U}^{\pi^{k+1}}$, which is the fixed-point of the contraction mapping that generates the sequence. Bringing everything together we have $\vec{U}^{\pi^{k+1}} = \lim_{i\to\infty}\vec{Y}^i \geq \lim_{i\to\infty}\vec{X}^i = \vec{U}^{\pi^k}$.*

### 5.4.2 Multigrid Approach

The accuracy of the approximation scheme depends on the fineness of the discretization. Finer grids have lower approximation error, but at the cost of increased computational effort. For a desired level of accuracy the number of grid points (and thus computation) necessary grows exponentially with the state space dimension.

One possible way to manage this trade-off between computation and accuracy is a multigrid approach. The idea is to first solve for the approximation on a coarse grid (e.g. grid spacing $2\Delta x_j$) and then use the final solution to initialize either value iteration or policy iteration on a finer grid. The procedure can also be stacked, i.e. given $m$ grids of increasing fineness we can produce approximations of increasing accuracy by using each as an initialization for the subsequent grid. This is only possible because contraction mappings allow great flexibility in the initialization, and yields faster convergence for good initializations. Due to the curse of dimensionality obtaining a good approximation is exponentially cheaper on the coarse grid.

Multigrid approaches have been applied extensively for SDR problems, where empirical and theoretical improvements have been shown [5], [25]. We compare multigrid approaches to value iteration in Section 5.6.

## 5.5 Learning Reachable (Safe) Sets Revisited

When the system model is unknown or complex, the reachable (safe) set must be learned from data. We first introduced the topic of learning reachable sets in Section 4, where the set was learned based on the standard MR reachability formulation. However, that formulation has its shortcomings since it does not yield a contraction mapping. Traditionally reachability analysis was performed once for a single model offline, and the value function was never modified once it was obtained. Learning the set is a paradigm shift–the value function is constantly changing and it can be done so online. In this context having a contraction mapping is beneficial for the model-based learning approach because as the model changes (due to new observations) the value function for the current model can be leveraged to compute the value function for the next model. The standard formulation essentially throws away this information in the current value function, even though the two might be quite similar, which is typically the case when the model estimate starts to converge. The model-free TD approach in Section 4.2.2 does better when it comes to leveraging old solutions, however the convergence of the solution is dependent on the initialization of the value function, which is a result of the backup operator having multiple fixed points.

In this section we will revisit the model-based and model-free approaches using the MDR formulation, and highlight the advantages of contraction mappings.

### 5.5.1 Model-based

In the model-based approach the model is assumed to be parameterized by a parameter vector $\mu$. The data is first used to fit the parameters, and then the value function is computed given the model. As more data is collected, the process can be repeated. This is the approach taken in [2], [43], and presented in Section 4.1. Here we are intentionally vague about the data and the fitting process, which can be done with a GP for example, and we focus our attention on how to obtain the value function given the fitted model.

The data collection and fitting produce a sequence of parameters $\{\mu_k\}_k$, which in turn corresponds to a sequence of models $\{f_{\mu_k}\}_k$ and vectorized value functions $\{\vec{V}_{\mu_k}\}_k$ for the MR setting and $\{\vec{U}_{\mu_k}\}_k$ for the MDR setting. With the MR formulation the value iteration algorithm must be initialized with $\vec{l}$ every time a new value function is computed. However, with the MDR formulation $\vec{U}_{\mu_k}$ can be used as the initialization when computing $\vec{U}_{\mu_{k+1}}$. Assuming regularity in the dynamics (with respect to $\mu$), if the parameters only deviate slightly between iterations then $\vec{U}_{\mu_k}$should be a good approximation of $\vec{U}_{\mu_{k+1}}$, resulting in faster convergence. If this is not the case then $\vec{l}$ can be used as the default initialization. Furthermore, the following classical result on contraction mappings can provide insight on selecting the initialization:

**Proposition 10** *If $M(\cdot) : \mathbb{R}^{n_G} \to \mathbb{R}^{n_G}$ is a contraction mapping in the norm $||\cdot||$ over the space $\mathbb{R}^{n_G}$ with Lipschitz constant $0 \leq \kappa < 1$ and fixed-point $\vec{A}^*$, then for any $\vec{A} \in \mathbb{R}^{n_G}$, $||\vec{A}^* - \vec{A}|| \leq \frac{1}{1-\kappa}||M(\vec{A}) - \vec{A}||$.*

Given Proposition 10, when computing $\vec{U}_{\mu_{k+1}}$an upper bound can be computed on its distance to $\vec{U}_{\mu_{k+1}}$ and $\vec{l}$ by applying the contraction mapping to each. The initialization can then be selected to minimize this upper bound. In the worst case only one additional backup operation is performed compared to the default case.

### 5.5.2 Model-free

*This section is meant only to introduce some ideas for future work. The ideas presented here are not investigated any further throughout the thesis.*

Another approach to handling an unknown model, is to compute the value function directly from the data. This is the approach taken in many RL algorithms that attempt to approximate the value function for SDR problems with unknown models.

Temporal difference (TD) learning is at the heart of many of these methods, which includes TD-lambda[83], Q-learning[90], Deep Q Networks[65], and actor-critic methods[57]. The TD method was introduced in Section for the RL problem, and it was extended in Section for learning reachable sets under the standard HJ reachability formulation. We invite the reader to revisit those sections. We now extend TD to the MDR setting.

