APPLICATION OF LEVEL SET METHODS TO CONTROL AND REACHABILITY PROBLEMS IN CONTINUOUS AND HYBRID SYSTEMS

A DISSERTATION
SUBMITTED TO THE PROGRAM IN
SCIENTIFIC COMPUTING AND COMPUTATIONAL MATHEMATICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ian Michael Mitchell
August 2002

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Claire Tomlin
(Aeronautics and Astronautics)
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ronald Fedkiw
(Computer Science)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen Boyd
(Electrical Engineering)

Approved for the University Committee on Graduate Studies:

Abstract

Computer control is an intimate part of many modern devices, and yet there exist few formal methods for designing and analyzing the complex interactions between the discrete realm of digital computers and the continuous dynamics that hold sway outside of it. Our goal in the research described here is verifying the safety of such hybrid systems: systems which involve both discrete and continuous behaviors. We have developed computational methods that determine in what states a system might find itself, and how a system can use its control authority to avoid reaching states that are known to be unsafe.

Our main tool is the backwards reachable set. Given some system dynamics and a set of target states, the backwards reachable set is the set of states which give rise to trajectories leading to the target set. If the target set is chosen as the known dangerous states of a system, the reachable set encompasses those states which lead to danger. Consequently, reachable sets can be used for safety verification and for synthesis of safe controls.

We describe, prove the correctness of, and implement an algorithm—based on a time dependent Hamilton-Jacobi-Isaacs partial differential equation—for computing the backwards reachable set of a continuous dynamic game. Unlike its alternatives, the method presented here can draw directly upon the accurate numerical schemes developed in the level set literature to solve such Hamilton-Jacobi equations. The differential game formulation allows us to handle not only traditional control inputs, but also conservatively treat noise, model uncertainty and the unknown actions of other agents as adversarial disturbance inputs.

While our technique can be applied to systems with nonlinear dynamics in any number of state space dimensions, its computational cost scales poorly with dimension. We therefore describe a method for overapproximating reachable sets using quickly computed lower

dimensional projections. For safety verification applications, overapproximations are sufficient.

Finally, we show how an existing algorithm for computing reachable sets of hybrid systems can be accurately implemented by incorporating our method for finding continuous reachable sets. The continuous, projective and hybrid reachable set algorithms are all demonstrated on a number of examples.

Preface

I'm moving in seventy two hours, and very soon I've got to decide whether I should eat or pack the box of Millenios* that I purchased back when we thought the lights would go out in January 2000 (we were only off by sixteen months). That, combined with a goodly dose of procrastination, has got me thinking what I would have done if I'd been born one thousand years ago. Since "perpetual student" hadn't yet been invented, I'm guessing soothsayer would have been a pretty good gig for me.

I'm always happiest when I know what is going to happen in advance.† I count myself lucky, therefore, to have been born in a time when I can prophesize with the tools of mathematics and computers rather than a deck of Tarot cards. I'm not alone in my fascination with prediction—the fields of mathematics and computer science were both born from the quest for accurate and consistent foresight—and I hope that this research is just the beginning of my contribution to this endeavor.

Simulation was the original killer app, and its widespread availability has revolutionized the process of design. In today's world, a major part of creating any new technological product is extensive testing through virtual models. One shortcoming of traditional simulation methods, however, is that they examine only one possible evolution of a system's state at a time. At the heart of my investigations is an alternative method of simulation that computes all possible evolutions at once, so that we can be more confident that we understand all of the behaviors that a system might exhibit, and can design away any that might not be safe.

*For those of you not familiar with them, Millenios™ were "once in a lifetime", "limited edition", "official cereal of the millennium" Cheerios® with not just the usual O's, but also extra special 2's. And I've had a box of them in my cupboard for almost two years. No kidding.

†Despite rumors to the contrary, I would not get into the scrying business because of its many opportunities for evil. That's not to say, however, that I wouldn't enjoy them as a perk.

The research described in this thesis constitutes just a few steps along the paths of simulation and verification, steps that are far from the first or the last. This thesis is not a tutorial on control, verification, reachability, hybrid systems, or level set methods. On the other hand, an undergraduate level background in numerical differential equations should be sufficient to understand the contents, and I hope that I have written clearly enough that the interested reader can pick up necessary portions of the aforementioned research fields on the fly.

It's been thirteen years since I started my university career, and I've had a lot of help getting to where I am today. Kaaren, my new wife but old flame, has been beside me through almost all of it; the last three years here at Stanford with her have been the happiest of my life. As she finally prepares to embark on law school, I can just hope that I haven't put any ideas in her head about spending the rest of her life there.

From one love of my life to the others. Science, mathematics, learning, teaching and writing are the reasons that I chose graduate school, and my parents are the reason that I enjoy them so much. It was my father who first introduced me to computers and simulation many years ago[‡] and thereby set the path which I have followed since. I have been carried far by my skills at writing, and they are the result of my mother's zealous editing ability; I look back fondly now at so many grade school essay drafts that were more red ink than black. If I had to pick one incident that lead me to where I am today, it would be my first high school science project. I raised some mold under a variety of temperature and nutrient conditions, measured its growth daily, and wrote a little computer simulation that interpolated growth patterns (in ASCII graphics!) for other conditions. I won a regional award for scientific method, all thanks to my mother and father, for it was their idea, explanations, and patience that allowed me to complete the project.

I would also like to thank the rest of my (newly enlarged) family: Eric & Johanna, Jean, Leanor & Henry, Margie, Adi and Una. I don't see you as often as I would like, but I love you all just the same.

I have had a fantastic time in my five years at Stanford, and that is because of the people with whom I've worked. Professor Claire Tomlin has been a wonderful supervisor and colleague. Her thesis research on hybrid systems is one of the two pillars upon which mine

[‡]When he started in the field, he actually did predict the future with decks of (punched) cards. I count myself doubly lucky to have lived in the age of Matlab.

is built, and the timing of her arrival at Stanford could not have been better. My approach to research has a tendency to be overly cautious, but her enthusiasm, optimism and fresh ideas have kept me moving forward these past three years (and flying all over the place for the last half of that).

Professor Ronald Fedkiw has taught me level set methods, the other pillar upon which I have built my research. He has spent portions of the last two years writing a book on these methods [109]. This process has afforded me two great benefits: first, he has been thinking hard about how best to explain the state of the art, and second, when writing he regularly procrastinates by coming into my office to chat, thus giving me the opportunity to have my many questions answered.

Much of the work in this thesis is collaborative in nature, and I would like to thank my coauthors profusely for their help. It has been a great pleasure to work closely with Alexandre Bayen on several papers, especially the ones that involved trips to Europe. Meeko Oishi introduced me to the pleasure of being a coauthor without any writing duties, a position I fear I will not often get to hold. And then there is Doug Enright. Cheer up Doug! You're first author on a JCP paper and I, for one, think your research is much cooler than mine.

Professor Stephen Boyd has played a major role in shaping my understanding of control theory and optimization through his incredible teaching and by serving on my reading committee. With help from him and from Haitham Hindi I hope to further investigate the relative pros and cons of reachable set determination using LMIs and SDPs for the types of nonlinear systems studied in this thesis. I would also like to thank Professors David Dill and Antony Jameson for making my oral defense such a painless affair.

I'd like to thank Claire's lab—Alex, Meeko, Ronojoy Ghosh, Inseok Hwang, Gokhan Inalhan, Jung Soon Jang, Rodney Teo and Dusan Stipanovic—and Ron's lab—Doug, Robert Bridson, Sergey Koltakov, Neil Molino, Igor Neverov, Joseph Teran, Eran Guendelman, Frederic Gibou and Duc Nguyen—for putting up with my often boisterous presence at group meetings. My studies at Stanford would not have been possible without Professors Gene Golub, Andrew Stuart, Walter Murray, Jim Varah and Mark Greenstreet, who each helped in their own way to get me into and through the Scientific Computing and Computational Mathematics program. I have also enjoyed working alongside the students in SCCM, including Eric Boman, Kris Buschelman, Maureen Doyle, Chen Grief, Hallgeir Melboe, Nhat Nguyen, Will Smit, Paul Tupper, and many others. Evelyn Boughton, Sherann

Ellsworth, Arden King and Dana Parga have helped me jump through all of the hoops that a doctoral degree entails.

On the technical side, hearty acknowledgment is due to Professors Stanley Osher and Hong-Kai Zhao for discussions about numerical schemes for solving Hamilton-Jacobi equations, Professors John Lygeros, L. C. Evans, Shankar Sastry and Alexander Kurzhanski for discussions about the previous and current time dependent Hamilton-Jacobi formulations, and Professors Patrick Saint-Pierre and Jean-Pierre Aubin for discussions about the connections with viability theory. The original idea for tackling continuous reachability in higher dimensional systems using projections came out of work with Professor Mark Greenstreet.

The research described here has been supported by the Defense Advanced Research Projects Agency under the Software Enabled Control program (AFRL contract F33615-99-C-3014). While at Stanford I have also been supported by a School of Engineering fellowship from the Groswith family and by Texas Instruments as part of the Digital Signal Processor University Research Fund.

Finally, I would like to thank my many friends for keeping me from disappearing completely into my studies. Without help from Fraser & Katrina (and little Chloe), Zeke, Stephanie, Kevin & Allison (and little Amelie), Joel, Brucek, Greg, Eric & Sara, the many Chris, Susannah & Jim, Nicky, Holly, Prita, Ujval, Igor, Florian and many others, this time at Stanford would have been a long five years, instead of just a wonderful flash gone by.

Computational science may be the preferred method of divination in this age, but for better or worse it works only for our machines and leaves us just as much in the dark as ever about our own future. What I can say, however, is that I've had a wonderful time here on the farm, I've married the beautiful princess, and I'm ready to ride off into the sunrise.[§]

Ian M. Mitchell
Stanford University
August 2002

[§]The sunset is no good, since there just aren't too many employment opportunities west of Stanford.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mathematical models are widely used to design and analyze all types of systems in advance of their construction. The core of the research described in this thesis is a computational tool that can be used to determine whether such systems—for example, airplane autopilots—may act in unexpected ways which their users would consider incorrect or unsafe.

1.1 What's So Interesting?

Many modern devices are composed of large numbers of subsystems, and while the behavior of each subsystem is well understood in isolation, their interaction may lead the whole system into states that are undesirable or unsafe. Consequently, verification and validation have received major attention in many fields of engineering. The simplest form of computational validation is simulation. Provided with a reasonable mathematical model, simulation can expose the flaws of a system much more quickly and cheaply than would construction of a physical prototype. Simulation has proven particularly useful during the iterative process of design, where engineers seek to rapidly modify and evaluate new features and functions.

The major drawback of simulation is that it only checks a single trajectory of the system at a time. For systems with many different state values and/or many input signals, it would be prohibitively expensive to check the safety of every possible system trajectory by simulation

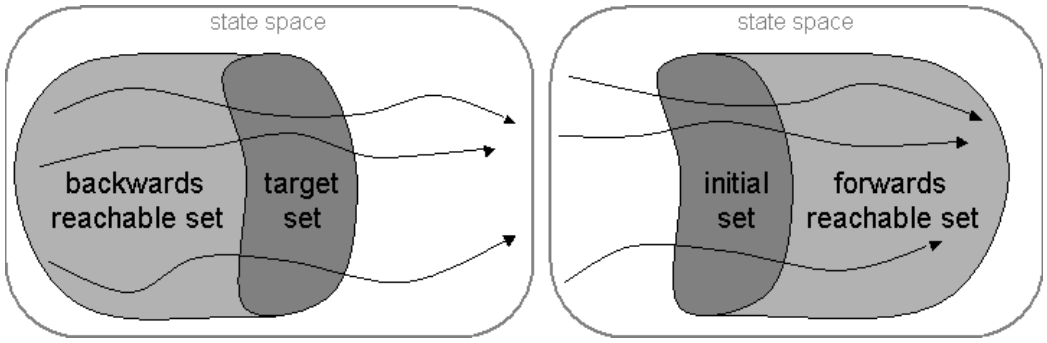

Figure 1.1: Difference between backwards and forwards reachable sets.

alone. The concern is that simulation of a few thousand, or even a few million, individual trajectories might miss some isolated but important unsafe case.

One avenue that researchers have followed in their quest to catch every potential failure mode is the computation of reachable sets. Reachable sets are a way of capturing the behavior of entire groups of trajectories at once. There are two basic types of reachable sets, depending on whether an initial or a final condition is specified. For a *forward reachable set*, we specify the initial conditions and seek to determine the set of all states that can be reached along trajectories that start in that set. Conversely, for a *backward reachable set* we specify a final or target set of states, and seek to determine the set of states from which trajectories start that can reach that target set. The difference is illustrated in figure 1.1, in which the arrows represent trajectories of the system.

Computing the set of states from which trajectories of a continuous dynamic game can reach a given target set has its roots in the work of Isaacs [75], who used his calculations to derive capture regions for evaders in pursuit-evasion games, games that were motivated by the study of military jets and surface to air missiles. Our major application for reachable sets will be to verify the correct behavior of systems. Figure 1.2 demonstrates how the backward reachable set can be used to verify a system's safety. To start with, we collect all states that are known a priori to be unsafe into the target set. The target set's backward reachable set is the set of states which give rise to unsafe trajectories. Therefore, if the initialization conditions for the system overlap with the backwards reachable set, the system may be unsafe and should be modified.

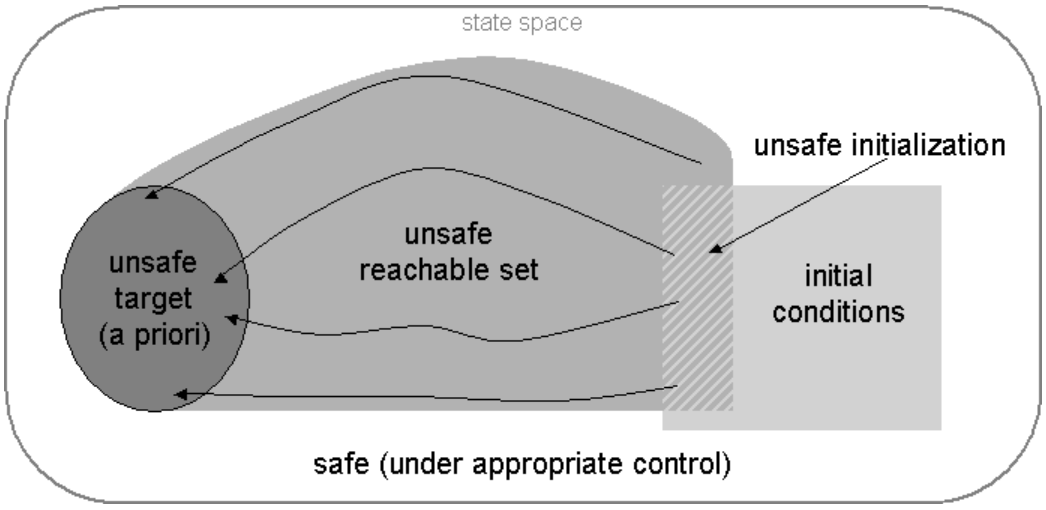

Figure 1.2: Using the backward reachable set to verify safety.

As an example, many of the systems we study involve collision avoidance protocols for aircraft. The target set in our analyses is the set of states considered a collision; for example, at cruising altitude the Federal Aviation Administration (FAA) mandates that commercial aircraft maintain a five mile horizontal separation. The backwards reachable set includes those states which will lead to a collision. In this case, positions many miles in front of the aircraft will be part of the backwards reachable set—far enough in front that the pilot could respond to a danger by changing heading or altitude—but not positions behind the aircraft, since it will always be moving forward. If another aircraft enters this reachable set, there is cause for alarm.

The search for methods of computing the reachable sets of purely discrete systems, such as those modeled by finite automata, has met with considerable success and has led to the development of powerful tools for automatic verification; for example, the binary decision diagram [35]. Most engineering systems, however, are not purely discrete. Continuous dynamics are the norm in control engineering problems, and in many modern systems important behaviors arise from the interaction between discrete and continuous components. While we have ways of simulating these systems, verification and validation of their safety demands a more rigorous approach.

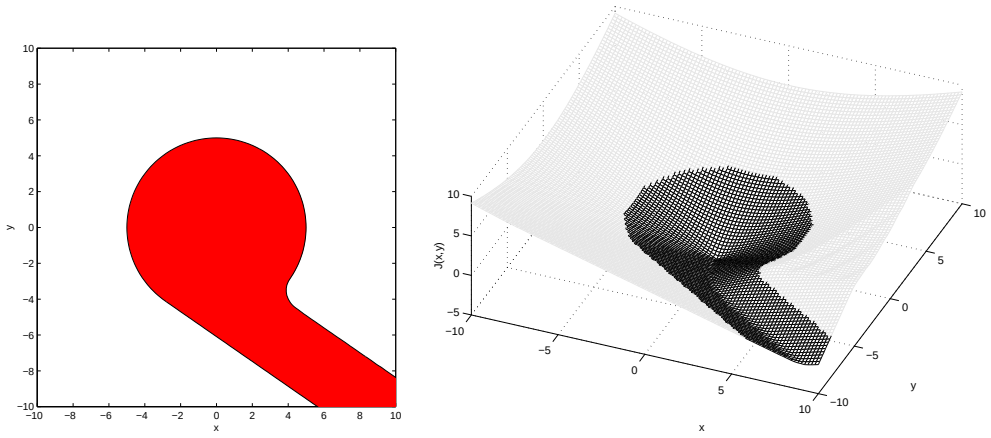

Figure 1.3: An example two dimensional set $\mathcal{G}$ (left), and one possible implicit surface representation $J(x, y)$ (right, darker portion of mesh is $J(x, y) \leq 0$).

1.2 Scope and Goals

The focus of this thesis is on computational methods for determining the reachable sets of continuous and hybrid systems. A *hybrid system* is one in which the interaction between discrete and continuous components plays an important role in determining the evolution of the system's state. As described above, reachable sets can be used for safety verification. In addition, we demonstrate how they can be used to synthesize controllers which are guaranteed to act safely.

The primary challenges faced by attempts to compute these reachable sets involve the uncountable number of distinct states in a continuous system. In a finite discrete system, we can describe the reachable sets by enumerating their member states and we can evolve them by following individual trajectories. This strategy cannot be applied in a continuous setting, and we must look for other ways of answering two questions: how do we represent reachable sets, and how do we evolve them according to the system's dynamics?

One popular way of describing sets of continuous states is called the implicit surface function representation. Consider a closed set $\mathcal{G} \subseteq \mathbb{R}^n$. An implicit surface representation of $\mathcal{G}$ would define a function $\phi : \mathbb{R}^n \to \mathbb{R}$ such that $\phi(x) \leq 0$ if $x \in \mathcal{G}$ and $\phi(x) > 0$ if $x \notin \mathcal{G}$. Implicit

surface representations are not unique. An example of a two dimensional set $\mathcal{G}$ is shown in figure 1.3, along with one possible implicit surface representation.

Level set methods [110, 109] are a collection of numerical techniques for evolving implicit surfaces according to underlying flow fields. Their governing equations are Hamilton-Jacobi (HJ) partial differential equations (PDEs), a class of nonlinear, first order hyperbolic PDEs whose solutions can exhibit shock and rarefaction events. Despite the challenges of working with such a PDE, level set methods can in many cases accurately compute the evolution of implicit surface functions under complex, nonlinear flow fields.

The novel theoretical contribution of this research is the formulation of and proof that the viscosity solution of a Hamilton-Jacobi-Isaacs (HJI) PDE describes the continuous backwards reachable set. The HJI PDE allows for system models that contain two adversarially opposed input signal vectors. One input will try to keep the systems away from the target set, while the other will try to drive it into the target set. Since the target set is an unsafe set in our examples, these two inputs can be given the traditional names of control and disturbance. The latter class of inputs is particularly useful as a way of conservatively treating not just the actions of an adversary, but also model uncertainties and noise.

Unlike any other formulation of the reachable set, our HJI PDE can be solved with the very accurate numerical methods drawn from the level set literature. The bulk of this thesis describes practical aspects of implementing level set algorithms to calculate reachable sets for various continuous and hybrid systems. Extensive examples demonstrate the accuracy with which we can determine the reachable sets for systems with complex nonlinear dynamics and multiple inputs. We also compare our method with competing algorithms for computing forwards and backwards reachable sets.

The eventual goal of this research is a tool that could automatically analyze the safety of general nonlinear continuous and hybrid systems with up to perhaps five continuous state space dimensions. The level set function basis on which we build is conceptually well suited to automatic analysis because it can handle such general systems, but it scales poorly with dimension. One chapter is devoted to a projection based technique that may allow efficient approximation of reachable sets for higher dimensional systems.

1.3 Outline

This thesis presents the theory and implementation of algorithms for computing the backwards reachable sets of continuous and hybrid systems.

- Chapter 2 contains the theory and implementation details behind our algorithm for continuous systems, which we will call our *continuous reachable set algorithm.* It formulates the backwards reachable set in terms of the solution of an HJI PDE, and proves that the viscosity solution of this PDE is an implicit surface representation of the backwards reachable set. After comparing the properties of this formulation with those of its competitors, we describe how to implement a level set algorithm for computing its viscosity solution.

- Chapter 3 demonstrates our continuous reachable set implementation on several examples. The main example, the "game of two identical vehicles" is a classical pursuit-evasion game. The chapter concludes with an example of how the reachable set can be used to study the autolanding procedure in a modern commercial airplane.

- Chapter 4 discusses a technique that may allow us to skirt Bellman's "curse of dimensionality." The cost of most methods for solving the HJI PDE from chapter 2 grows exponentially with the continuous dimension of the system. This chapter examines a technique for overapproximating the true reachable set using a series of overlapping projections into lower dimensional subspaces; these projections would be much less expensive to compute.

- Chapter 5 describes hybrid systems, and how the algorithm for computing continuous reachable sets can be adapted to work on a general hybrid system.

- Chapter 6 demonstrates the computation of hybrid system reachable sets on several collision avoidance protocol examples and a multimodal aircraft autolanding system.

Because the methods in the later chapters draw upon those in the previous ones, the reader will notice that the first chapters treat their subjects with more rigor. Our early focus in this research was the implementation and demonstration of these methods, and we are now

returning to formalize our results. While we have completed work on the continuous case, formal treatment of the hybrid case is still in progress.

This thesis is a compilation of results and examples from many papers with many coauthors [102, 100, 137, 98, 99, 105, 103, 23]. The contribution of this author lies in the theory, algorithms and implementations that compute the reachable sets. Where possible, the presentation in these pages has been restricted to this author's contribution; however, clarity sometimes demands additional details be included. In particular, the models in all of the examples were developed either by coauthors (sections 3.3 and 6.2) or drawn from the literature (the remaining examples). The algorithm for hybrid reachable sets presented in section 5.2 is drawn from [138]; this author's contribution lies in the implementation of the reach-avoid operator given in section 5.3. Responsibility for the remaining results and text lies, for better or worse, with this author.

Chapter 2

Reachable Sets for Continuous Systems

This chapter details our method for computing the reachable sets of purely continuous systems. The first section focuses on the theory, the second on practical implementation, and the third on related work in continuous reachability. The final section is a proof of the correctness of our time-dependent Hamilton-Jacobi-Isaacs formulation, and may be omitted by the casual reader. The bulk of this chapter is taken from [99].

2.1 How to Compute the Reachable Set

In this section we formally define the reachable set for a continuous system, discuss a few of its properties, and formulate a terminal value HJI PDE whose solution describes it.

2.1.1 The Reachable Set

We model our system with the ordinary differential equation

$$\frac{dx}{dt} = \dot{x} = f(x, a, b), \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the state, a is the input for player I and b is the input for player II.

Assumption 1. The input signals are drawn from the following sets

$$a(\cdot) \in \mathfrak{A}(t) \triangleq \{\eta : [t,0] \to \mathcal{A} | \eta(\cdot) \text{ is measurable}\}$$
$$b(\cdot) \in \mathfrak{B}(t) \triangleq \{\eta : [t,0] \to \mathcal{B} | \eta(\cdot) \text{ is measurable}\}$$

where $\mathcal{A} \subset \mathbb{R}^{n_a}$ and $\mathcal{B} \subset \mathbb{R}^{n_b}$ are compact and $t \in [-T, 0]$ for some $T > 0$. We will consider input signals which agree almost everywhere to be identical.

Notice the notational difference between the instantaneous value $a \in \mathcal{A}$ of the input of player I and the input signal $a(\cdot) \in \mathfrak{A}(t)$, and likewise $b \in \mathcal{B}$ and $b(\cdot) \in \mathfrak{B}(t)$ for player II.

Assumption 2. The flow field $f : \mathbb{R}^n \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}^n$ is uniformly continuous, bounded, and Lipschitz continuous in x for fixed a and b. Consequently, given a fixed $a(\cdot) \in \mathfrak{A}(t)$, $b(\cdot) \in \mathfrak{B}(t)$ and initial point, there exists a unique trajectory solving (2.1).

Solutions of (2.1) are trajectories of our system and will be denoted by

$$\xi_f(s; x, t, a(\cdot), b(\cdot)) : [t, 0] \to \mathbb{R}^n$$

where

$$\frac{d}{ds}\xi_f(s; x, t, a(\cdot), b(\cdot)) = f(\xi_f(s; x, t, a(\cdot), b(\cdot)), a(s), b(s)) \text{ almost everywhere,}$$
$$\xi_f(t; x, t, a(\cdot), b(\cdot)) = x.$$

In words, $\xi_f(s; x, t, a(\cdot), b(\cdot))$ is the state space location at time s of a trajectory whose flow field is given by function f and whose initial condition at time $t \leq s$ was the state space location x. Along this trajectory player I has been using input signal $a(\cdot)$ and player II has been using input signal $b(\cdot)$. We use $\xi_f(\cdot; x, t, a(\cdot), b(\cdot))$ to denote the entire trajectory over all time greater than t. Note that we employ a semi-colon to distinguish between the argument s of ξ_f and the trajectory parameters x, t, $a(\cdot)$ and $b(\cdot)$. This somewhat complicated notation is necessary because at various points in the remainder of this thesis we must differentiate trajectories based on some or all of these parameters.

Assumption 3. The *target set* $\mathcal{G}_0 \subset \mathbb{R}^n$ for our reachability problem is closed and can be represented as the zero sublevel set of a bounded and Lipschitz continuous function

$g : \mathbb{R}^n \to \mathbb{R}$

$$\mathcal{G}_0 = \{x \in \mathbb{R}^n | g(x) \leq 0\}. \tag{2.2}$$

We assume that player I will try to steer the system away from the target with her input $a(\cdot)$, and player II will try to steer the system towards the target with her input $b(\cdot)$. For readers who prefer a more intuitive understanding of the inputs, consider that in many of our examples the target set will represent the capture set in a pursuit-evasion game. Our control will then be player I and the adversarial disturbance will be player II.

In a differential game setting, it is important to address what information the players know about each other's decisions. To specify our information pattern, define first a *strategy* for the second player as a map $\gamma : \mathfrak{A}(t) \to \mathfrak{B}(t)$ which specifies an input signal for player II as a function of the input signal that player I chooses. We will allow player II to use only *nonanticipative strategies*; that is strategies

$$\gamma \in \Gamma(t) \triangleq \{\beta : \mathfrak{A}(t) \to \mathfrak{B}(t) | a(r) = \hat{a}(r)\ \forall r \in [t, s] \implies \beta[a](r) = \beta[\hat{a}](r)\ \forall r \in [t, s]\}.$$

Simply put, a nonanticipative strategy may not make an input decision for $b(r)$ based on information about $a(s)$ if $s > r$. It will turn out that allowing player II to use nonanticipative strategies gives an advantage to player II over player I, but we postpone further discussion of information patterns and whether this is the correct one for our reachability purposes until section 2.1.5.

Note that in our formulation of the problem, a trajectory starts at some initial time $t < 0$ and we would like to know if it has passed into or through the target set by time zero. We will sometimes want to discuss the length of time that a trajectory has had to evolve; we adopt the differential game notation $\tau = -t$ to denote this positive quantity. We use the free variables s and r to denote times in the range $[t, 0]$.

To solve the backwards reachability problem, we want to determine the *backwards reachable set* $\mathcal{G}(\tau)$ for $\tau \in [0, T]$. Remembering that $t = -\tau$, we define this set as

$$\mathcal{G}(\tau) \triangleq \{x \in \mathbb{R}^n | \exists \gamma \in \Gamma(t), \forall a(\cdot) \in \mathfrak{A}(t), \exists s \in [t, 0], \xi_f(s; x, t, a(\cdot), \gamma[a](\cdot)) \in \mathcal{G}_0\}. \tag{2.3}$$

Informally, $\mathcal{G}(\tau)$ is the set of states from which there exists strategies for player II that for all inputs of player I will generate trajectories which lead to the target set within time τ.

2.1.2 Properties of the Reachable Set

In subsequent sections we will discuss a variety of methods for determining the reachable set $\mathcal{G}(\tau)$, but first we will state two of its important properties. In the theorems that follow let $\mathbb{B}^n(x,\delta)$ be the open ball in $\mathbb{R}^n$ around point x of size $\delta > 0$, and $\overline{\mathbb{B}}^n(x,\delta)$ be its closure.

Theorem 1. *If $\mathcal{G}_0$ is closed, then $\mathcal{G}(\tau)$ is closed. Conversely, if $\mathcal{G}_0$ is open, then $\mathcal{G}(\tau)$ is open.*

Proof. We prove the first assertion, because under Assumption 3 it applies to the case studied in the remainder of this paper. The proof for the second assertion is similar.

If $\mathcal{G}_0$ is closed, then $\mathcal{G}_0^{\complement} = \mathbb{R}^n \setminus \mathcal{G}_0$ is open. In the proof below, we show that $\mathcal{G}(\tau)^{\complement}$ is open; consequently, $\mathcal{G}(\tau)$ is closed. This proof is an adaptation of an ODE uniqueness proof from [30, section 2.12].

Recall that $t = -\tau$, consider a point $\hat{x}_t \in \mathcal{G}(\tau)^{\complement}$ and choose any $\gamma \in \Gamma(t)$. Complementing the definition (2.3), we see that there exists $a(\cdot) \in \mathfrak{A}(t)$ such that

$$\hat{x}_s = \xi_f(s; \hat{x}_t, t, a(\cdot), \gamma[a](\cdot)) \in \mathcal{G}_0^{\complement}$$

for all $s \in [t, 0]$. Since $\mathcal{G}_0^{\complement}$ is open, there exists $\epsilon > 0$ such that for any $s \in [t,0]$, $x_s \in \mathcal{G}_0^{\complement}$ for any $x_s \in \mathbb{B}^n(\hat{x}_s, \epsilon)$.

Now consider a point $x_t \in \mathbb{B}^n(\hat{x}_t, \delta)$ for some $\delta > 0$ whose value we will fix later. Defining the shorthand

$$\begin{aligned}\hat{\xi}(r) &= \xi_f(r; \hat{x}_t, t, a(\cdot), b(\cdot)),\\ \xi(r) &= \xi_f(r; x_t, t, a(\cdot), b(\cdot)),\end{aligned}$$

we can write

$$\begin{aligned}\hat{\xi}(r) &= \hat{x}_t + \int_t^r f(\hat{\xi}(\lambda), a(\lambda), b(\lambda))\, d\lambda,\\ \xi(r) &= x_t + \int_t^r f(\xi(\lambda), a(\lambda), b(\lambda))\, d\lambda,\end{aligned}$$

and hence

$$\begin{aligned}
\hat{\xi}(r) - \xi(r) &= \hat{x}_t - x_t + \int_t^r (f(\hat{\xi}(\lambda), a(\lambda), b(\lambda)) - f(\xi(\lambda), a(\lambda), b(\lambda)))\, d\lambda, \\
\|\hat{\xi}(r) - \xi(r)\| &\leq \|\hat{x}_t - x_t\| + \int_t^r \|f(\hat{\xi}(\lambda), a(\lambda), b(\lambda)) - f(\xi(\lambda), a(\lambda), b(\lambda))\|\, d\lambda, \\
\|\hat{\xi}(r) - \xi(r)\| &\leq \delta + K \int_t^r \|\hat{\xi}(\lambda) - \xi(\lambda)\|\, d\lambda,
\end{aligned} \tag{2.4}$$

where K is the Lipschitz constant for the flow field f. Letting

$$\psi(r) = \frac{\delta}{K} + \int_t^r \|\hat{\xi}(\lambda) - \xi(\lambda)\|\, d\lambda,$$

we see that $\psi(t) = \delta/K$, $\psi(r) \geq \psi(t)$, and $\dot{\psi}(r) = \|\hat{\xi}(r) - \xi(r)\|$. Rewriting (2.4) in terms of ψ yields the differential inequality

$$\dot{\psi}(r) - K\psi(r) \leq 0, \tag{2.5}$$

which we can rewrite as

$$\begin{aligned}
e^{-Kr}(\dot{\psi}(r) - K\psi(r)) &\leq 0, \\
\frac{d}{dr}(e^{-Kr}\psi(r)) &\leq 0, \\
\int_t^r \frac{d}{d\lambda}(e^{-K\lambda}\psi(\lambda))\, d\lambda &\leq 0, \\
e^{-Kr}\psi(r) - e^{-Kt}\psi(t) &\leq 0, \\
\psi(r) &\leq e^{K(r-t)}\delta/K.
\end{aligned} \tag{2.6}$$

Choose any $s \in [t, 0]$ and let $\delta = e^{-K(s-t)}K\epsilon/2$. From (2.5) and (2.6) we can see

$$\begin{aligned}
\|\hat{\xi}(s) - \xi(s)\| &= \dot{\psi}(s), \\
&\leq K\psi(s), \\
&\leq e^{K(s-t)}\delta/K, \\
&\leq e^{K(s-t)}e^{-K(s-t)}K\epsilon/(2K), \\
&\leq \epsilon/2.
\end{aligned}$$

Hence $\xi(s) \in \mathbb{B}^n(\hat{x}_s, \epsilon) \subseteq \mathcal{G}_0^{\complement}$. Since $\gamma \in \Gamma(t)$ and $s \in [t, 0]$ were arbitrary, $x_t \in \mathcal{G}(\tau)^{\complement}$. Since

$x_t \in \mathbb{B}^n(\hat{x}_t, \delta)$ was arbitrary, $\mathbb{B}^n(\hat{x}_t, \delta) \subseteq \mathcal{G}(\tau)^{\complement}$ and therefore $\mathcal{G}(\tau)^{\complement}$ is open. □

Theorem 2. *The reachable set only grows as τ increases*

$$\mathcal{G}(\tau) \subseteq \mathcal{G}(\hat{\tau})$$

for $0 \leq \tau \leq \hat{\tau} \leq T$.

The proof of this theorem is a trivial consequence of Theorem 3, and so we postpone it.