For ease of presentation consider an autonomous system $f(x)$ (which might be due to a fixed policy), and a sequence of state transitions obtained from the system $\{(x_i, x_i^+)\}$, where $x^+ = \xi_x(\Delta t)$ and $\Delta t$ is the time step. Taking the approximation

$U_\theta(x)$ with parameter $\theta$, the loss function and update rule for the MDR setting would be

$$\begin{aligned}&\mathcal{L}(\theta) = \frac{1}{2}\big(U_\theta(x) - \min\{h(x), \gamma U_\theta(x^+)\}\big)^2,\\&\theta^{i+1} \leftarrow \theta^i + \alpha \nabla_\theta U_{\theta^i}(x_i)\big(\min\{h(x_i), \gamma U_{\theta^i}(x_i^+)\} - U_{\theta^i}(x_i)\big).\end{aligned} \tag{5.42}$$

As before control actions can be accounted for by pairing TD with policy iteration. A key thing to note the loss function here is less susceptible to local minimizers since the backup operator has a unique fixed-point. In fact, in the tabular case for discrete state spaces loss function has one global minimizer.

So far there has not been any analysis done on the convergence of TD for computing reachable sets, neither with the MR or MDR formulation. At the very least we can conclude that any convergence result that exists for the MR formulation will be initialization dependent since the backup operator used to generate the loss function has many fixed-points.

Producing convergence results for the MDR setting seems to be more promising because of the similarities that it shares with the sum of discounted rewards problem that shows up in RL. The two problem formulations ultimately produce contraction mappings, which is key because there is a large body of work on the convergence of TD for RL that relies primarily on the fact that the backup operator is a contraction mapping. These convergence proofs would be a good start for analyzing the convergence of TD for MDR problems.

In addition to the convergence analysis, it is also likely that the TD-based RL algorithms mentioned earlier, like Q-learning, Deep Q Networks, etc., can be used for learning reachable sets under the MDR formulation.

## 5.6 Experiments

This section uses two benchmark models, double integrator and pursuit-evasion game, to exemplify the numerical properties of the MDR formulation. The double integrator model will be used to display the over-/under-approximation of the reachable set, as well as to compare policy iteration with value iteration. Both of the benchmarks will be used to demonstrate the advantages of multigridding, and initializing value iteration with pre-computed solutions to similar problems.

Unless stated otherwise all algorithms are initialized with $\vec{h} = \vec{l} - L$, and are considered converged when the distance (in the infinity norm) between consecutive iterates falls below $\epsilon = .001$. All experiments were run on a 2016 MacBook Pro with Core i7 processor and 16GB RAM.

Table 5.1: Value Iteration vs Policy Iteration

| # actions | VI | Policy Iteration | |
|---|---|---|---|
| | $T_{total}$ | $T_{total}$ | $T_{total} - T_{\Phi_{\pi_u}}$ |
| 2 | 1.468 | 78.197 | 0.102 |
| 50 | 8.753 | 302.456 | 1.007 |
| 250 | 36.973 | 308.565 | 4.318 |
| 500 | 65.305 | 326.280 | 9.760 |

### 5.6.1 Double Integrator

The doube integrator consists of two states $(x_1, x_2)$ , and control $u \in [-u_{max}, u_{max}]$ with dynamics,

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= u \end{aligned} \tag{5.43}$$

The state space is discretized into a 161 × 161 grid on the domain $[-1, 5] \times [-5, 5]$, and $u_{max} = 2$.

The task is to keep the state trajectory inside the box $\mathcal{K} = [0, 4] \times [-3, 3]$, thus the target is taken to be its complement $\mathcal{T} = \bar{\mathcal{K}}$. Given the application we refer to the safe set in the following experiments.

We first show, in Fig. 5.1 that different level curves of $Z$over approximates (bold line) and under approximates (dotted lines) the analytic safe set (shown in black) for two values of $\lambda = 0.1, 0.2$, with $\bar{\tau} = 2$. As expected smaller values of $\lambda$, yield tighter approximations.

To verify the convergence properties of the MDR formulation we initialize value iteration with the zero vector $\vec{0} \in \mathbb{R}^{N_G}$ with $\lambda = 0.1$. The error (in the infinity norm) between the converged solution and the one visualized in Fig. 5.1 is 0.000299, suggesting convergence to the same fixed point. Under the MR setting value iteration fails to converge with this particular initialization.

We now compare value iteration and policy iteration with increasing number of discrete actions in Table 5.1. In the table we see that the runtime of value iteration increases linearly with the increase in the number of actions, and policy iteration scales much better. In our particular implementation, our overall runtime favors value iteration, but it is important to note that the majority of the time for policy iteration is spent constructing $\Phi_{\pi_u}$, which is denoted by $T_{\Phi_{\pi_u}}$.[2] Excluding this cost, policy iteration becomes more attractive.

Next, we look at a multigrid approach versus value iteration. For the multigrid approach we also need to run value iteration on a coarse grid, which we construct to

[2] The data structure used to represent the interpolation is very efficient for sparse matrix multiplication, but is not ideal for indexing, which is necessary to create $\Phi_{\pi_u}$, and results in a relatively large $T_{\Phi_{\pi_u}}$.