2.1.3 A Hamilton-Jacobi-Isaacs Equation for the Reachable Set

In this section we state the main theoretical result of this chapter—that the reachable set can be determined by solving for the viscosity solution of a time dependent HJI equation.

Theorem 3. *Let $\phi : \mathbb{R}^n \times [-T, 0] \to \mathbb{R}$ be the viscosity solution of the terminal value HJI PDE*

$$\begin{aligned} D_t\phi(x,t) + \min[0, H(x, D_x\phi(x,t))] = 0, \quad & \textit{for } t \in [-T,0], x \in \mathbb{R}^n; \\ \phi(x,0) = g(x), & \textit{for } x \in \mathbb{R}^n; \end{aligned} \tag{2.7}$$

where

$$H(x,p) = \max_{a \in \mathcal{A}} \min_{b \in \mathcal{B}} p^T f(x,a,b). \tag{2.8}$$

If the zero sublevel set of g describes the target set $\mathcal{G}_0$ according to (2.2), then the zero sublevel set of ϕ describes the backwards reachable set $\mathcal{G}(\tau)$

$$\mathcal{G}(\tau) = \{x \in \mathbb{R}^n \,|\, \phi(x,t) \leq 0\}. \tag{2.9}$$

The significance of this theorem is that we can harness well developed numerical schemes from the level set literature to compute accurate approximations of $\phi(x,t)$, and therefore accurate approximations of $\mathcal{G}(\tau)$, for even complicated nonlinear dynamics. The proof of this theorem is presented in section 2.4, while section 2.2 describes its implementation and chapter 3 demonstrates its application to several example systems. A completely different proof of the single player version of this theorem was developed independently in [90].

Remark. Under Assumptions 1–3, it can be shown that $\phi(x,t)$ is bounded and Lipschitz continuous in both x and t [51, Theorem 3.2].

In the past three years we have presented several alternative HJI PDE formulations for computing the backwards reachable set. In [135], the Hamiltonian was restricted to negative values only within the target set; unfortunately, the resulting potential for discontinuities in the solution makes accurate numerical implementation difficult. In [102], minimization was performed as a separate, post-processing step. While this formulation is more efficient, it is more difficult to reason about formally and may produce incorrect results when the Hamiltonian and/or target set are nonconvex. Consequently, we now advocate using the formulation above for determining reachable sets.

We conclude this section with the postponed proof.

Proof of Theorem 2. We will show that $x \in \mathcal{G}(\tau)$ implies $x \in \mathcal{G}(\hat{\tau})$ for $0 \leq \tau \leq \hat{\tau} \leq T$. Recall that $t = -\tau$ and let $\hat{t} = -\hat{\tau}$. Assume $x \in \mathcal{G}(\tau)$, which by Theorem 3 implies $\phi(x,t) \leq 0$. If ϕ is the solution of (2.7), then

$$D_t\phi(x,t) = -\min[0, H(x, D_x\phi(x,t))] \geq 0.$$

Thus, $\phi(x,\hat{t}) \leq \phi(x,t) \leq 0$, which implies $x \in \mathcal{G}(\hat{\tau})$. □

2.1.4 Hamilton-Jacobi Equations and Viscosity Solutions

The proof of Theorem 3 in section 2.4 proceeds by drawing a connection between the backwards reachable set and a zero sum differential game, from which (2.7) arises. In this section we give a brief introduction to Hamilton-Jacobi equations, their connection with optimal control and differential games, and the concept of viscosity solutions. The purpose of this section is to provide the reader with enough context to understand remarks made in the remainder of the thesis. For a comprehensive discussion of these topics, see [19].

We begin with the concept of value function and the Dynamic Programming Principle. To simplify the discussion, we will restrict ourselves to the single player optimal control case, although the definitions and results can be extended to the two player, zero sum

differential game. Consider a single input system with dynamics $\dot{x} = f(x, b)$ and trajectories $\xi_f(\cdot; x, t, b(\cdot))$. Define the *cost* or *payoff* of a trajectory as

$$C(x, t, b(\cdot)) = \int_t^0 \ell\left(\xi_f(\lambda; x, t, b(\cdot))\right)\, d\lambda + g\left(\xi_f(0; x, t, b(\cdot))\right),$$

and assume that the input will seek to minimize the cost. The components of the cost are the *running cost* $\ell : \mathbb{R}^n \to \mathbb{R}$ and the *terminal cost* $g : \mathbb{R}^n \to \mathbb{R}$. The *value function* for this problem is a function $V : \mathbb{R}^n \times [-T, 0] \to \mathbb{R}$ such that

$$V(x, t) = \inf_{b(\cdot) \in \mathfrak{B}(t)} C(x, t, b(\cdot)).$$

In words, $V(x, t)$ specifies the cost of the optimal trajectory which starts at point x at time t.

The *Dynamic Programming Principle* (DPP) provides a way to compute the value function. Assume that we have two valid input signals $b_1(\cdot) \in \mathfrak{B}(t)$ and $b_2(\cdot) \in \mathfrak{B}(t)$ for some system. Define two more signals

$$b_3(s) = b_1(s - h) \qquad \text{for } s \in [t + h, 0], h > 0$$
$$b_4(s) = \begin{cases} b_1(s), & \text{for } s \in [t, r]; \\ b_2(s), & \text{for } s \in\,]r, 0]; \end{cases} \text{for any } r \in [t, 0], s \in [t, 0].$$

Informally, $b_3(\cdot)$ is a copy of $b_1(\cdot)$ delayed by time h and $b_4(\cdot)$ is a concatenation of $b_1(\cdot)$ and $b_2(\cdot)$. If $b_3(\cdot) \in \mathfrak{B}(t + h)$ and $b_4(\cdot) \in \mathfrak{B}(t)$, then the DPP holds for that system and its corresponding value function satisfies the equation

$$V(x, t) = \min_{b(\cdot) \in \mathfrak{B}(t)} \left[\int_t^{t+h} \ell\left(\xi_f(\lambda; x, t, b(\cdot))\right)\, d\lambda + V\left(\xi_f(t + h; x, t, b(\cdot)), t + h\right) \right], \qquad (2.10)$$

for $-T \leq t \leq t + h \leq 0$ and $V(x, 0) = g(x)$. In words, (2.10) says that the best possible cost at the present time and state is given by choosing an input that minimizes the sum of the cost of using that input for a short time h and the best possible cost to go from the state achieved after using that input for time h.

If we assume that V is differentiable, we can arrive at an HJ PDE by rearranging (2.10)

and dividing by h to get

$$\min_{b(\cdot)\in\mathfrak{B}(t)}\left[\frac{V\left(\xi_f(t+h;x,t,b(\cdot)),t+h\right)-V(x,t)}{h}+\frac{\int_t^{t+h}\ell\left(\xi_f(\lambda;x,t,b(\cdot))\right)\,d\lambda}{h}\right]=0,$$

and then letting $h\to 0$

$$\begin{aligned}\min_{b\in\mathcal{B}}\left[\frac{d}{dt}V(x,t)+\ell(x)\right]&=0,\\ D_tV(x,t)+\min_{b\in\mathcal{B}}D_xV(x,t)\cdot f(x,b)+\ell(x)&=0,\\ D_tV(x,t)+H(x,D_xV(x,t))&=-\ell(x),\end{aligned}\tag{2.11}$$

where the Hamiltonian $H(x,p)=\min_{b\in\mathcal{B}}p^Tf(x,b)$ is the single player version of (2.8). If we set the running cost $\ell(x)\equiv 0$, then (2.11) becomes the single player version of (2.7).

The DPP was applied to optimal control problems by Bellman in the late fifties and to differential games by Isaacs in the sixties (for references and more details, see [19, sections I.9 and VIII.4]). Consequently, we will (somewhat inconsistently) call HJ PDEs involving a single player Hamilton-Jacobi-Bellman (HJB) equations, and those involving two players Hamilton-Jacobi-Isaacs (HJI) equations.

While the derivation above is intuitively attractive, it was recognized immediately that for all but a few cases the value function V is not differentiable. Consequently, the derivation is not technically correct and, even if it were, classical solutions to the HJ PDEs would not exist. The lack of a classical solution arises because HJ PDEs can exhibit shocks and rarefactions. To define these terms, we need to look first at the characteristics of the PDEs.

The *characteristics* of HJB and HJI PDEs correspond to optimal trajectories of the underlying system's dynamics. Because our PDE (2.7) contains no running cost, the terminal condition's values are transmitted without modification along characteristics—for example, given some optimal trajectory $\xi_f(\cdot;x,t,b(\cdot))$ of the single input system, $\phi(x,t)$ is equal to $\phi(\xi_f(0;x,t,b(\cdot)),0)$, and $\xi_f(\cdot;x,t,b(\cdot))$ is a characteristic of (2.11) with $\ell(x)\equiv 0$. A *shock* occurs when characteristics collide; in other words, there exist multiple optimal input signals and hence trajectories that lead to the same point in the state space. A *rarefaction* is in some sense the opposite, and occurs when multiple optimal input signals and hence

trajectories emanate from the same terminal point in state space. Examples of shock points can be seen in figure 3.3, and an example of a rarefaction in figure 3.4.

If shocks and rarefactions are present, a classical solution to an HJ PDE may not exist. Researchers therefore sought an appropriate definition of a non-classical or weak solution to the PDE. Viscosity solutions—first defined in [44] but described in their more useful present form in [43]—were a significant breakthrough in this process. A bounded, uniformly continuous function $\phi(x,t)$ is a *viscosity solution* to the HJ PDE

$$D_t\phi(x,t) + H(x, D_x\phi(x,t)) = 0,$$

provided that for each infinitely differentiable test function $\psi(x,t)$

- if $\phi(x_0,t_0) - \psi(x_0,t_0)$ is a local maximum of the function $\phi - \psi$, then

$$D_t\psi(x_0,t_0) + H(x, D_x\psi(x_0,t_0)) \leq 0$$

- if $\phi(x_0,t_0) - \psi(x_0,t_0)$ is a local minimum of the function $\phi - \psi$, then

$$D_t\psi(x_0,t_0) + H(x, D_x\psi(x_0,t_0)) \geq 0$$

While this definition is neither particularly enlightening nor constructive, it turns out that viscosity solutions are of great practical value. For an introduction to viscosity solutions, we recommend [50, chapter 10], which contains a reasonably understandable proof of their existence and uniqueness. The existence proof includes the derivation of the HJB equation from the DPP, and a demonstration that the viscosity solution is the appropriate weak solution to describe the value function of an optimal control problem. For the two input case [51] proves that the viscosity solution of the HJI equation is the value function for a two player, zero sum differential game.

Note that viscosity solutions are not the same as *vanishing viscosity solutions*. The latter are the solutions $\phi^{(\epsilon)}(x,t)$ in the limit $\epsilon \to 0$ of the linear second order PDE

$$D_t\phi^{(\epsilon)} + H(x, D_x\phi^{(\epsilon)}) = \epsilon D_x^2\phi^{(\epsilon)}. \tag{2.12}$$

For $\epsilon > 0$, the Laplacian operator on the right hand side (the "viscosity") guarantees that $\phi^{(\epsilon)}$ is differentiable, and hence that classical solutions to the PDE exist. As $\epsilon \to 0$, this viscosity term vanishes and a unique limit solution may not exist. However, if this vanishing viscosity solution does exist, it is the same as the (Crandall & Lions) viscosity solution defined above.

The literature also contains several other weak solutions to Hamilton-Jacobi and related PDEs, including some multivalued solutions that have applications in wave propagation and imaging problems, but we will not discuss them further here.

2.1.5 Information Patterns for Differential Games

Throughout this thesis, we have chosen to let player II select a nonanticipative strategy that can respond to the input choices of player I. In this section we discuss some possible alternatives to this information pattern. We consider four basic types of controls for the game players—open loop, state feedback, nonanticipative strategies, and anticipative strategies.

Because our reachable sets generally represent "unsafe" portions of the state space, we usually prefer to overapproximate them rather than underapproximate them. Therefore, whenever a choice must be made between giving player I or player II an advantage, we choose to give it to player II, who is trying to make the reachable set larger. If in another context player I should be given the advantage, it is straightforward to modify the definition of the reachable set (2.3) and the Hamiltonian (2.8).

An *open loop strategy* requires that both players decide their entire input signals $a(s)$ and $b(s)$ for all $s \in [t, 0]$ without any knowledge of the other players' decisions. *State feedback* allows players I and II to choose $a(s)$ and $b(s)$ respectively based on the current value of $\xi_f(s; x, t, a(\cdot), b(\cdot))$. We defined nonanticipative strategies in section 2.1.1. Our system dynamics are deterministic, so by allowing player II to make decisions about $b(s)$ with full knowledge of $a(r)$ for $r \in [t, s]$, a nonanticipative strategy gives player II all the information of state feedback, plus player I's current input $a(s)$. While player I is at a slight disadvantage under this information pattern, at a minimum she has access to sufficient information to use state feedback, because player II must declare her strategy before player I chooses a specific input and thus player I can determine the response of player II to any input signal.

An *anticipative strategy* would be equivalent to allowing player II to choose $b(s)$ based on knowledge of $a(r)$ for all $r \in [t, 0]$; in other words, player I would have to reveal her entire input signal in advance to player II.

The systems in which we are interested use state feedback controllers. Clearly the open loop pattern of information is unsuitable for verifying such systems, and the anticipative strategy model is inappropriate as well because it effectively causes player I to operate open loop. While state feedback might be a more appropriate model of our systems than nonanticipative strategies, it is not so easily turned into an HJ PDE. We have therefore chosen to use nonanticipative strategies, and give whatever advantage they confer to player II. It can be proven that the value of the differential game (2.30) under nonanticipative strategies is always less than the value under state feedback [18], and consequently we will only overapproximate the reachable set.

2.2 Implementing a Level Set Algorithm

Nonlinear PDEs such as (2.7) exhibit a number of properties that make their solutions difficult to determine either analytically or numerically; for example, even with smooth initial conditions $g(x)$ and flow field $f(x, a, b)$, the solution of (2.7) can develop kinks—locations where the derivatives become discontinuous—in finite time. However, because HJ PDEs describe a number of important physical processes, techniques have been developed to find numeric approximations of their solutions.

As described section 2.1.4, we seek the viscosity solution of (2.7). A family of algorithms called *level set methods* have been designed specifically to compute approximations to the viscosity solution for time dependent HJ PDEs with continuous initial conditions and Hamiltonians such as (2.7). In this section we examine the details of adapting level set methods to the approximation of reachable sets.

We assume throughout that the human modeler provides a way of computing the optimization over inputs a and b necessary to compute $H(x, p)$ in (2.8), and concentrate on numerically determining the zero level set of the solution ϕ to (2.7). Note that we do not require the modeler to perform a dynamic optimization over the entire input signals $a(\cdot)$ and $b(\cdot)$. In almost all of the examples we have studied so far, the static optimization of a

and b for a particular x and p is trivial—in most cases $f(x, a, b)$ is a linear function of a and b (although nonlinear in x). For an example where this optimization was not so simple, see sections 3.3 or 6.2.

2.2.1 The Numerical Scheme

As discussed in section 2.3.3, the goal of our implementation is to compute with as much accuracy as possible a signed distance function for the boundary of the reachable set. The accuracy of the derivative approximations described below is measured in terms of the order of their local truncation errors: on a grid with spacing h, an order p method for approximating a function u with a numerically computed $\hat{u}$ has error $\|u - \hat{u}\| = \mathcal{O}(h^p)$. In general, we will call any scheme with order two or greater ($p \geq 2$) a *high order* scheme.

Because our state space is $\mathbb{R}^n$, we compute an approximation of the value of $\phi(x, t)$ at the nodes of a fixed Cartesian grid in $\mathbb{R}^n \times [T, 0]$. Within (2.7), there are three terms that must be evaluated: the spatial derivative $D_x\phi(x, t)$, the Hamiltonian $H(x, p)$ and the time derivative $D_t\phi(x, t)$. One of the appealing properties of level set methods is that we can separately choose techniques for approximating each of these terms at each node using values of ϕ at the node and its neighbors.

Spatial Derivative

Traditional finite difference approximations of order p for the spatial derivative of a function represented on a grid assume that the function and at least its first $p - 1$ derivatives are continuous. Clearly this property will not hold in the presence of the kinks in $\phi(x, t)$. Nevertheless, convergent numerical approximations of $D_x\phi(x, t)$ were developed shortly after viscosity solutions were first proposed [45]. In our code, we rely primarily on a weighted, essentially non-oscillatory fifth order accurate approximation for our high fidelity computations [111, 109], although we have implemented a basic first order accurate scheme for speed [110, 125].

A key feature of all these schemes is their use of directional approximations. Consider approximating $D_x\phi(x, t)$ for $x \in \mathbb{R}$ (so $n = 1$). At a grid point x_i, there exists a *left*

approximation $D_x^- \phi$ and a *right approximation* $D_x^+ \phi$; a first order accurate version would be

$$D_x^- \phi(x_i, t) = \frac{\phi(x_i, t) - \phi(x_{i-1}, t)}{x_i - x_{i-1}},$$
$$D_x^+ \phi(x_i, t) = \frac{\phi(x_{i+1}, t) - \phi(x_i, t)}{x_{i+1} - x_i}.$$

Achieving higher order accuracy requires the use of values from more than a grid point's immediate neighbors and, as mentioned above, assumes continuity of higher derivatives. The assumption will fail near kinks, and as a result the solution will become oscillatory and unstable. *Essentially non-oscillatory* (ENO) schemes compute several different approximations to the left and right, and then choose to use only the least oscillatory. A *weighted essentially non-oscillatory* (WENO) scheme takes advantage of all the approximations in smooth regions of the solution to increase the order of accuracy, but reverts to ENO near kinks. Our fifth order accurate WENO scheme uses three neighbors on each side to compute a node's left and right approximations to $D_x \phi(x, t)$. Extension to multidimensional spaces ($n > 1$) is conceptually trivial, since the approximation of $D_x \phi(x, t)$ can be computed separately for each dimension.