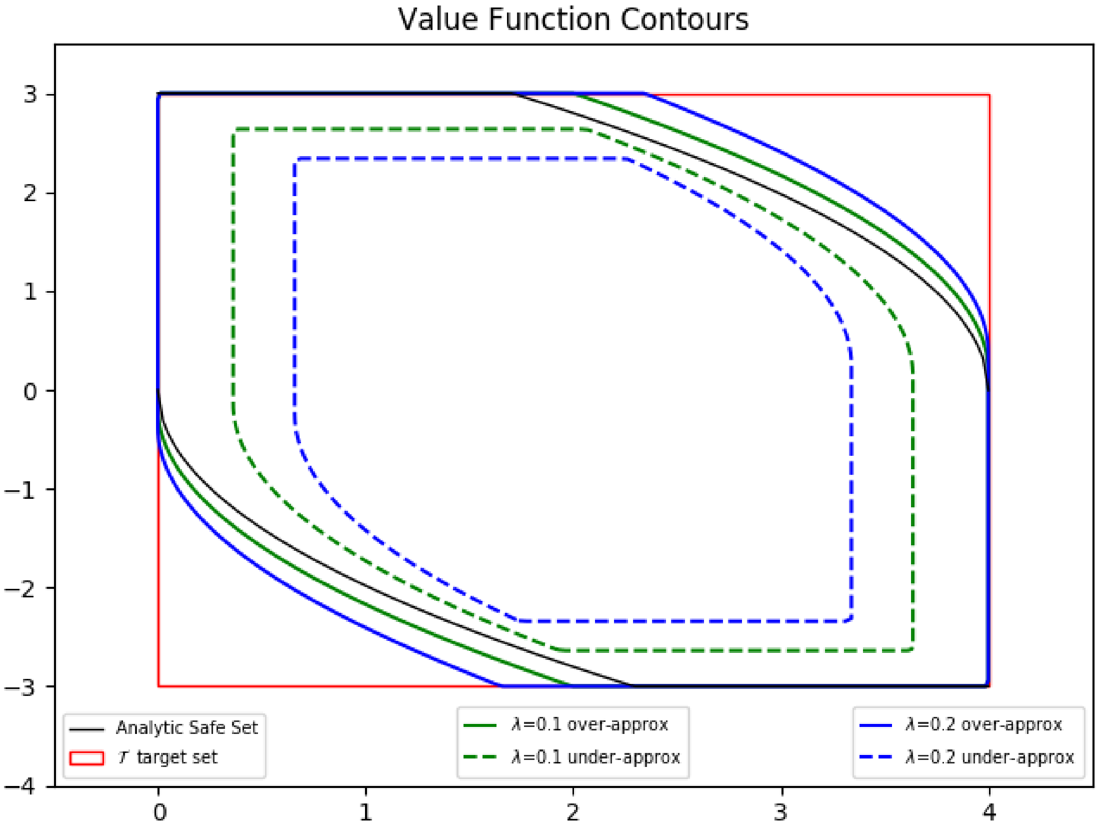


Figure 5.1: The analytic safe set and target set $\mathcal{T}$ are shown in black (interior) and red (exterior), respectively. The over and under approximated $Z$ are shown in bold green and dotted green for $\lambda = 0.1$; and bold blue and dotted blue line for $\lambda = 0.2$ (all interior).

Table 5.2: Double Integrator: Value iteration (VI) with Multigrid

| # nodes | Coarse grid | Fine grid | Fine grid with CS | Multigrid |
|---|---|---|---|---|
| $40^2$ | 0.012 | 0.025 | 0.019 | 0.031 |
| $80^2$ | 0.019 | 0.116 | 0.041 | 0.060 |
| $160^2$ | 0.153 | 1.110 | 0.084 | 0.237 |

Table 5.3: Double Integrator: Value Iteration (VI) with Warm Start (WS)

| # nodes | $M_n$ | $M_l$ | $M_l$ with WS | $M_h$ | $M_h$ with WS |
|---|---|---|---|---|---|
| $40^2$ | 0.029 | 0.044 | 0.041 | 0.022 | 0.014 |
| $80^2$ | 0.205 | .364 | 0.273 | 0.097 | 0.095 |
| $160^2$ | 1.444 | 2.727 | 2.304 | 1.257 | 1.116 |

have half the resolution per dimension of the nominal grid, e.g. if the nominal grid has $41^2$ nodes then the coarse grid has $21^2$ nodes. The results are shown in Table 5.2. We first run value iteration with the standard initialization on both the coarse and fine grid. This produces the values in columns two and three. We then run value iteration on the fine grid initialized with the coarse solution (CS), which makes up column four. Column five (multigrid) is obtained by adding columns two and four. From the table it is clear that the multigrid approach outperforms value iteration, especially as the number of nodes increases.

Lastly, denoting the current model as the nominal model $M_n$, we construct two different models: a heavy model $M_h$ with $u_{max} = 1.0$, and a light model $M_l$ with $u_{max} = 4.0$. This can be interpreted as two systems that have different control authorities due to their different masses. The $M_n$ MDR value function will be less than that of model $M_l$, but greater than that of $M_h$. We compute the value functions for $M_l$ and $M_h$ both initialized with the default initialization, and with the solution for model $M_n$, which we refer to as a *warm start* (WS). This experiment is motivated by Section 5.5.1, where we discussed the computation of reachable sets as the system model changes due to new observations from the system. Here we are not concerned with how the model estimates are obtained, but rather how to produce the reachable set for the latest model estimate as quickly as possible. Within this context $M_n$ can be viewed as an old model, and $M_l$ and $M_h$ can be thought of as two possible new models that were inferred from observations. The results for the experiment are shown in Table 5.3. In both cases ($M_l$ and $M_h$) we see that leveraging the solution for $M_n$ improves the convergence time.

### 5.6.2 Pursuit-Evasion Game

We now consider the pursuit-evasion game described in [62]. In the game player I (the control) tries to avoid being captured by player II (the disturbance) on a two

Table 5.4: Pursuit Evasion: Value iteration (VI) with Multigrid

| # nodes | Coarse grid | Fine grid | Fine grid with WS | Multigrid |
|---|---|---|---|---|
| $40^3$ | 2.068 | 29.684 | 24.320 | 26.388 |
| $80^3$ | 33.166 | 352.099 | 317.444 | 350.610 |

dimensional plane. Each player is modeled as a simple kinematic point object with planar position and heading, fixed linear velocity and controllable angular velocity. Taking player I to be at the origin the states $(x_1, x_2, x_3)$ are the relative position and heading of player II and the dynamics are

$$\begin{aligned} \dot{x_1} &= -v_u + v_d \cos x_3 + u x_2 \\ \dot{x_2} &= v_d \sin x_3 - u x_1 \\ \dot{x_3} &= d - u \end{aligned} \tag{5.44}$$

The state space is over the domain $[-6, 20] \times [-10, 10] \times [0, 2\pi[$ with $\mathcal{U} = [-u_{max}, u_{max}]$ and $\mathcal{D} = [-d_{max}, d_{max}]$.

Player I is considered captured when the relative distance (in position) between both players is less than $R > 0$, thus the target set is given by

$$\mathcal{T} = \{x | x_1^2 + x_2^2 < R^2\} \tag{5.45}$$

We first compute the value functions for the MR and MDR on a $41 \times 41 \times 41$ grid, and setting the model parameters to $v_u = v_d = 5$, $u_{max} = d_{max} = 1$, and $R = 5$. This will be referred to as the nominal model $M_n$. A visualization of the zero sub-level set for both $V(x)$ and $Z(x)$ for $\lambda = 0.001$ is shown in Fig. 5.2.

In the first experiment we compare a multigrid approach to value iteration. The results are shown in Table 5.4. The experiment and table follows the same structure used for the double integrator model. Similar to the double integrator model, the multigrid approach outperforms value iteration for the pursuit-evasion game.

We now construct two other models by tweaking $M_n$: setting $u_{max} = 1.5$, which gives the evader an advantage, we get model $M_e$, and setting $d_{max} = 1.5$, which gives the pursuer an advantage, we get model $M_p$. In the final experiment we look at the impact of initializing value iteration with a solution from a similar model. Just like in the previous benchmark example, this experiment is motivated by Section 5.5.1, where now $M_e$ and $M_p$ represent two possible models inferred from the system observations. In this context we have just “learned” that the evader/pursuer is more maneuverable ($M_e/M_p$). We compute both value functions with and without setting the initialization to the solution for $M_n$. Again, we refer to this initialization as a warm start. The results are shown in Table 5.5.