It should be noted that none of these finite difference schemes can achieve better than first order accuracy in the immediate vicinity of a kink, because the first derivative does not exist at such a point. The added complexity of the schemes adds accuracy only away from these points; a property which is sometimes called *high resolution* to distinguish it from true high order accuracy. Experimentally, we have found that high resolution methods like WENO are worth the added complexity because the fully first order accurate schemes cannot deliver sufficiently accurate reachable sets.

Hamiltonian

We have chosen to use the well studied *Lax-Friedrichs* (LF) approximation

$$\hat{H}(x, p^+, p^-) \triangleq H\left(x, \tfrac{p^- + p^+}{2}\right) - \tfrac{1}{2}\alpha^T (p^+ - p^-), \tag{2.13}$$

where p^+ and p^- are the right and left approximations of p respectively and $H(x, p)$ is given by (2.8). The second term in this approximation is a high order numerical dissipation

added to damp out spurious oscillations in the solution. The components of the vector α depend on the partial derivatives of H with respect to its second argument

$$\alpha_i = \max_{p \in \mathcal{I}} \left| \frac{\partial H}{\partial p_i} \right| \tag{2.14}$$

where $\mathcal{I}$ is a hypercube containing all the values that the vector p takes on over the computational domain (see [111] for details). We can understand this dissipative term as being analogous to an $\epsilon \neq 0$ Laplacian term $\epsilon D_x^2 \phi$ in the vanishing viscosity version (2.12) of the HJ PDE. When $p^+ \neq p^-$ and $\alpha \neq 0$, we add some dissipation to the equation in order to avoid a sharp kink that would lead to numerical instability. Too much dissipation will excessively smooth the approximate solution (rounding off what should be sharp corners in the reachable set), while too little will lead to numerical instability. The amount chosen by (2.14) is sufficient to guarantee stability and experimentally appears not to be overly dissipative.

A number of other options for the numerical Hamiltonian were considered [111, 45, 109]. A *Local Lax-Friedrichs* (LLF) scheme reduces the size of the set $\mathcal{I}$ in (2.14) and therefore adds less dissipation. Experimentally, we saw little difference between LF and LLF in the examples we have studied thus far for two reasons: regular reinitialization of ϕ (see section 2.2.2) already tightly restricts the range of possible values of p, and the switching nature of the optimal inputs a and b in (2.8) mean that the Hamiltonian's partials depend only slightly on the actual value of p. For these same reasons we did not use the *Roe with entropy fix* or *Gudonov* Hamiltonians described in [111]. Both attempt to further reduce dissipation by choosing either the left or right approximation of the spatial derivative according to which is the upwind approximation; however, in our experience the slight reduction in dissipation was not worth the effort required to determine the upwind direction.

Time Derivative

We appeal to the method of lines to treat the time derivative of (2.7). From (2.13) we see how to compute $\hat{H}$ at any node, and so we can treat the value of ϕ at that node as the solution to the ordinary differential equation (ODE) $D_t \phi + \min[0, \hat{H}] = 0$. Among the many numerical ODE solvers that exist, the explicit Runge-Kutta (RK) schemes are particularly easy to implement. Like any explicit solver for time dependent PDEs, the timestep Δt

that can be taken by our RK integrator is restricted by the Courant-Friedrichs-Lewy (CFL) condition to be some flow speed dependent multiple of the spatial grid size Δx. In fact, applying standard RK schemes to the solution of HJ PDEs will lead to instability unless Δt is proportional to Δx^2, a restriction that would greatly increase computational cost for a fixed time interval $[-T, 0]$. Therefore, we use *Total Variation Diminishing* (TVD) RK schemes [129, 109], which will not introduce oscillations into the solution when Δt is proportional to Δx. We have implemented first (which is just forward Euler) and second order accurate TVD RK schemes. Because of the CFL condition, the timestep is usually much smaller than the grid spacing; consequently, it is possible to use less accurate methods in time than in space without a noticeable degradation in solution quality.

2.2.2 Practical Details

After the numerical scheme is chosen, a number of practical details must be worked out in order to produce reasonable approximations to reachable sets.

Initial Conditions

We assume that the modeler can provide a signed distance function representation for $\mathcal{G}_0$. Constructing such a function manually is straightforward for many basic geometric sets such as circles, polygons, cylinders and prisms (see section 3.1.2 for an example involving a cylinder). Using minimum, maximum and negation operators it is possible to form approximate signed distance functions for unions, intersections, complements and set differences of such basic sets. For example, if sets $\mathcal{G}_1$ and $\mathcal{G}_2$ are represented by signed distance functions $g_1(x)$ and $g_2(x)$ then

$$\begin{aligned} \mathcal{G}_1 \cup \mathcal{G}_2 &\text{ is represented by } \min[g_1(x), g_2(x)], \\ \mathcal{G}_1 \cap \mathcal{G}_2 &\text{ is represented by } \max[g_1(x), g_2(x)], \\ \mathcal{G}_1^{\complement} &\text{ is represented by } -g_1(x), \\ \mathcal{G}_1 \backslash \mathcal{G}_2 &\text{ is represented by } \max[g_1(x), -g_2(x)]. \end{aligned} \tag{2.15}$$

While the resulting functions are only approximately signed distance, they can be turned into true signed distance functions by applying reinitialization to them (see below).

Boundary Conditions

The HJ PDE (2.7) that we are trying to solve is defined over all of $\mathbb{R}^n$, and hence has no physical boundary. Unfortunately, we can numerically approximate the solution only on a finite domain, so we must introduce boundaries and enforce some form of boundary conditions. For periodic dimensions we choose our computational domain to include one complete period and enforce periodic boundary conditions; for example, the relative heading ψ_r in section 3.1. For the remaining dimensions, our options are more limited and none are physically correct. We have chosen to use an homogeneous Neumann boundary condition, which sets the directional derivative of $\phi(x,t)$ normal to the boundary to zero. This choice seems the least likely to introduce instability and thereby destroy the solution globally, although it will disturb the solution locally. Through regular reinitialization and by working on a computational domain large enough that the zero level set never approaches the boundary, however, we ensure that the boundary condition does not wreck our reachability results by disturbing the motion of the zero level set.

Reinitialization

As discussed in section 2.3.3, there are a number of advantages to having ϕ in the form of a signed distance function for the boundary of the reachable set. However, even if the modeler provides a signed distance function for the terminal conditions $g(x)$, evolution according to (2.7) can quickly distort ϕ. Physically incorrect boundary conditions on the edges of the computational domain can also cause problems. Therefore, we periodically halt the regular computation in our algorithm and reconstruct a proper signed distance representation of ϕ. Since we are only concerned with the location of the zero level set of ϕ for determining reachability, we can modify its value away from this level set as much as necessary to ensure that $\|D_x\phi\| = 1$.

Because of its wide use in level set methods, several different techniques for reinitialization have been developed [39, 132, 53, 125]. We have chosen to execute a few discrete timesteps of a solver for the PDE

$$\begin{aligned} D_{\tilde{t}}\tilde{\phi}(x,\tilde{t}) &= \text{sign}(\tilde{\phi}(x,\tilde{t}))(1 - \|D_x\tilde{\phi}(x,\tilde{t})\|), \\ \tilde{\phi}(x,0) &= \phi(x,t). \end{aligned} \tag{2.16}$$

Note that this PDE is run in an auxiliary timeframe $\tilde{t}$. If it were run to convergence, then $\|D_x\tilde{\phi}\| = 1$, but in practice we run one to ten discrete timesteps to some $\tilde{t}_f$, and then reset $\phi(x,t) = \tilde{\phi}(x,\tilde{t}_f)$ to get $\|D_x\phi\| \approx 1$. Analytically, $\text{sign}(\tilde{\phi}) = 0$ on the zero level set of ϕ, and so that level set should never move. In practice, we need to use a smoothed sign function, such as

$$\text{sign}(\phi) = \frac{\phi}{\sqrt{\phi^2 + \Delta x^2}},$$

to avoid moving the zero level set too much.

We have chosen this method for two reasons. First, its initial conditions do not require explicit determination of the zero level set of ϕ. It is difficult to perform such explicit constructions at higher than first order accuracy or to extend them to higher dimensional spaces. Second, we can use the high resolution techniques from section 2.2.1 for computing spatial and time derivatives, and so our reinitialization will not cause a loss of accuracy in our computation. It is also possible to implement a fast but accurate Gudonov solver for (2.16) [53], and so reinitialization will not introduce any added dissipation. The major disadvantage of this method is its speed when compared against competitors such as the fast marching method with explicit front construction [139, 125]. Approximately half of the execution time of our current implementation is spent in reinitialization.

Localizing Computation

The HJ PDE (2.7) describes the evolution of ϕ in all of $\mathbb{R}^n$; however, we are only interested in its zero level set. Consequently, we can restrict our effort to grid nodes near the boundary between positive and negative values of ϕ. In the level set literature this idea has been variously called *local level set* [116] or *narrowbanding* [125]. We have implemented a new variant of this method in our code [101], and typically restrict our effort to within three to six nodes on each side of the interface.

Because the boundary of the reachable set is of one dimension less than the state space, considerable savings are available for two and three dimensional problems. If the number of nodes in each dimension is n (proportional to Δx^{-1}) and the dimension d, the total number of nodes is $\mathcal{O}(n^d)$; the CFL condition on timestep means that total computational cost for a

fixed time interval $[-T, 0]$ is $\mathcal{O}(n^{d+1})$. With local level sets, we reduce computational costs back down to $\mathcal{O}(n^d)$, and we have seen this cost behavior experimentally.

2.3 Alternative Algorithms

In this section we review a variety of alternative techniques for finding reachable sets. We break these schemes into two classes. We will call a scheme *convergent* if there exists some proof that its approximation of the reachable set converges to the true reachable set as the approximation is refined; for the convergent schemes discussed below an approximation is refined by using a finer grid in computations. The other class of schemes is called *overapproximative*, because they are generally designed to guarantee that any errors in the approximation make the reachable set larger. We are not aware of any proofs of convergence for the overapproximative schemes, but none were designed with that formal goal in mind. Do not take these names too seriously, because many overapproximative techniques can be modified to produce guaranteed underapproximations, and at least one convergent technique can guarantee overapproximations as well.

Drawing on the vocabulary of PDEs, another way to differentiate these two classes of reachable set methods is as Eulerian or Lagrangian approximations. An *Eulerian* approach approximates the solution's values at the nodes of a fixed grid* using finite difference, finite element or finite volume techniques. In contrast, a *Lagrangian* approach follows the flow of the solution by computing along trajectories of the dynamics; a process that is equivalent to solving a PDE by the method of characteristics. All of the convergent schemes fall into the Eulerian category, while most of the overapproximative schemes are Lagrangian.

A final distinction between the two classes of algorithms is whether they compute forwards or backwards reachable sets. All of the convergent algorithms work backwards, while all of the overapproximative techniques were designed to work forwards. This division is directly linked to the difference between Eulerian and Lagrangian approaches and may be

*By "fixed" we mean that the mesh points do not move during computation. These algorithms may add or subtract mesh points; for example, via adaptive mesh refinement strategies. The grids are usually Cartesian, although there is no reason that the algorithms could not be implemented on irregular meshes.

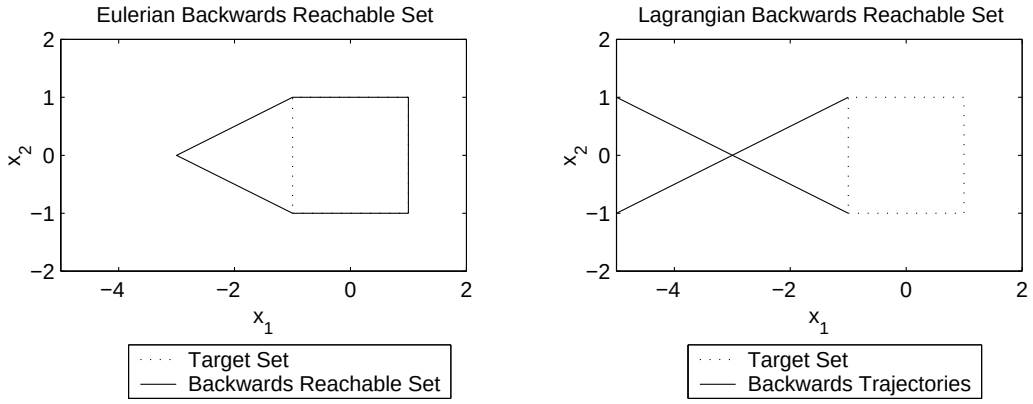

Figure 2.1: Comparing Eulerian and Lagrangian backwards reachable sets.

fundamental. Consider the simple two dimensional system

$$\begin{aligned} \dot{x}_1 &= 2, \\ \dot{x}_2 &= a, \\ a &\in [-1, +1], \\ \mathcal{G}_0 &= [-1, +1] \times [-1, +1]. \end{aligned}$$

The left side of figure 2.1 shows the (correct) backwards reachable set as computed by our Eulerian time-dependent HJI formulation. Notice that for any initial state with $x_1 < -3$, player I may choose her input $a(\cdot)$ so as to avoid the target set $\mathcal{G}_0$ entirely. In the HJI equation, this behavior manifests as a shock in the solution at $x_1 = -3$. A Lagrangian solution to this problem would track trajectories leaving the boundary of the usable part of $\mathcal{G}_0$ (see [17]); those trajectories are shown on the right side of figure 2.1 and continue beyond the shock at $x_1 = -3$. The challenge in the Lagrangian approach is to detect and stop the trajectories at the shock. Such detection is relatively easy in two dimensions (where shocks are points), but becomes almost impossible in three or more dimensions. There is a clear parallel between this difficulty in computing backwards reachable sets and the failure of characteristic based methods to correctly compute the solution of nonlinear PDEs beyond shocks. Consequently, Lagrangian techniques are poorly suited to backwards reachable set computation in the presence of shocks. In a converse way, Eulerian techniques incorrectly introduce shocks when computing the forwards reachable sets of certain systems; however, we will not further discuss the issue here.

2.3.1 Convergent Methods

In this section we examine two formulations of the reachability problem which can theoretically compute the exact reachable set $\mathcal{G}(\tau)$. Analytically, their results are equivalent to one another and to those produced by our formulation (see Theorem 5), but their practical implementations differ.

To simplify the presentation below, we restrict ourselves to the optimal control case of a single input attempting to drive the system into the target set. The dynamics in this case are

$$\dot{x} = f(x, b), \tag{2.17}$$

because this restriction is equivalent to removing player I from the game in section 2.1. As a consequence, we no longer need worry about the strategy of player II and can define the reachable set in the slightly simpler form

$$\mathcal{G}(\tau) \triangleq \{x \in \mathbb{R}^n \,|\, \exists b(\cdot) \in \mathfrak{B}(t), \exists s \in [t, 0], \xi_f(s; x, t, b(\cdot)) \in \mathcal{G}_0\}. \tag{2.18}$$

A Static Hamilton-Jacobi Formulation

The first, and perhaps most basic among all reachability formulations draws on the *time to reach* function, which we will define as

$$\mathsf{t}(x, b(\cdot)) \triangleq \begin{cases} \min\{\tau = -t | \xi(0; x, t, b(\cdot)) \in \mathcal{G}_0\}, & \text{if } \{\tau | \xi(0; x, t, b(\cdot)) \in \mathcal{G}_0\} \neq \emptyset; \\ +\infty & \text{otherwise.} \end{cases} \tag{2.19}$$

Note that $\mathsf{t}(x, b(\cdot)) \geq 0$, because our trajectories always start from some $t \leq 0$. The *minimum time to reach* function is then

$$\mathsf{T}(x) \triangleq \inf_{b(\cdot) \in \mathfrak{B}(-\infty)} \mathsf{t}(x, b(\cdot)). \tag{2.20}$$

For more details on this formulation, see [18, 19] (the definitions (2.19) and (2.20) are respectively equivalent to the functions $t_x(b)$ and $T(x)$ discussed in [18, 19], although the statement of (2.19) differs slightly due to the shifted time domain in our formulation). Those

references also discuss how to move these definitions and most of the results described below into the differential game setting.

Based on (2.19) and (2.20), it is easy to deduce the following fact.