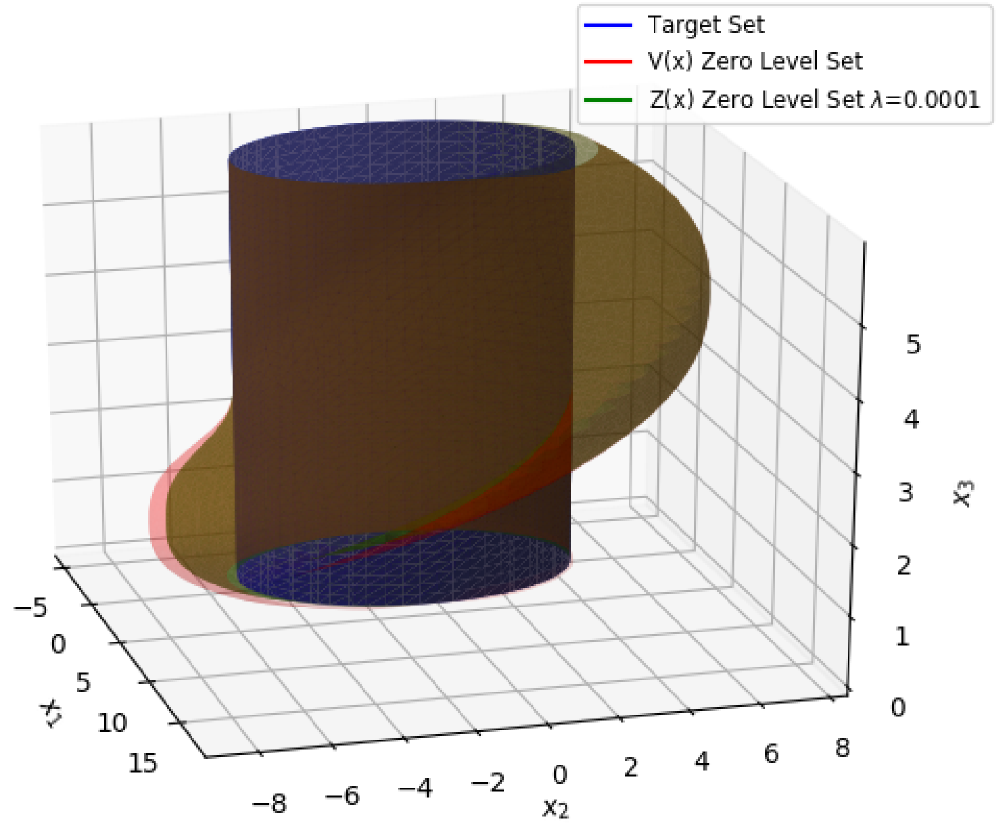
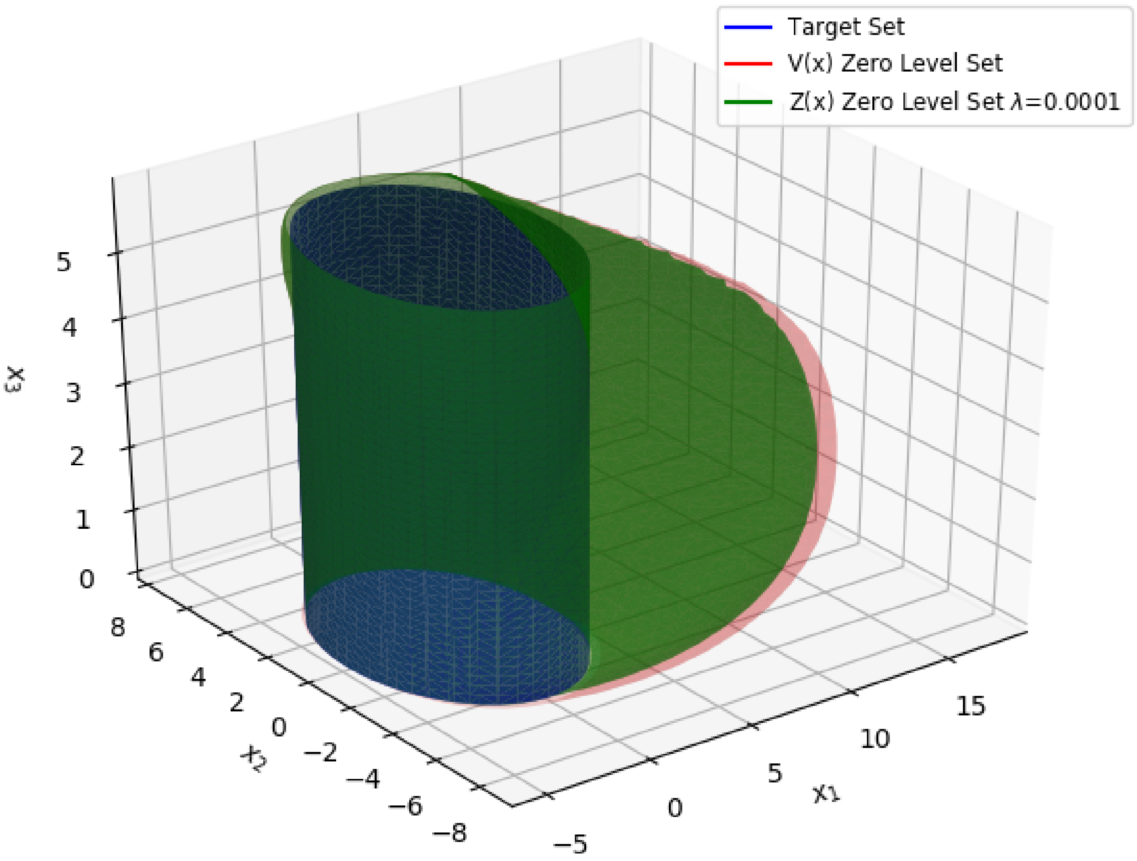


Table 5.5: Pursuit Evasion: Value iteration (VI) with Warm Start (WS)

| # nodes | $M_n$ | $M_e$ | $M_e$ with WS | $M_p$ | $M_p$ with WS |
|---|---|---|---|---|---|
| $40^3$ | 35.043 | 32.242 | 21.483 | 26.319 | 23.366 |
| $80^3$ | 439.751 | 416.965 | 308.821 | 300.847 | 296.568 |


Target Set
V(x) Zero Level Set
Z(x) Zero Level Set $\lambda$=0.0001
$x_3$
0
1
2
3
4
5
$x_2$
8
6
4
2
0
−2
−4
−6
−8
$x_1$
−5
0
5
10
15

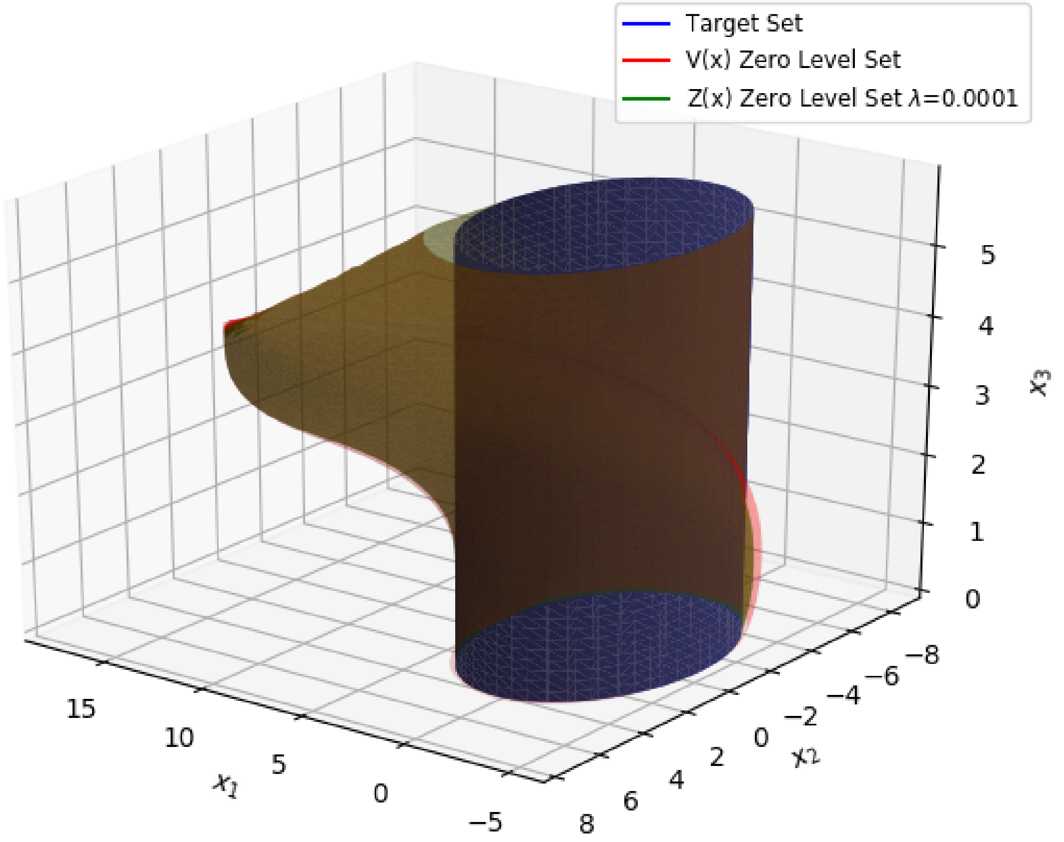


Figure 5.2: The target set $\mathcal{T}$ (blue cylinder), zero sub-level sets of $V$ (red) and $Z$ (green) shown from three different perspectives. The discount rate for $Z$ is $\lambda = 0.01$. Note that the zero sub-level set of $Z$ is a subset of the zero sub-level set of $V$.

# Chapter 6

# Conclusion

We gave an introduction to HJ reachability analysis, and showed how it can be used as a robust analysis tool for ensuring safety in robotic systems with uncertainty. In particular we focused on the application of safe learning. In the safe learning problem our system attempts to learn a policy to optimize its performance for a non-safety objective, while simultaneously satisfying a constraint (safety objective).

The reachability-based safe learning approach was exemplified on a quadrotor learning to track an altitude trajectory without crashing into the floor or ceiling. In this example the quadrotor ultimately learned a good tracking controller, and the safety controller from reachability analysis prevented the quadrotor from crashing.

In the safe learning example we saw the trade-off between conservativeness and performance. The first place we encounter this is in the cart-pole system example, where we tried to learn a swing up controller, while keeping the position confined to a certain interval. The learning control constantly pushes the cart-pole towards the boundary of the safe set, which leads to chattering between the safe control and learning control, ultimately resulting in the task not being accomplished. We showed that this issue could be addressed by incorporating the value function for safety into the objective of the learning algorithm.

We briefly talked about model-validation as a means for additional safety. The basic idea was to compare the observations from the system against the model used to generate the safe set. In the event that these observations deviated significantly from the model, the safe set would be contracted, only allowing the system to operate in regions for which the model was accurate.

In the next part of the thesis we looked at data-driven approaches to computing the safe set. As stated earlier the robust safety analysis can potentially yield behavior that is too conservative and hinders performance. By using observations modified the safe set, so that it produces a level of conservativeness consistent with the uncertainty that we have about the system. We discussed two approaches of leveraging data for this purpose: model-based, and model-free.

As an example of a model-based method we consider systems with additive uncertainty, and modeled that uncertainty as a GP. The confidence intervals of the GP provide a convenient way to bound the uncertainty. As more data was collected we refit the GP, determined the uncertainty bounds, and updated the safe set globally via reachability analysis. As a consequence of the GP, we also developed a probabilistic model validation algorithm.

Following the model-based example, we demonstrated a model-free approach based on temporal difference learning. A key advantage of TD is that it only makes local updates to the value function, and thus is better suited for on-line usage. Another related benefit, is that it only focuses on updating the safe set in regions that the system visits, so the safe set is modified as to aid in the system's non-safety objectives. This approach was demonstrated on the double integrator and *Catch Me if You Can* examples.

Finally, we finished by introducing a novel discounted formulation for computing reachable sets. The main benefit of this formulation is that the value iteration algorithm used to solve the problem uses a contraction mapping as the backup operator thus the algorithm is agnostic to its initialization. This ultimately leads to improved solution methods for computing reachable sets, e.g. policy iteration and multigrid approaches. It also enhances the model-based approach for learning-based reachability by making it possible to use warm starts. These benefits were shown on the double integrator and pursuit-evasion examples. We also hinted at how this formulation could be combined with model-free techniques developed in RL.

In conclusion, we have developed a new formulation for computing reachable sets that more easily allows us to modify the reachable sets based on observations from the environment. Adapting the reachable set is critical to retaining strong performance while ensuring safety for robotic systems with uncertainty.

# Bibliography

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, “An application of reinforcement learning to aerobatic helicopter flight,” *Advances in Neural Information Processing Systems*, 2007.

[2] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, “Reachability-based safe learning with gaussian processes,” *Proceedings of IEEE Conference on Decision and Control*, 2014.