Fact 4. *The reachable set $\mathcal{G}(\tau)$ is the τ sublevel set of the minimum time to reach function*

$$\mathcal{G}(\tau) = \{x \in \mathbb{R}^n \,|\, \mathsf{T}(x) \leq \tau\}. \tag{2.21}$$

Although there exists a static Hamilton-Jacobi equation whose solution is $\mathsf{T}(x)$, it involves infinite boundary conditions applied along a boundary which cannot always be determined a priori. Consequently, practical implementations work with the bounded *discounted minimum time to reach* function $\mathsf{TD}(x)$, which is related to $\mathsf{T}(x)$ through the *Kružkov transform*

$$\mathsf{TD}(x) \triangleq 1 - e^{-\mathsf{T}(x)}.$$

Using a single player version of the Hamiltonian (2.8)

$$H(x,p) = \min_{b \in \mathcal{B}} p^T f(x,b),$$

$\mathsf{TD}(x)$ is the solution to the following *static Hamilton-Jacobi equation*

$$\begin{aligned} \mathsf{TD} + H(x, D_x\mathsf{TD}) + 1 = 0 &\text{for } x \in \mathcal{G}_0^{\complement}, \\ \mathsf{TD}(x) = 0 &\text{for } x \in \partial\mathcal{G}_0. \end{aligned} \tag{2.22}$$

Since $0 \leq \mathsf{TD}(x) \leq 1$, we do not need to represent unbounded values in floating point arithmetic. For many systems of interest, however, we cannot guarantee that continuous solutions of (2.22) exist. A system is *small time controllable* if at every point in the state space the system's trajectory can be driven in any possible direction by some choice of the input. For a small time controllable system, the solution of (2.22) is continuous. Unfortunately, many important examples are not small time controllable; for example, our airplane models always have a positive forward velocity, so there does not exist any input which will drive the system instantaneously backward. The TD function which would arise while studying such a system would in most cases be discontinuous. Continuity is even less likely for differential game models, and without continuity level set methods cannot be applied.

Using an appropriately constructed finite difference approximation of the partial derivative $D_x\mathsf{TD}$, however, an iterative numerical algorithm for finding the viscosity solution of (2.22) has been developed and implemented [52, 20]. The resulting reachable set is $\mathcal{G}(\tau) = \{x \in \mathbb{R}^n | \mathsf{TD}(x) \leq 1 - e^{-\tau}\}$. When the true solution $\mathsf{TD}(x)$ is continuous, it can be proved that this algorithm's approximation converges to that true solution as the grid is refined. The cost of the algorithm is proportional to the number of grid points, which is usually exponential in the dimension of the state space.

Extracting the boundary of the reachable set from $\mathsf{TD}(x)$ may prove difficult for two reasons. For cases in which $\mathsf{TD}(x)$ is discontinuous, the solution's accuracy near the discontinuity—which is frequently the boundary of the reachable set—is significantly degraded. Even grid level resolution of a discontinuity's location may be difficult to achieve. In regions where $\mathsf{TD}(x)$ is smooth, the exponential mapping can cause problems as $e^{-\tau}$ gets very close to zero. This effect will limit subgrid resolution of the boundary even when using schemes with higher order accuracy on problems with completely continuous solutions.

An alternative algorithm for computing reachable sets from the minimum time to reach function is described in [34]; it is based on Dijkstra's algorithm [48] for computing minimum time paths over discrete grids. A very efficient scheme for computing $\mathsf{T}(x)$ under certain conditions on the flow field (2.17) has recently been developed for the single player case [126].

A Formulation from Viability Theory

An alternative approach to reachability is based on *viability theory* [12] and *set valued analysis* [14]. The first step is to transform our ordinary differential equation with one input (2.17) into an input free *differential inclusion*

$$\frac{dx}{dt} = \dot{x} \in F(x) \triangleq \{f(x, b) \in \mathbb{R}^n | b \in \mathcal{B}\}. \tag{2.23}$$

The right hand side F of this equation is a *set valued map*—for any x, $F(x) \subseteq \mathbb{R}^n$. For reasons related to the existence and uniqueness of solutions to (2.23) [13], viability theory typically assumes that $F(x)$ is convex and nonempty for any $x \in \mathbb{R}^n$ and that the graph of F is closed with linear growth in x; an F with these properties is called a *Marchaud map*. We make the following assumption in any subsequent discussion involving viability theory.

Assumption 4. The set valued map F is a Marchaud map. If the flow field f satisfies Assumptions 1 and 2, then the only additional constraint we need to put on F to guarantee that it is Marchaud is that its value $F(x)$ is a convex set for any $x \in \mathbb{R}^n$.

A solution of (2.23) is a trajectory $\zeta_F(\tau; x) : [0, \infty] \to \mathbb{R}^n$ such that

$$\begin{aligned} \frac{d}{d\tau}\zeta_F(\tau; x) &\in F(\zeta_F(\tau; x)) \text{ almost everywhere,} \\ \zeta_F(0; x) &= x. \end{aligned}$$

We use the symbol $\rightsquigarrow$ for set valued maps in the same way that $\to$ is used for regular functions. Define $\mathcal{S}_F : \mathbb{R}^n \rightsquigarrow C([0, +\infty], \mathbb{R}^n)$ so that $\mathcal{S}_F(x)$ is the set of absolutely continuous trajectories leading from a point $x \in \mathbb{R}^n$. Thus $\zeta_F(\cdot; x) \in \mathcal{S}_F(x)$.

With these definitions in place, we can define the τ finite horizon *capture basin* of the target set $\mathcal{G}_0$ under differential inclusion F as

$$\mathsf{Capt}_F(\mathcal{G}_0, \tau) \triangleq \{x \in \mathbb{R}^n \,|\, \exists \zeta_F(\cdot; x) \in \mathcal{S}_F(x), \exists \hat{\tau} \in [0, \tau], \zeta_F(\hat{\tau}; x) \in \mathcal{G}_0\}. \tag{2.24}$$

In words, the capture basin is the set of states from which emanate at least one trajectory leading to $\mathcal{G}_0$ in time τ or less. The equivalence of the capture basin and reachable set is stated in Theorem 5. The concept of capture basin can also be extended to the differential game setting in the form of leadership and discriminating kernels, see [13] for more details.

Based on earlier work [56, 123] an algorithm has been developed to compute $\mathsf{Capt}_F(\mathcal{G}_0, \tau)$ directly [38]. The scheme divides the state space into a fixed grid. Each point on the grid is labeled by a boolean variable which is true if that mesh point is within the capture basin—initially, only points in $\mathcal{G}_0$ will have a true value. Given a fixed timestep $\Delta\tau$, a discrete approximation of F at each mesh point is computed, from which one can determine what other mesh points can be reached in a single timestep. The procedure is iterative, where at each step a mesh point's value is set to true if it can be reached from any mesh point that is already true (this procedure is the forward Euler scheme adapted to set valued differential inclusions). Any mesh point which is true after $\tau/\Delta\tau$ timesteps is in $\mathsf{Capt}_F(\mathcal{G}_0, \tau)$. The cost of the algorithm is doubly exponential in the dimension of the state space, because in general F must be discretized for each mesh point separately. A slightly modified version of

this algorithm can be used to compute underapproximations of the viscosity solution $\mathsf{T}(x)$ of the static Hamilton-Jacobi equation [38, 22].

The approximation provided by this algorithm can be shown to converge to the true capture basin as the grid is refined [38], although no explicit convergence rate is given. Unlike the methods based on PDEs, this algorithm can guarantee an overapproximation of the reachable set if the discretization of F is sufficiently overapproximated. Because of the boolean nature of the data on the grid, this algorithm has no problem treating systems whose time to reach function is discontinuous. Conversely, absolutely no subgrid resolution of the capture basin is possible because every grid point is either completely inside or completely outside of it. Adaptive mesh refinement in the neighborhood of the capture basin's boundary has been used to develop more accurate approximations, but the cost grows very quickly as the grid is made finer.

Convergent Formulations Generate the Same Reachable Set

Theorem 5 (Equivalence of Convergent Formulations). *The following sets are equivalent*

$$\mathcal{G}(\tau) = \mathsf{Capt}_F\left(\mathcal{G}_0, \tau\right) = \{x \in \mathbb{R}^n | \mathsf{T}(x) \leq \tau\} = \{x \in \mathbb{R}^n | \phi(x,t) \leq 0\}$$

We keep our proof brief in order to maintain the focus of this section on the comparison of various reachability algorithms.

Proof. Because the set of trajectories which solve (2.17) is the same as the set of trajectories which solve (2.23) [42, section 0.4], the equivalence of $\mathcal{G}(\tau)$ and $\mathsf{Capt}_F\left(\mathcal{G}_0, \tau\right)$ is a trivial consequence of their definitions. In the viability literature, the τ sublevel set of $\mathsf{T}(x)$ is given as an alternative definition for the capture basin [13, definition 2.7.4], so clearly the second and third sets are equivalent. Theorem 3 provides the equivalence of $\mathcal{G}(\tau)$ and the zero sublevel set of $\phi(x,t)$; the proof is developed in section 2.4. □

2.3.2 Overapproximative Methods

As a group, these methods share the goal of efficiently computing an approximation of the reachable set whose error is of a guaranteed sign. All methods can produce an overapproximation, and with modified parameters some can instead produce an underapproximation. We will discuss the former, but note those cases where the latter can be found as well.

All these methods have used the same two part strategy to reach this goal, although their implementations of that strategy differ significantly. The first part of the strategy is to choose some fixed representation for the reachable sets; for example, a polyhedra. The second is to restrict the class of continuous flow field, in most cases to linear dynamics. In some cases the assumptions are strong enough that a closed form solution or overapproximation is possible. In others, the assumptions lead to an optimization problem whose solution represents an overapproximation. If that optimization problem is convex (a linear program for example), then it can be solved efficiently and reliably.

In the remainder of this section, we describe the particular assumptions made by each of these overapproximative reachability algorithms; however, because the focus of this thesis is on an analytically convergent technique, we will not discuss their implementations in detail.

Many early reachability tools [89, 142] operated on *timed automata*, which are systems with constant dynamics ($\dot{x} = c$ where $c \in \mathbb{R}^n$ is a constant). Research on such systems is still ongoing, see [24] for example. Another early tool, *HyTech* [68, 69] applied to systems in which the dynamics lay in a bounded, constant interval $\dot{x} \in [\dot{x}_{\min}, \dot{x}_{\max}]$ and the sets were convex polyhedra. A more recent offspring of HyTech is *HyperTech* [71]. It uses interval arithmetic to bound the dynamics locally in space and time, so it is much more appropriate for nonlinear systems than its predecessor. Interval arithmetic also means that its overapproximation comes with a very strong guarantee, but restricts the representation of the reachable set to (possibly nonconvex) unions of hyperrectangles. These algorithms are basically treating differential inclusions, so a single input can be handled trivially, but treating two adversarial inputs, as would arise in a differential game, is not possible.

Teaming polyhedral representations with linear or affine dynamics has been a popular strategy. Dynamics of the form $\dot{x} = Ax$ where $A \in \mathbb{R}^{n \times n}$ is a constant matrix have explicit solutions, and a convex polyhedron evolving under such a flow field remains a convex polyhedron, although the reachable set is usually nonconvex. The tool *d/dt* [46, 29, 9, 47] tracks

the motion of such convex polyhedra under linear flows, and collects their nonconvex union into "orthogonal polyhedra". This algorithm has also been adapted to derive underapproximations, since such a computation is required to guarantee their overapproximations. *CheckMate* [40, 41] uses an optimization based reachability algorithm formulated to handle general dynamics, but only guaranteed to work for linear dynamics. Another optimization based method for handling piecewise affine dynamics ($\dot{x} = Ax + c$, where $c \in \mathbb{R}^n$ is a constant) and polyhedral reachable sets is described in [25]. The algorithm in *Coho* [63, 64] uses linear programming to tackle nonlinear dynamics with bounding affine intervals, and represents reachable sets in dimensions higher than two as projections into two dimensional subspaces.

Representation size scales well with dimension for some restrictive classes of polyhedra—for example, convex polyhedra specified as the intersection of a collection of halfspaces—but in many cases the true reachable sets for systems will be grossly overapproximated by such polyhedra. Practical implementations frequently resort to representations that scale poorly but can manage at least three to six dimensions; to our knowledge none has been demonstrated on a system with higher dimension. Many of these algorithms can treat single input systems, and a few can be extended to differential game settings.

Another set representation that scales very well to higher dimension is ellipsoids. Under time varying linear dynamics with a single input and starting with an ellipsoidal target set, ellipsoids which tightly overapproximate or underapproximate the evolved shape of the target set can be computed via explicit formulas [86, 85]. The approximations can be refined by taking intersections or unions of additional ellipsoids. The *VeriSHIFT* tool [28] implements these methods.

All of the methods described previously in this section are similar in that they start with an explicit representation of the target set, and compute an explicit representation of the reachable set. An alternative approach would be to divide the state space into a finite number of sets a priori, and then compute the reachable set using a discrete algorithm. In an early version of such a scheme [84], the state space was divided into a uniform grid; this algorithm turns out to be very similar to the capture basin algorithm described in section 2.3.1, but lacked any formal convergence results. A more recent algorithm [134] works only with polynomial dynamics and the zero sublevel sets of polynomials. By partitioning

the state space with a "cylindrical algebraic decomposition" based on the system's polynomials, a discrete approximation of the dynamics can be constructed which requires fewer states and less overapproximation than does the uniform grid partitioning. Another set of analysis tools [88, 87] based on similar ideas for linear systems is under development.

In practical terms, many nonlinear systems are studied as though they were linear with varying parameters. The reachable sets generated by several linear parameter varying approximations to one particular nonlinear system are compared in [128].

Finally, there are clear ties between reachable set computation and the huge field of Lyapunov theory (for more on Lyapunov theory, see [124, chapter 5]). In its simplest form, a *Lyapunov function* is a function of the system's state which decreases in value along every trajectory of the system's dynamics. Among the many uses of Lyapunov functions is the determination of *invariant sets*, which are sets of states from which trajectories cannot escape. Because the function's value decreases along every system trajectory, the sublevel set of a Lyapunov function is an implicit surface representation of an invariant set. An example of a Lyapunov function is the minimum time to reach function (2.20).

In many applications, the reachable set of interest is also an invariant set. It should come as no surprise that for general systems, Lyapunov functions are difficult to find. However, algorithms are now available to efficiently construct quadratic Lyapunov functions (yielding ellipsoidal invariant sets) for systems with linear dynamics and multiple inputs [31]. The algorithms use linear matrix inequalities and semidefinite programming, so they can handle some types of nonlinearities in the dynamics, including sector bounded nonlinearities [74, 73].

2.3.3 Comparing the Various Techniques

They share the same general strategy, and so the overapproximative methods generally share the same strengths and weaknesses when computing reachable sets. Their representations of reachable sets are usually chosen to scale polynomially with state space dimension n, although the worst case for a few can be exponential in n; for example, orthogonal polyhedra. If their representation's size exhibits polynomial growth for practical applications, these methods score a significant win over the convergent schemes, since execution time and

memory requirements generally scale linearly with the size of the reachable set's representation. Despite this advantage, none of these schemes is appropriate to the nonlinear systems that we study. Few of the overapproximative methods can handle nonlinear dynamics, and the approximations of those that do are too coarse for practical application.

The algorithms for the three convergent methods—our time dependent Hamilton-Jacobi, the static Hamilton-Jacobi, and the capture basin approach—all compute numerical solutions on Eulerian grids. As a consequence, these algorithms are less prone to numerical instability than those implemented with moving representations (such as most overapproximative schemes), but they will fail to resolve any features of the reachable set smaller than the spacing between grid nodes. Because the number of grid nodes usually grows exponentially with dimension, these methods all suffer from Bellman's curse of dimensionality and are not immediately practical for dimension greater than four or five. On the other hand, all three schemes can handle nonlinear dynamics in a differential game setting, and make no assumptions about the shape of the reachable set. Their approximations of the reachable set are not only theoretically convergent, but also accurate to about the grid resolution in practice.

The differences between the three algorithms are more subtle, but still worth examining. In our mind, the most important is accuracy. In practice, neither the static HJ nor the capture basin algorithm can deliver subgrid resolution of the reachable set. As shown in section 3.1.4, our algorithm resolves the boundary of the reachable set to less than one tenth of a grid cell in most of the state space. This accuracy is significant because of the high cost of refining the grid—doubling the resolution of the static HJ or capture basin algorithms requires eight times as much work in three dimensions or sixteen in four. Our implementation generates a *signed distance function* representation of the reachable set: at any point x in state space, $|\phi(x,t)|$ is the distance to the nearest point on the boundary of the reachable set, and $\text{sign}(\phi(x,t))$ determines whether x is inside or outside. Consequently, for any x we can extract not only the distance to the boundary, but also the direction of the nearest point on the boundary.[†] However, the other two algorithms do have strengths. Both naturally handle state constraints, while our time dependent formulation cannot in its current form. On a grid of fixed size, both are likely to be faster than our time dependent

[†]The direction is given by $\pm D_x\phi(x,t)/\|D_x\phi(x,t)\|$, and can prove useful for synthesizing safe control inputs (see section 3.1.5).

formulation, although it has been difficult to come by comparable execution times. The capture basin algorithm can guarantee an overapproximation of the reachable set. On the other hand, if we wish to use the reachable set to synthesize optimal control inputs, it will be easier to determine those controls from the minimum time to reach function generated by the static HJ formulation.

2.4 Direct Proof of the Time Dependent Formulation

At the end of this section we prove Theorem 3. The proof depends on some results from the literature of viscosity solutions and differential games, and on the definition of a new system which has an augmented set of inputs for player II.

2.4.1 Augmenting the Dynamics

In the proof, we will use a modified set of system dynamics in which we augment player II's inputs with the scalar

$$\underline{b}(\cdot) \in \underline{\mathfrak{B}}(t) \triangleq \{\eta : [t, 0] \to [0, 1] | \eta(\cdot) \text{ is measurable}\}.$$

Define the augmented input for player II as

$$\tilde{b} = \begin{bmatrix} b & \underline{b} \end{bmatrix} \in \mathcal{B} \times [0, 1],$$

and similarly define $\tilde{\mathcal{B}}$, $\tilde{\mathfrak{B}}(t)$ and $\tilde{\Gamma}(t)$. The differential game referred to in the remainder of this section will be played with dynamics

$$\tilde{f}(x, a, \tilde{b}) \triangleq \underline{b} f(x, a, b), \tag{2.25}$$

and its trajectories will be denoted by $\xi_{\tilde{f}}(s; x, t, a(\cdot), \tilde{b}(\cdot))$.

From (2.25), we see that player II may choose to play the game with normal dynamics by taking $\underline{b} = 1$, may choose slowed dynamics with $\underline{b} \in]0, 1[$, or may choose to freeze the dynamics entirely by taking $\underline{b} = 0$. Because the latter case proves important, we will call this additional scalar $\underline{b}$ the *freezing input*. We need to add the freezing input to the system

because the HJI PDE which we introduce in the next section is only able to determine whether a trajectory is in the target set at exactly time zero. If we used this PDE on the original system, player I could "avoid" the target by driving a trajectory into the target and then out the other side before time zero. But with the freezing input available, player II can stop a trajectory's evolution if it ever enters the target set.

Clearly, there is a close connection between trajectories of the augmented system (2.25) and trajectories of the original system (2.1). We can formalize the connection through the pseudo-time variable $\sigma : [t, 0] \to [t, 0]$, which for any $\underline{b}(\cdot) \in \underline{\mathfrak{B}}(t)$ is given by

$$\sigma(s) \triangleq t + \int_t^s \underline{b}(\lambda)\, d\lambda \tag{2.26}$$

Note that $\sigma(s)$ is continuous and monotonically increasing. We will also need the (possibly discontinuous) inverse function $\sigma^{-1} : [t, \sigma(0)] \to [t, 0]$, which we define by

$$\sigma^{-1}(\rho) \triangleq \inf\{\lambda \in [t, 0] | \sigma(\lambda) \geq \rho\} \tag{2.27}$$

Lemma 6 (Equivalence of Trajectories). *For any $a(\cdot) \in \mathfrak{A}(t)$ and $\tilde{b}(\cdot) = \begin{bmatrix} b(\cdot) & \underline{b}(\cdot) \end{bmatrix} \in \tilde{\mathfrak{B}}(t)$, define σ as in (2.26) and σ^{-1} as in (2.27). Then for every trajectory of the original system, there is a trajectory of the augmented system related through the pseudo-time variable σ*

$$\xi_f(\sigma(s); x, t, a(\sigma^{-1}(\cdot)), b(\sigma^{-1}(\cdot))) = \xi_{\tilde{f}}(s; x, t, a(\cdot), \tilde{b}(\cdot))$$

for any $s \in [t, 0]$.

Proof. Define the shorthand

$$\begin{aligned} \xi_f(s) &\triangleq \xi_f(s; x, t, a(\sigma^{-1}(\cdot)), b(\sigma^{-1}(\cdot))), \\ \xi_{\tilde{f}}(s) &\triangleq \xi_{\tilde{f}}(s; x, t, a(\cdot), \tilde{b}(\cdot)). \end{aligned} \tag{2.28}$$

Then we can write

$$\begin{aligned} \xi_{\tilde{f}}(s) &= \xi_{\tilde{f}}(t) + \int_t^s \frac{d\xi_{\tilde{f}}(\lambda)}{d\lambda}\, d\lambda, \\ &= x + \int_t^s f(\xi_{\tilde{f}}(\lambda), a(\lambda), b(\lambda))\underline{b}(\lambda)\, d\lambda, \end{aligned}$$

and

$$\begin{aligned}\xi_f(\sigma(s)) &= \xi_f(t) + \int_t^{\sigma(s)} \frac{d\xi_f(\lambda)}{d\lambda}\, d\lambda, \\ &= x + \int_t^{\sigma(s)} f(\xi_f(\rho), a(\sigma^{-1}(\rho)), b(\sigma^{-1}(\rho)))\, d\rho, \\ &= x + \int_t^{s} f(\xi_f(\sigma(\lambda)), a(\lambda), b(\lambda))\underline{b}(\lambda)\, d\lambda,\end{aligned}$$

where we have used a change of variables $\rho = \sigma(\lambda)$ after the second step. From these two equations and the fact that $\underline{b}(\lambda) \in [0, 1]$,

$$\begin{aligned}\xi_f(\sigma(s)) - \xi_{\tilde{f}}(s) &= \int_t^s (f(\xi_f(\sigma(\lambda)), a(\lambda), b(\lambda)) - f(\xi_{\tilde{f}}(\lambda), a(\lambda), b(\lambda)))\underline{b}(\lambda)\, d\lambda, \\ \|\xi_f(\sigma(s)) - \xi_{\tilde{f}}(s)\| &\le \int_t^s \|(f(\xi_f(\sigma(\lambda)), a(\lambda), b(\lambda)) - f(\xi_{\tilde{f}}(\lambda), a(\lambda), b(\lambda)))\underline{b}(\lambda)\|\, d\lambda, \\ &\le \int_t^s \|f(\xi_f(\sigma(\lambda)), a(\lambda), b(\lambda)) - f(\xi_{\tilde{f}}(\lambda), a(\lambda), b(\lambda))\|\, d\lambda, \\ &\le K \int_t^s \|\xi_f(\sigma(\lambda)) - \xi_{\tilde{f}}(\lambda)\|\, d\lambda,\end{aligned} \tag{2.29}$$

where K is the Lipschitz constant for the flow field f. Letting

$$\psi(s) = \int_t^s \|\xi_f(\sigma(\lambda)) - \xi_{\tilde{f}}(\lambda)\|\, d\lambda,$$

we see that $\psi(t) = 0$, $\psi(s) \ge 0$, and $\dot{\psi}(s) = \|\xi_f(\sigma(s)) - \xi_{\tilde{f}}(s)\|$. Rewriting (2.29) in terms of ψ we get the differential inequality

$$\dot{\psi}(s) - K\psi(s) \le 0,$$

whose only solution is $\psi(s) \equiv 0$ (the steps needed to show this fact are the same as those given in the proof of Theorem 1 in section 2.1.2). Therefore $\xi_f(\sigma(s)) = \xi_{\tilde{f}}(s)$. □

2.4.2 The Differential Game and its Solution

We will work with a finite horizon differential game [17] played over time horizon $[-T, 0]$ whose dynamics are governed by the flow field (2.25). A trajectory in this game has a

terminal cost

$$C(x,t,a(\cdot),\tilde{b}(\cdot)) = g(\xi_{\tilde{f}}(0;x,t,a(\cdot),\tilde{b}(\cdot))).$$

and no running cost. The goal of player I will be to maximize this cost, while player II will try to minimize it. Consequently, the value of our differential game will be

$$\begin{aligned}\phi(x,t) &= \inf_{\tilde{\gamma}\in\tilde{\Gamma}(t)} \sup_{a(\cdot)\in\mathfrak{A}(t)} C(x,t,a(\cdot),\tilde{\gamma}[a](\cdot)) \\ &= \inf_{\tilde{\gamma}\in\tilde{\Gamma}(t)} \sup_{a(\cdot)\in\mathfrak{A}(t)} g(\xi_{\tilde{f}}(0;x,t,a(\cdot),\tilde{\gamma}[a](\cdot)))\end{aligned} \tag{2.30}$$

Lemma 7. *The value function $\phi(x,t)$ of our game is the viscosity solution of the Hamilton-Jacobi-Isaacs terminal value PDE*

$$\begin{aligned}D_t\phi(x,t) + \tilde{H}(x, D_x\phi(x,t)) = 0, \quad &\textit{for } t\in[T,0], x\in\mathbb{R}^n; \\ \phi(x,0) = g(x), &\textit{for } x\in\mathbb{R}^n;\end{aligned} \tag{2.31}$$

where

$$\tilde{H}(x,p) = \max_{a\in\mathcal{A}} \min_{\tilde{b}\in\tilde{\mathcal{B}}} p^T\tilde{f}(x,a,\tilde{b}). \tag{2.32}$$

Proof. This lemma is just a special case of Theorem 4.1 in [51]. □

2.4.3 The Proof of Theorem 3

We need one more intermediate result before proving Theorem 3.

Lemma 8. *For $t\in[T,0]$, the value function $\phi(x,t)$ given by (2.30) describes the reachable set $\mathcal{G}(\tau)$*

$$\{x\in\mathbb{R}^n \,|\, \phi(x,t) < 0\} \subseteq \mathcal{G}(\tau) \subseteq \{x\in\mathbb{R}^n \,|\, \phi(x,t)\le 0\}. \tag{2.33}$$

Proof. We prove the relations in (2.33) by showing that

$$x\in\mathcal{G}(\tau) \implies \phi(x,t)\le 0, \tag{2.34}$$

$$\phi(x,t) < 0 \implies x\in\mathcal{G}(\tau). \tag{2.35}$$

Case 1: We will assume that $x \in \mathcal{G}(\tau)$ and $\phi(x,t) > 0$ and derive a contradiction. Consider first the implications of (2.30).

$$\begin{aligned}
\phi(x,t) = \inf_{\tilde{\gamma}\in\tilde{\Gamma}(t)} \sup_{a(\cdot)\in\mathfrak{A}(t)} & C(x,t,a(\cdot),\tilde{\gamma}[a](\cdot)) > 0, \\
\Longrightarrow\ & \exists \epsilon > 0, \forall \tilde{\gamma} \in \tilde{\Gamma}(t), \sup_{a(\cdot)\in\mathfrak{A}(t)} C(x,t,a(\cdot),\tilde{\gamma}[a](\cdot)) > 2\epsilon > 0, \qquad (2.36) \\
\Longrightarrow\ & \exists \epsilon > 0, \forall \tilde{\gamma} \in \tilde{\Gamma}(t), \exists \hat{a}(\cdot) \in \mathfrak{A}(t), C(x,t,\hat{a}(\cdot),\tilde{\gamma}[\hat{a}](\cdot)) > \epsilon > 0.
\end{aligned}$$

Now consider the implications of $x \in \mathcal{G}(\tau)$. By (2.3) there is a $\gamma \in \Gamma(t)$ such that for the $\hat{a}(\cdot)$ from (2.36) and $b(\cdot) = \gamma[\hat{a}](\cdot)$ there exists $s \in [t,0]$ such that $\xi_f(s;x,t,\hat{a}(\cdot),b(\cdot)) \in \mathcal{G}_0$. By (2.2), $g(\xi_f(s;x,t,\hat{a}(\cdot),b(\cdot))) < 0$. Choose freezing input signal

$$\underline{b}(r) = \begin{cases} 1, & \text{for } r \in [t,s[; \\ 0, & \text{for } r \in [s,0]. \end{cases}$$

Combine this $\underline{b}(\cdot)$ with the $b(\cdot)$ chosen above to get $\tilde{b}(\cdot)$, an input which will generate a trajectory

$$\xi_{\tilde{f}}(r;x,t,\hat{a}(\cdot),\tilde{b}(\cdot)) = \begin{cases} \xi_f(r;x,t,\hat{a}(\cdot),b(\cdot)), & \text{for } r \in [t,s]; \\ \xi_f(s;x,t,\hat{a}(\cdot),b(\cdot)), & \text{for } r \in [s,0]. \end{cases}$$

In particular, $\xi_{\tilde{f}}(0;x,t,\hat{a}(\cdot),\tilde{b}(\cdot)) = \xi_f(s;x,t,\hat{a}(\cdot),b(\cdot))$, and so $g(\xi_{\tilde{f}}(0;x,t,\hat{a}(\cdot),\tilde{b}(\cdot))) < 0$. Since $b(\cdot) = \gamma[\hat{a}](\cdot)$ and $\underline{b}(\cdot)$ is clearly nonanticipative, $\tilde{b}(\cdot)$ is also nonanticipative and we have a contradiction of (2.36). Therefore we have proved (2.34).

Case 2: Assume $\phi(x,t) < 0$. Fix $\epsilon > 0$ such that

$$\phi(x,t) < -2\epsilon < 0.$$

By (2.30) there exists $\tilde{\gamma} \in \tilde{\Gamma}(t)$ such that

$$\sup_{a(\cdot)\in\mathfrak{A}(t)} C(x,t,a(\cdot),\tilde{b}(\cdot)) = \sup_{a(\cdot)\in\mathfrak{A}(t)} g(\xi_{\tilde{f}}(0;x,t,a(\cdot),\tilde{b}(\cdot))) < -\epsilon,$$

where $\tilde{b}(\cdot) = \tilde{\gamma}[a](\cdot)$. Therefore, for all $a(\cdot) \in \mathfrak{A}(t)$,

$$g(\xi_{\tilde{f}}(0; x, t, a(\cdot), \tilde{b}(\cdot))) < -\epsilon < 0,$$

or equivalently,

$$\xi_{\tilde{f}}(0; x, t, a(\cdot), \tilde{b}(\cdot)) \in \mathcal{G}_0.$$

So choose an arbitrary $a(\cdot) \in \mathfrak{A}(t)$. Let

$$\tilde{\gamma}[a](r) = \tilde{b}(r) = \begin{bmatrix} b(r) & \underline{b}(r) \end{bmatrix},$$

for $r \in [t, 0]$ and note that since $\tilde{b}(\cdot)$ is nonanticipative, $b(\cdot)$ must be nonanticipative. Define σ as in (2.26). Then by Lemma 6,

$$\xi_f(\sigma(0); x, t, a(\sigma^{-1}(\cdot)), b(\sigma^{-1}(\cdot))) = \xi_{\tilde{f}}(0; x, t, a(\cdot), \tilde{b}(\cdot)) \in \mathcal{G}_0.$$

We have therefore shown for arbitrary $a(\cdot) \in \mathfrak{A}(t)$ that there exists a nonanticipative $b(\cdot) \in \mathfrak{B}(t)$ and $s = \sigma(0) \in [t, 0]$ such that

$$\xi_f(s; x, t, a(\cdot), b(\cdot)) \in \mathcal{G}_0.$$

By (2.3), $x \in \mathcal{G}(\tau)$ and we have proved (2.35). □

The proof of Theorem 3 is now straightforward.

Proof of Theorem 3. From Lemma 7 we know that the value function ϕ for the differential game is the viscosity solution to (2.31). If ϕ does not develop plateaus—regions of constant value—at its zero level set, then the fact that $\mathcal{G}(\tau)$ is closed from Theorem 1 and the bounds (2.33) from Lemma 8 imply

$$\mathcal{G}(\tau) = \{x \in \mathbb{R}^n \,|\, \phi(x, t) \leq 0\}.$$

A sufficient but not necessary condition to avoid plateaus is that whenever $D_x\phi$ exists, $\|D_x\phi\| \neq 0$; this condition is enforced by the level set algorithms in our implementation via reinitialization (see section 2.2.2).

For the final step of the proof, start with $\tilde{H}$ from (2.32) and H from (2.8). Then we see that

$$\begin{aligned}
\tilde{H}(x,p) &= \max_{a\in\mathcal{A}} \min_{\tilde{b}\in\tilde{\mathcal{B}}} p^T \tilde{f}(x,a,\tilde{b}), \\
&= \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} \min_{\underline{b}\in[0,1]} p^T (\underline{b} f(x,a,b)), \\
&= \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} \min_{\underline{b}\in[0,1]} \underline{b}\, p^T f(x,a,b), \\
&= \min_{\underline{b}\in[0,1]} \underline{b} \left(\max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} p^T f(x,a,b) \right), \\
&= \min \left[0, \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} p^T f(x,a,b) \right], \\
&= \min[0, H(x,p)].
\end{aligned}$$

Consequently, the two HJI PDEs (2.7) and (2.31) are equivalent, and so ϕ is also the solution of (2.7). □

Chapter 3

Examples of Continuous Reachable Sets

The classical "game of two identical vehicles" [75] is our primary example, and in the first section of this chapter we use it to demonstrate and numerically validate our implementation. The acoustic capture example is drawn from the literature on computing viability kernels and capture basins [38]. The final example shows how we can use reachable sets to analyze the autolanding protocol followed by pilots of a complex modern commercial jetliner, so as to detect potentially confusing displays or procedures.

3.1 The Game of Two Identical Vehicles

In this section we use the algorithms from section 2.2 to compute reachability for a three dimensional kinematic model of two adversarial vehicles: the *pursuer* wishes to get within a certain distance of the *evader*. In the dynamic game literature this problem is called *the game of two identical cars* [95], and the reachable set corresponds to the set within which the pursuer can capture the evader. Our previous publications [135, 102, 100] have called this problem the *three dimensional aircraft collision avoidance example*.

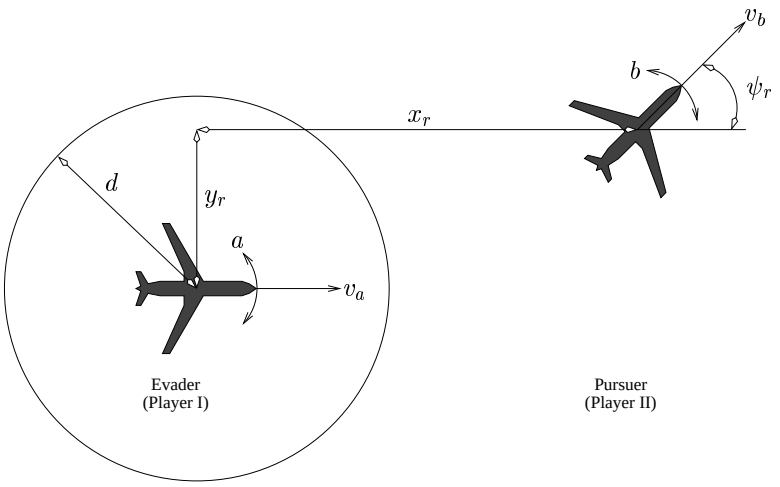

Figure 3.1: Coordinate system for the game of two identical vehicles.