[3] A. K. Akametalu, S. Ghosh, J. F. Fisac, and C. J. Tomlin, “A minimum discounted reward hamilton-jacobi formulation for computing reachable sets,” *Submitted to IEEE Transactions on Automatic Control*, 2018.

[4] A. K. Akametalu and C. J. Tomlin, “Temporal-difference learning for online reachability analysis,” *Proceedings of IEEE European Control Conference*, 2015.

[5] A. Alla, M. Falcone, and D. Kalise, “An efficient policy iteration algorithm for dynamic programming equations,” *SIAM Journal on Scientific Computing*, 2015.

[6] Amazon.com, Inc. (2016). Amazon prime air, [Online]. Available: `http://www.amazon.com/b?node=8037720011` (visited on 02/16/2016).

[7] A. Ames, J Grizzle, and P. Tabuada, “Control barrier function based quadratic programs with application to adaptive cruise control,” *Proceedings of Conference on Decision and Control*, 2014.

[8] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, “Provably safe and robust learning-based model predictive control,” *Automatica*, 2013.

[9] J.-P. Aubin, A. Bayen, and P. Saint-Pierre, *Viability Theory: New Directions*. Springer, 2011.

[10] AUVSI News. (2016). Uas aid in south carolina tornado investigation, [Online]. Available: `http://www.auvsi.org/blogs/auvsi-news/2016/01/29/tornado` (visited on 02/16/2016).

[11] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.

[12] M. Bardi, M. Falcone, and P. Soravia, "Numerical methods for pursuit-evasion games via viscosity solutions," *Stochastic and differential games*, Springer, 1999.

[13] E. Barron, "Differential games with maximum cost," *Nonlinear analysis: Theory, methods & applications*, 1990.

[14] E. Barron and H. Ishii, "The bellman equation for minimizing the maximum cost," *Nonlinear Analysis: Theory, Methods & Applications*, 1989.

[15] BBC Technology. (2016). Google plans drone delivery service for 2017, [Online]. Available: `http://www.bbc.com/news/technology-34704868` (visited on 02/16/2016).

[16] R. Bellman, "Dynamic programming," 1957.

[17] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," *Proceedings of Conference on Decision and Control*, 2016.

[18] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995.

[19] K. Bimbraw, "Autonomous cars: past , present and future," *Thapar University*, 2010.

[20] O. Bokanowski, S. Maroso, and H. Zidani, "Some convergence results for howard's algorithm," *SIAM Journal on Numerical Analysis*, 2009.

[21] M. Chen, "High dimensional reachability analysis: Addressing the curse of dimensionality in formal verification," PhD thesis, EECS Department, University of California, Berkeley, 2017.

[22] M. Chen, S. Herbert, and C. J. Tomlin, "Fast reachable set approximations via state decoupling disturbances," *Proceedings of Conference on Decision and Control*, 2016.

[23] ——, "Exact and efficient hamilton-jacobi-based guaranteed safety analysis via system decomposition," *Proceedings of International Conference on Robotics and Automation*, 2017.

[24] M. Chen, Q. Hu, C. Mackin, J. F. Fisac, and C. J. Tomlin, "Safe platooning of unmanned aerial vehicles via reachability," *Proceedings of IEEE Conference on Decision and Control*, 2015.

[25] C Chow and J. N. Tsitsiklis, "An optimal one-way multigrid algorithm for discrete-time stochastic control," *IEEE Transactions on Automatic Control*, 1991.

[26] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *ArXiv preprint*, 2016.

[27] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," *International Conference on Machine Learning (ICML)*, 2008.

[28] E. A. Coddington and N. Levinson, *Theory of ordinary differential equations.* Tata McGraw-Hill, 1955.

[29] J. Ding, E. Li, H. Huang, and C. J. Tomlin, "Reachability-based synthesis of feedback policies for motion planning under bounded disturbances," *Proceedings of International Conference on Robotics and Automation*, 2011.

[30] J. Ding, C. J. Tomlin, L. R. Hook, and J. Fuller, "Initial designs for an automatic forced landing system for safer inclusion of small unmanned air vehicles into the national airspace," *Proceedings of the 2016 IEEE/AIAA 35th Digital Avionics Systems Conference*, IEEE, 2016.

[31] B. Djeridane and J. Lygeros, "Neural approximation of pde solutions: An application to reachability computations," *Proceedings of Conference on Decision and Control*, IEEE, 2006.

[32] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Adaptive control and the nasa x-15-3 flight revisited," *IEEE Control Systems*, 2010.

[33] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of hamilton-jacobi-isaacs equations," *Indiana University Mathematics Journal*, 1984.

[34] M. Falcone and R. Ferretti, "Discrete time high-order schemes for viscosity solutions of hamilton-jacobi-bellman equations," *Numerische Mathematik*, 1994.

[35] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *ArXiv preprint arXiv:1705.01292*, 2017.

[36] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," *Proceedings of the International Conference on Hybrid systems: Computation and Control*, ACM, 2015.

[37] D. Fitzgerald, R. Walker, and D. Campbell, "A vision based forced landing site selection system for an autonomous uav," *Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference*, IEEE, 2005.

[38] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," *International Conference on Computer Aided Verification*, Springer, 2011.

[39] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, 2015.

[40] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, 2005.

[41] A Genz, "Numerical computation of multivariate normal probabilities," *Journal of computational and graphical statistics*, 1992.

[42] J. H. Gillula and C. J. Tomlin, "Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor," *International Conference on Robotics and Automation*, 2012.