3.1.1 The Model

We model our two vehicles with a commonly used, very simple kinematic system. The state of each vehicle is represented by a location in the $x-y$ plane and a heading ψ relative to the x-axis. The evolution of these states is governed by the vehicle's forward velocity v and rotational velocity ω

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \psi \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \omega \end{bmatrix}.$$

For the purposes of this example, we fix the linear velocities of the vehicles and use the angular velocities as the inputs, so v will be a constant while a and b will correspond to ω.

We say that a *collision* has occurred if the two vehicles come within distance d of one another. Our goal is to determine the set of states from which the pursuer can cause a collision to occur. Translating into reachability terms, $\mathcal{G}_0$ is the set of all states where the two vehicles are within d units of one another, the evader is player I (input a), the pursuer is player II (input b), and the set in which the pursuer can cause a collision despite the best efforts of the evader is $\mathcal{G}(\tau)$. Because $\mathcal{G}_0$ depends only on the relative positions of the

variable	meaning
x_r	relative position in flight direction of evader
y_r	relative position perpendicular to flight direction of evader
ψ_r	relative heading ($0 \leq \psi_r < 2\pi$)
z	state vector ($z = \begin{bmatrix} x_r & y_r & \psi_r \end{bmatrix}^T$)
a	angular velocity and input of evader ($\lvert a \rvert \leq 1$)
b	angular velocity and input of pursuer ($\lvert b \rvert \leq 1$)
v_a	speed of evader ($v_a = 5$)
v_b	speed of pursuer ($v_b = 5$)
d	minimum safe separation distance ($d = 5$)
$\mathcal{G}_0$	collision set ($\mathcal{G}_0 = \overline{\mathbb{B}}^2(0, d) \times [0, 2\pi] \subset \mathbb{R}^3$)

Table 3.1: Variable definitions for the game of two identical vehicles.

vehicles, we can simplify the system down to three dimensions by working in relative coordinates. Furthermore, because the variable x has special meaning in the plane, throughout this section we will denote the state vector as $z \in \mathbb{R}^3$. We fix the evader at the origin and facing along the positive x_r axis (see figure 3.1 and table 3.1). Then the pursuer's relative location and heading are described by the flow field

$$\dot{z} = \frac{d}{dt} \begin{bmatrix} x_r \\ y_r \\ \psi_r \end{bmatrix} = \begin{bmatrix} -v_a + v_b \cos\psi_r + a y_r \\ v_a \sin\psi_r - a x_r \\ b - a \end{bmatrix} = f(z, a, b). \tag{3.1}$$

The reachability algorithm we have presented can solve this problem for any choices of parameters. However, because it can be solved almost analytically (see section 3.1.3), we will focus on a particular instance in which the two vehicles' control authority and speed are identical

$$\begin{aligned} d &= 5, \\ v_a = v_b &= 5, \\ \mathcal{A} = \mathcal{B} &= [-1, +1]. \end{aligned}$$

Using the analytic solution, we can validate our numerical results.

3.1.2 The Hamilton-Jacobi Formulation

Since a collision can occur at any relative heading, the target set $\mathcal{G}_0$ depends only on x_r and y_r and includes any state within distance d of the planar origin

$$\mathcal{G}_0 = \{z \in R^3 | x_r^2 + y_r^2 \leq d^2\},$$

which can be converted into a signed distance function

$$g(z) = \sqrt{x_r^2 + y_r^2} - d, \tag{3.2}$$

for our HJ PDE's terminal conditions.

From (2.8) we see that our Hamiltonian is

$$\begin{aligned} H(z,p) &= \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} \left[p^T f(z,a,b)\right], \\ &= \max_{a\in[-1,+1]} \min_{b\in[-1,+1]} \begin{bmatrix} -p_1 v_a + p_1 v_b \cos\psi_r + p_2 v_a \sin\psi_r \\ + a(p_1 y_r - p_2 x_r - p_3) + b p_3 \end{bmatrix}, \\ &= -p_1 v_a + p_1 v_b \cos\psi_r + p_2 v_a \sin\psi_r + |p_1 y_r - p_2 x_r - p_3| - |p_3|, \end{aligned} \tag{3.3}$$

once we have plugged in the bounds on the inputs a and b.

We wish to determine the *infinite horizon reachable set*

$$\mathcal{G} \triangleq \lim_{\tau\to\infty} \mathcal{G}(\tau).$$

For this example, we find that $H(z, D_z\phi(z,t)) \geq 0$ for $\tau > 2.6$, and so we take

$$\phi(z) = \lim_{t\to-\infty} \phi(z,t) = \phi(z,-2.6).$$

Note that taking this limit $t \to -\infty$ is not appropriate for all systems. Based on the results in the next section, we know for this example that the reachable set has ceased to grow for $\tau > 2.6$, and consequently that the Hamiltonian is non-negative for $t < -2.6$. We can likewise take the limit in other examples where we can show that the Hamiltonian is guaranteed to be non-negative for all $t \leq \hat{t}$ for some bounded $\hat{t}$. But a problem arises if the Hamiltonian merely approaches zero asymptotically from below as $t \to -\infty$. In that case, Theorem 1 and hence Theorem 3 may not apply to the infinite horizon reachable set.

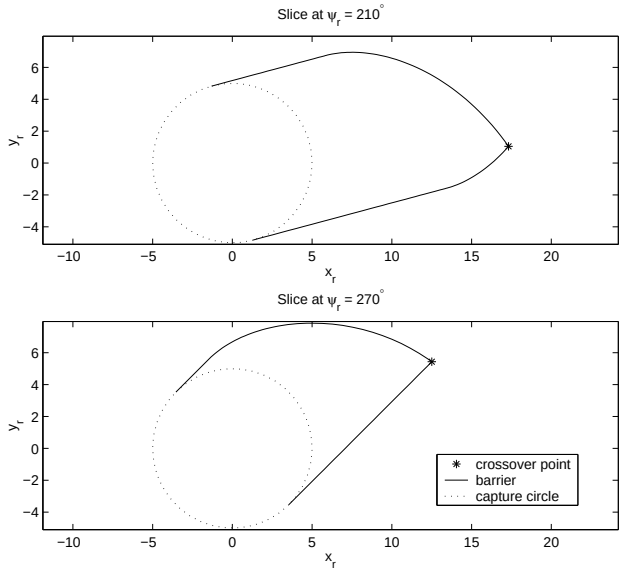

Figure 3.2: Two slices at constant ψ_r through the reachable set, as determined by Merz's method.

3.1.3 An "Almost Analytic" Solution

Using differential game theory, it is possible to determine optimal inputs for both pursuer and evader and thereby find points lying on the surface of the reachable set. In [95], A. W. Merz solved for the game with the pursuer at the origin; we have recently recreated these results, and then modified them to solve the game with the evader at the origin [98] (the two cases are not quite symmetric).

All points on the boundary of the reachable set are the endpoints of backwards time trajectories whose input signals (both evader and pursuer) are selected from a limited set of piecewise constant functions which were shown in [95] to cover the optimal solutions possible over all valid (not necessarily piecewise continuous) input signals. The evader's input is always extremal (± 1), but the pursuer will sometimes choose a zero input instead of an extremum. Note that multiple input histories may exist leading to any particular point on the boundary, but we need follow only one of those histories to find each point. Using Merz's technique, we can determine the state of these specially chosen trajectories as an explicit (although complex) function of time.

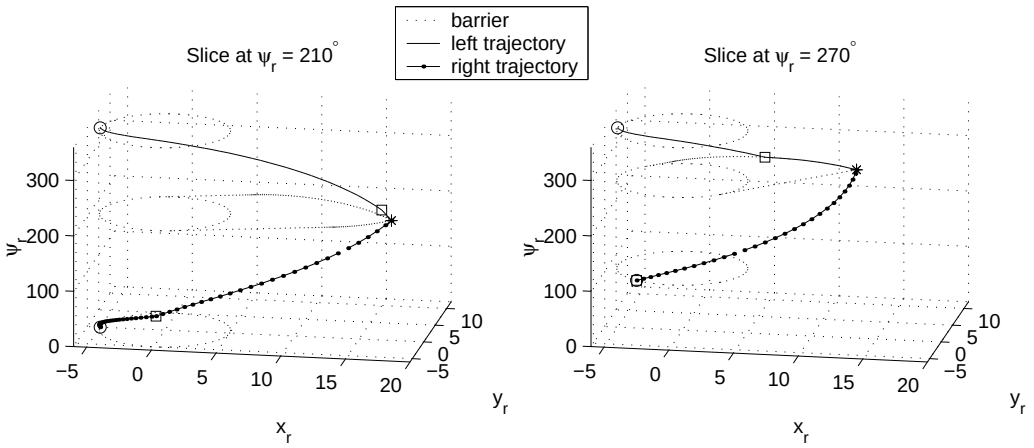

Figure 3.3: Trajectories leading to the crossover point for two slices of the reachable set, as determined by Merz's method. Because there is more than one distinct trajectory leading to these points, they are examples of shock points of the corresponding HJI PDE.

Two slices at constant ψ_r through the reachable set are shown in figure 3.2. The dotted circle is the collision set $\mathcal{G}_0$, and the solid line is the slice of $\mathcal{G}$. The "crossover points" labeled in the figure can be reached by two separate trajectories whose input signals are radically different. Those trajectories are shown in figure 3.3 for the same two slices through the reachable set. Since trajectories correspond to characteristics of the related HJI equation, the fact that two radically different characteristics meet indicates the presence of a shock in the underlying optimal flow field.

Computationally, the crossover point for a slice is the solution of an implicit trigonometric equation. Without a closed form explicit solution, these points must be determined by numerical root finding algorithms; however, such algorithms easily find roots to a relative precision significantly higher than that possible in an iterative PDE solver. All other points on the boundary can be determined to within roundoff error from the explicit trajectory functions. We call Merz's solution "almost analytic" because we can compute most points on the boundary explicitly and the remaining few to very high relative precision.

Figure 3.4 shows some of the trajectories leading to points on the upper surface of the boundary for one of the slices. Notice that many points on the boundary arise from trajectories that start at a single point $z = \begin{bmatrix} -5 & 0 & 2\pi \end{bmatrix}^T$ on the target set. As mentioned before,

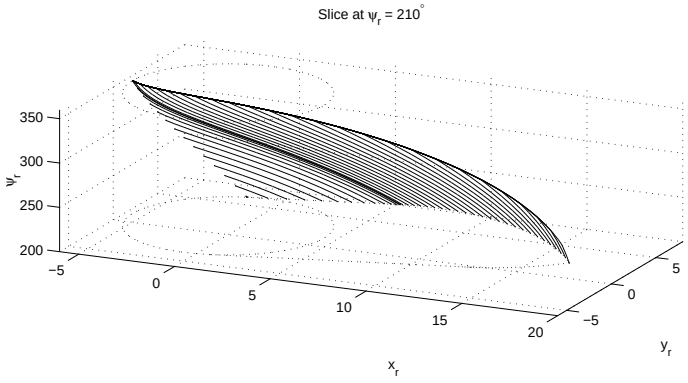

Figure 3.4: A sampling of trajectories leading to points on the boundary of the reachable set, as determined by Merz's method. Because many of these backwards time trajectories emanate from a single point in state space (lying in the upper left portion of the figure, on the collision circle), this point is the start of a rarefaction in the corresponding HJI PDE.

trajectories correspond to characteristics of the HJI equation, so this behavior demonstrates a rarefaction in the underlying optimal flow field.

3.1.4 Computational Results

Figure 3.5 shows the reachable set for this example, as computed by our algorithm on a 100^3 grid using a fifth order WENO spatial approximation of D_z, the Lax-Friedrichs approximation of the Hamiltonian (3.3), and a second order TVD RK approximation of D_t. Overlaid on the surface are approximately 2600 boundary points as determined by the almost analytic solution. Notice the sharp looking ridge that runs along the top side of the top right half of the helical bulge, and then drops to the bottom side on the bottom left half of the bulge. This ridge corresponds to the crossover points that appeared on the slices of the reachable set in the previous section. The shock in the optimal flow field at these points generates a kink in the level set function, which appears as a sharp ridge in the visualization.

Figure 3.6 shows the growth of the reachable set. On the left is the initial cylinder represented by $g(z)$ in (3.2). Figure 3.7 shows the converged set from several different angles. The rendering software for figures 3.6 and 3.7 was written by Professor Ronald Fedkiw. Animated versions of these sequences are available at [96].

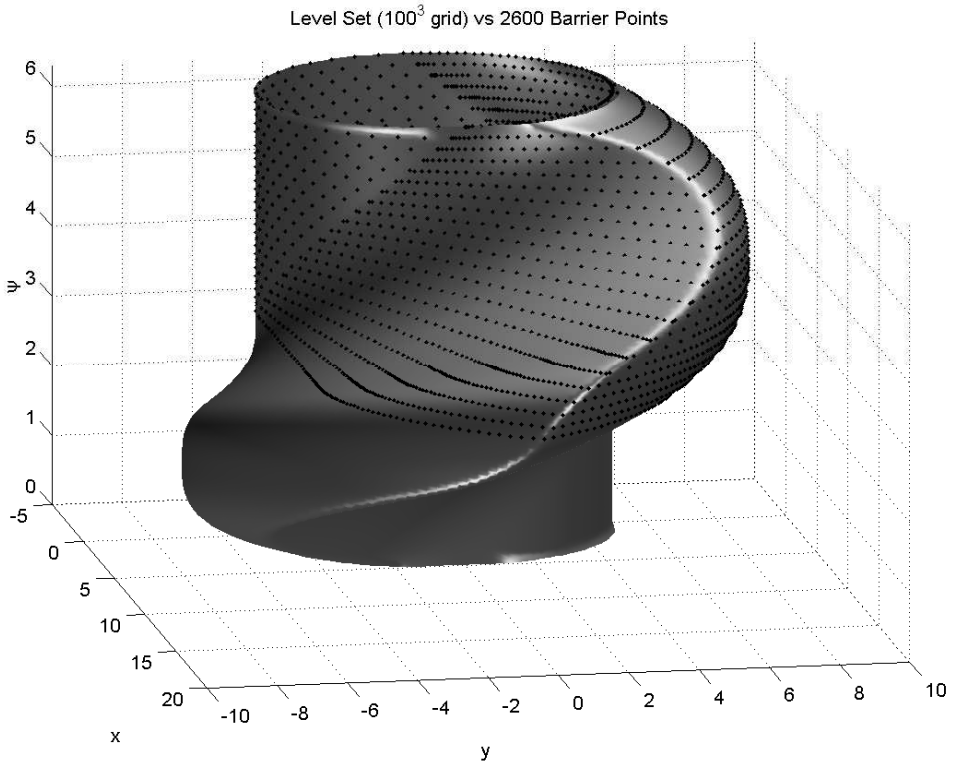

Figure 3.5: Reachable set for the game of two identical vehicles (as computed by the level set algorithm) and points on its boundary (as computed by Merz's method). Points are only shown on the top half, since the bottom is symmetric. The sharp ridge on the bulge of the set (visible as a whiter curve due to lighting) corresponds to the crossover points of the analytic solution and lies on a shock of the underlying optimal flow field. The analytic solution to the HJI PDE would have a discontinuous derivative along this curve, although dissipation in our numerical method has slightly smoothed our approximate solution.

We can build some intuition for the shape of the reachable set by considering a few horizontal slices through it. The relative heading coordinate ψ_r is the vertical coordinate in these figures, so a horizontal slice represents all possible relative planar coordinates of the two vehicles at a fixed relative heading. Now consider a horizontal slice at the vertical midpoint of the reachable set shown in figure 3.7—the slice through the most extended part of the helical bulge. The relative heading for this slice is $\psi_r = \pi$; the case in which the two aircraft have exactly opposite headings, so it is not surprising that the reachable set is largest at this point. If we look instead at a horizontal slice at the top or bottom of the reachable set ($\psi_r = 0$ or $\psi_r = 2\pi$, which are equivalent), then the slice is no more than the initial

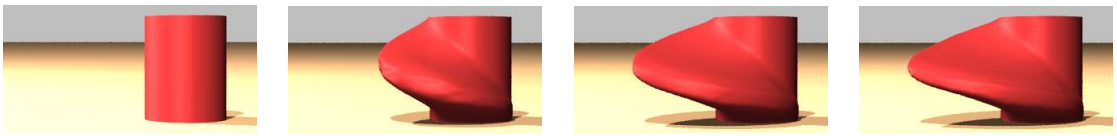

Figure 3.6: Growth of the reachable set (animation at [96]).

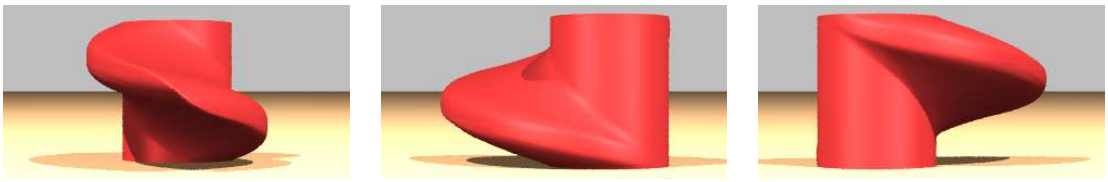

Figure 3.7: Other views of the reachable set (animation at [96]).

collision set—the aircraft start facing the same direction and have identical dynamics, so the evader can always avoid a collision unless the system starts in a collision state.