[43] J. H. Gillula and C. J. Tomlin, "Reducing conservativeness in safety guarantees by learning disturbances online: iterated guaranteed safe online learning.," *Robotics: Science and Systems*, 2012.

[44] M. R. Greenstreet and I. Mitchell, "Integrating projections," T. A. Henzinger and S. Sastry, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1998.

[45] A. Hobbs, "Unmanned aircraft systems," *Human factors in aviation*, E. Salas and D. Maurino, Eds., 2nd ed., Elsevier, 2010.

[46] G. M. Hoffmann and C. J. Tomlin, "Decentralized cooperative collision avoidance for acceleration constrained vehicles," *Proceedings of IEEE Conference on Decision and Control*, 2008.

[47] R. A. Howard, *Dynamic programming and Markov processes.* Wiley for The Massachusetts Institute of Technology, 1964.

[48] H. Huang, J. Ding, W. Zhang, and C. J. Tomlin, "A differential game approach to planning in adversarial scenarios: a case study on capture-the-flag," *International Conference on Robotics and Automation*, 2011.

[49] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *ArXiv preprint*, 2017.

[50] D. R. Jenkins, "Hypersonics before the shuttle: A concise history of the x-15 research airplane," 2000.

[51] S. Kaynama and M. Oishi, "Schur-based decomposition for reachability analysis of linear time-invariant systems," *Proceedings of Conference on Decision and Control*, 2009.

[52] ——, "Complexity reduction through a schur-based decomposition for reachability analysis of linear time-invariant systems," *International Journal of Control*, 2011.

[53] ——, "A modified riccati transformation for decentralized computation of the viability kernel under lti dynamics," *IEEE Transactions on Automatic Control*, 2013.

[54] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Machine learning*, 2002.

[55] J. Z. Kolter, C. Plagemann, D. T. Jackson, A. Y. Ng, and S. Thrun, "A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving," *Proceedings of International Conference on Robotics and Automation*, 2010.

[56] J. Z. Kolter and A. Y. Ng, "Policy search via the signed derivative.," *Proceedings of Robotics: Science and Systems Conference*, 2009.

[57] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, 2000.

[58] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis: Internal approximation," *Systems & Control Letters*, 2000.

[59] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," *Proceedings of International Conference on Robotics and Automation*, 2010.

[60] K. Margellos and J. Lygeros, "Hamilton-jacobi formulation for reach-avoid differential games," *IEEE Transactions on Automatic Control*, 2011.

[61] I. M. Mitchell, "Scalable calculation of reach sets and tubes for nonlinear systems with terminal integrators: A mixed implicit explicit formulation," 2011.

[62] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, 2005.

[63] I. M. Mitchell and J. A. Templeton, "A toolbox of hamilton-jacobi solvers for analysis of nondeterministic continuous and hybrid systems," *Proceedings of the 8th International Workshop on Hybrid Systems: Computation and Control*, Berlin, Heidelberg: Springer, 2005.

[64] I. M. Mitchell and C. J. Tomlin, "Overapproximating reachable sets by hamilton-jacobi projections," *Journal of Scientific Computing*, 2003.

[65] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *Proceedings of International Conference on Machine Learning*, 2016.

[66] V. Mnih, K. Kavukcuoglu, D. Silver, A. a. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[67] T. M. Moldovan and P Abbeel, "Safe exploration in markov decision processes," *Proceedings of International Conference on Machine Learning*, 2012.

[68] National Aeronautics and Space Administration. (2016). Challenge is on to design sky for all, [Online]. Available: `http://www.nasa.gov/feature/challenge-is-on-to-design-sky-for-all` (visited on 02/12/2016).

[69] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces.* Springer Science & Business Media, 2003.

[70] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *The Journal of Machine Learning Research*, 2003.

[71] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," *Proceedings of International Conference on Hybrid Systems: Computation and Control*, 2004.

[72] M. L. Puterman and S. L. Brumelle, "On the convergence of policy iteration in stationary dynamic programming," *Mathematics of Operations Research*, 1979.

[73] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.* MIT Press, 2006.

[74] J. W. Roberts, I. R. Manchester, and R. Tedrake, "Feedback controller parameterizations for reinforcement learning," *Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2011.

[75] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering, 1994.

[76] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: A modern approach*. Prentice hall Upper Saddle River, 2003.

[77] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," *Proceedings of Robotics: Science and Systems Conference*, 2016.

[78] J. Schulman, S. Levine, M. Jordan, and P. Abbeel, "Trust region policy optimization," *International Conference on Machine Learning (ICML)*, 2015.

[79] J. Sethian, "A fast marching level set method for monotonically advancing fronts.," *Proceedings of the National Academy of Sciences of the United States of America*, 1996.

[80] C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, 1988.

[81] C. Sloth, G. J. Pappas, and R. Wisniewski, "Compositional safety analysis using barrier certificates," *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, 2012.

[82] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[83] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, 1988.

[84] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, 2000.

[85] G. Tesauro, "Temporal difference learning and td-gammon," *Communications of the ACM*, 1995.

[86] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, 1997.

[87] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[88] J. Van Den Berg, S. Miller, K. Goldberg, and P. Abbeel, "Gravity-based robotic cloth folding," *Algorithmic Foundations of Robotics IX*, Springer, 2010.

[89] P. C. Wang, S. Miller, M. Fritz, T. Darrell, and P. Abbeel, "Perception for the manipulation of socks," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011*, IEEE, 2011.

[90] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, 1992.