The goal of our algorithms is to produce a signed distance function representation of the reachable set. Using the almost analytic solution, we can determine a large collection of points z_i that lie exactly on the boundary, so if our PDE solver were exact we should have $\phi(z_i) = 0$ for all i. Since ϕ is a signed distance function, $|\phi(z_i)|$ measures how far z_i lies from the approximated boundary and therefore our error. If z_i is not a grid point, we use interpolation to determine the value of $\phi(z_i)$. Figure 3.8 demonstrates convergence of our algorithm in both maximum and average error for a set of approximately 60000 analytic surface points z_i as the grid resolution is increased. The grid spacing Δx for each grid size is shown for comparison. Our algorithm manages to keep the maximum error (even near the kink) to approximately one grid cell, and the average error much smaller than one tenth of a grid cell.

The flow field in this example is fully three dimensional and includes curving rarefaction and shock fronts. We know of no other three dimensional Hamilton-Jacobi problem with similarly complex behavior for which an analytic representation of the solution is available. Consequently, we believe this example presents an excellent validation tool for other numerical Hamilton-Jacobi and level set implementations. We have made Matlab scripts available for generating the boundary point sets used in this section for validation of our implementation [97] and more details on the validation procedure can be found in [98].

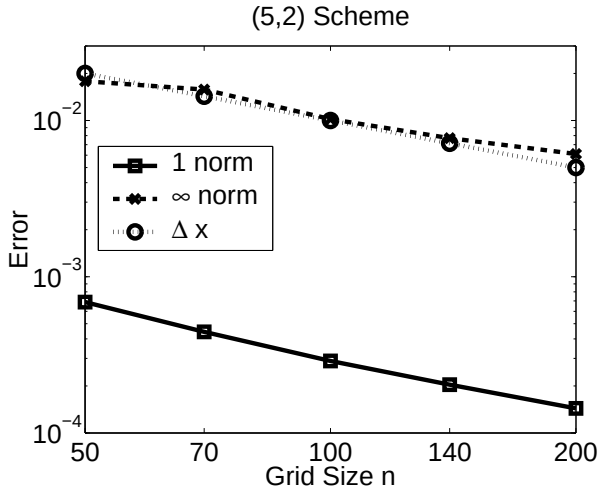

Figure 3.8: Maximum and average error in the approximation of the boundary decrease linearly as the grid is refined. Error is measured by the absolute value of the signed distance function $|\phi(z_i)|$ at each of approximately 60000 points z_i lying on the boundary of the analytically determined reachable set. The "(5,2) Scheme" refers to the use of a fifth order WENO approximation of D_x and a second order TVD-RK approximation of D_t.

3.1.5 Synthesizing a Safe Controller from the Reachable Set

It is straightforward to use the results from the previous section to evaluate the safety of a particular configuration in the game of two identical vehicles. For some relative coordinate state z and using their respective optimal input signals, the pursuer can cause a collision with the evader if $\phi(z) \leq 0$, but the evader can escape if $\phi(z) > 0$.

A more interesting application, for which we have only preliminary results, is the filtering of potentially unsafe inputs such that they guarantee safety in a minimally intrusive manner. The idea is to take some potentially unsafe input for the evader vehicle—for example, one chosen to minimize travel time or fuel consumption, or one generated by an inattentive human pilot—and modify it as little as possible to guarantee that no collision will occur. If the pursuer is far away, no filtering is necessary. The same is true if the vehicles are close together but have the same heading. If the vehicles are traveling in opposite directions on a collision course, however, the filter may have to take over complete control.

Our preliminary implementation takes as input the current relative coordinates of the vehicles z and the desired but potentially unsafe input a_u. The instantaneous dynamics of the

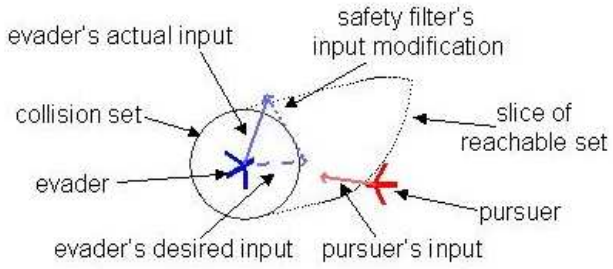

Figure 3.9: Annotated frame from the collision avoidance example animation.

Figure 3.10: Evader keeps pursuer from entering the reachable set, and hence avoids collision (animation at [96]). Note that the shape of the slice of the reachable set depends on the relative heading of the two vehicles.

system are computed $\dot{z} = f(z, a_u, b)$, conservatively assuming the worst possible pursuer input b. The projection of $\dot{z}$ onto the gradient $D_z\phi(z)$ is the component of the dynamics which is allowing the pursuer to come closer to the reachable set. By inverting the dynamics, we can determine what modification a_f must be made to a_u in order to remove this component. From $\phi(z)$ we can determine whether the pursuer is close to the reachable set's boundary. In theory, we should only have to apply a_f if the pursuer is right on the boundary of the reachable set, thereby generating an input $a_s = a_u + a_f$ which is guaranteed to keep the pursuer from entering the reachable set, and hence from causing a collision. In practice, we linearly increase the fraction of a_f applied as the pursuer approaches the reachable set in order to keep the system from stuttering along the boundary. The range at which we begin filtering depends on the coarseness of the grid used to compute $\phi(z)$.

Figure 3.9 shows an annotated frame from an animation of the collision system, and a series of frames from that animation are shown in figure 3.10. The evader starts on the left surrounded by the solid collision circle, while the pursuer starts on the right. The dotted shape surrounding the evader is the slice of the reachable set for the current relative heading of the two vehicles; for example, in the leftmost figure the vehicles have relative heading

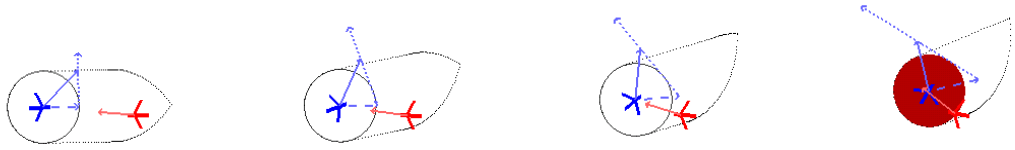

Figure 3.11: Pursuer starts within the reachable set, and can thus cause a collision despite the evader's efforts (animation at [96]).

$\psi_r \approx \pi$ and so the horizontal midplane slice of the reachable set is shown. The dashed arrow extending from the evader's nose is a_u, the dotted arrow a_f and the solid arrow a_s; the solid arrow extending from the pursuer shows its input choice b. By implementing the filter described above, the evader keeps the pursuer from entering the reachable set and thus from causing a collision as time progresses from left to right.

Figure 3.11 shows a sequence in which the pursuer starts within the reachable set and causes a collision (denoted by the solid collision circle). Although the pursuer starts well within the reachable set at the beginning of the sequence, it lies much closer to the edge of the collision circle at the end. This drift occurs in part because the pursuer in this simulation is not implementing an optimal strategy, but merely steering itself like a heat seeking missile toward the current position of the evader. Although not implemented at present, it is possible to create an optimal pursuit strategy based on state feedback and the reachable set representation $\phi(z)$; were this strategy implemented then the pursuer could come closer to the evader by the end of the sequence.

These results are only preliminary, so it should come as no surprise that the algorithm outlined above has some deficiencies. Inversion of the model dynamics to determine the appropriate filtering input a_f is prone to instability. Despite the gradual introduction of a_f as the pursuer approaches the reachable set's boundary, the system still shows chattering behavior under certain circumstances. We are currently investigating improvements to this application.

3.2 Acoustic Capture

This example is a variation of the classical homicidal chauffeur problem. As in that problem, the evader is free to travel in any direction, while the pursuer has a limited turn radius.

variable	meaning
x	relative position along pursuer's broad dimension
y	relative position along pursuer's narrow dimension
z	state vector ($z = \begin{bmatrix} x & y \end{bmatrix}^T$)
a	velocity vector and input of evader ($a \in \overline{\mathbb{B}}^2(0,1) \subset \mathbb{R}^2$)
b	angular velocity and input of pursuer ($\lvert b \rvert \leq 1$)
W_e	speed of evader ($W_e = 1.3$)
W_p	speed of pursuer ($W_p = 1.0$)
R	turn radius of pursuer ($R = 0.8$)
S	radius beyond which evader can safety use maximum speed ($S = 0.5$)
$\mathcal{G}_0$	pursuer's capture region ($\mathcal{G}_0 = [-3.5, +3.5] \times [-0.2, 0] \subset \mathbb{R}^2$)

Table 3.2: Variable definitions for the acoustic capture example.

The difference in this case is that the evader's limited speed is further reduced if she gets too close to the pursuer. This restriction might appear in situations where the evader must reduce speed as the pursuer approaches in order to keep her acoustic signal from being detected.

Our version of the problem is taken from [38], although we rotate their coordinate frame 90° counterclockwise to make x horizontal and increasing to the right. The game can be analyzed in two dimensional relative coordinates with the pursuer fixed at the origin. The relative dynamics are

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = W_p \begin{bmatrix} 0 \\ -1 \end{bmatrix} + \frac{W_p}{R}\begin{bmatrix} y \\ -x \end{bmatrix} b + 2W_e \min\left(\sqrt{x^2+y^2}, S\right) a = f(z,a,b), \tag{3.4}$$

where the variables are defined in table 3.2. From (3.4) we see that the evader can go in any direction, but her speed decreases with proximity to the pursuer if the pursuer is within distance S. The pursuer's capture region is a wide but shallow rectangle near the origin.

From (2.8) and (3.4) we find the optimal Hamiltonian

$$\begin{aligned} H(z,p) &= \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} \left[p^T f(z,a,b)\right], \\ &= \max_{a\in\overline{\mathbb{B}}^2(0,1)} \min_{b\in[-1,+1]} \begin{bmatrix} -p_2 W_p + b\frac{W_p}{R}(p_1 y - p_2 x) \\ \qquad + (p^T a)(2W_e)\min\left(\sqrt{x^2+y^2}, S\right) \end{bmatrix}, \\ &= -p_2 W_p - \frac{W_p}{R}\lvert p_1 y - p_2 x\rvert + \lVert p\rVert (2W_e)\min\left(\sqrt{x^2+y^2}, S\right), \end{aligned}$$

where we choose inputs

$$a = \frac{p}{\|p\|} \in \overline{\mathbb{B}}^2(0,1),$$
$$b = -\operatorname{sign}(p_1 y - p_2 x) \in [-1, +1].$$

The growth of the reachable set is shown in figure 3.12. The grey region is $\mathcal{G}(\tau)$ for the specified τ values, and the dashed rectangle is the pursuer's capture region $\mathcal{G}_0$. The unusual behavior in this example is the development of a hole in $\mathcal{G}(\tau)$ for $\tau \approx 2$, a hole which is entirely detached from $\mathcal{G}_0$. Because the hole does not touch $\mathcal{G}_0$, any attempt to compute its boundary by straightforward Lagrangian methods—for example, by following trajectories backwards from $\mathcal{G}_0$ in the hope that they would identify the hole—could not succeed.

The solution of this problem is computed in [38] by an algorithm for finding discriminating kernels, which are differential game relatives of the optimal control based capture basin mentioned in section 2.3.1. An alternative approach draws on Isaacs' ideas and determines the value function of this game by constructing appropriate semi-permeable surfaces [114, 115]. However, it seems unlikely that the geometrical techniques used in this latter construction could be extended to problems with dimension higher than two.

3.3 Take Off / Go Around Procedure Analysis

This section describes the reachable set analysis performed as part of a study of the pilot's protocol for automated landing in a modern commercial passenger jet. We focus here on the portions of the analysis which involve reachable set calculation; for the full protocol verification methodology, see [105].

3.3.1 Model of a Landing Aircraft

The point mass model of longitudinal dynamics that we use is taken from [100] and adapted by modifying lift and drag parameters to suit the aircraft under consideration.

$$\frac{d}{dt}\begin{bmatrix} V \\ \gamma \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{m}[T\cos\alpha - D(\alpha, V) - mg\sin\gamma] \\ \frac{1}{mV}[T\sin\alpha + L(\alpha, V) - mg\cos\gamma] \\ V\sin\gamma \end{bmatrix} \tag{3.5}$$

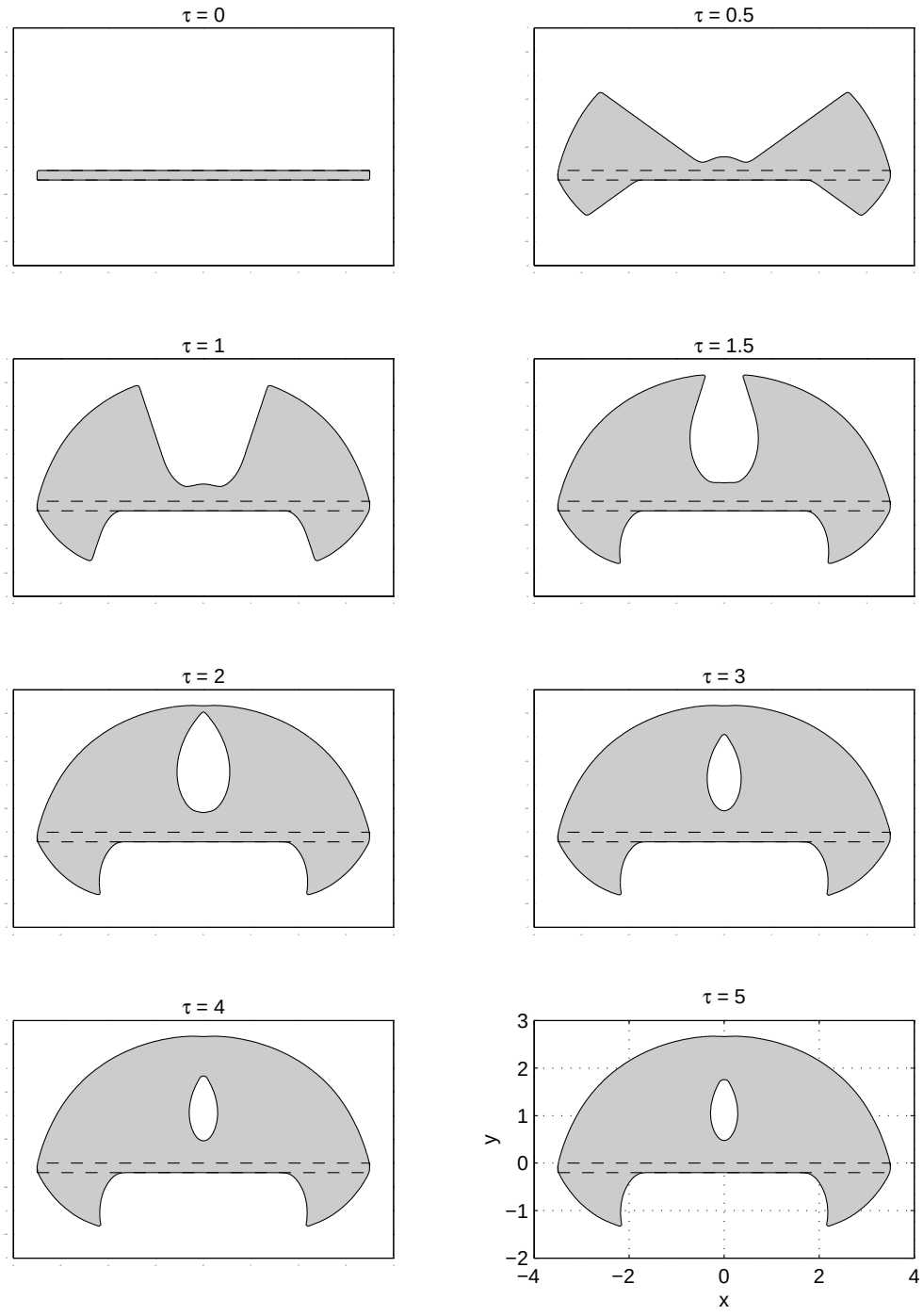

Figure 3.12: Growth of the reachable set for the acoustic capture example.

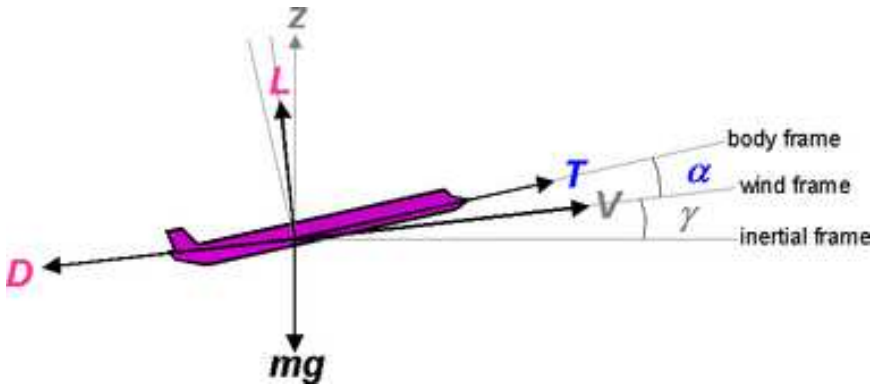

Figure 3.13: Notation for the landing aircraft example.

variable	meaning
V	speed of aircraft
γ	flightpath angle of aircraft
z	altitude of aircraft
x	state vector ($x = \begin{bmatrix} V & \gamma & z \end{bmatrix}^T$)
T	engine thrust and control input
α	angle of attack and control input
θ	pitch angle ($\theta = \alpha + \gamma$)
m	aircraft mass ($m = 190000$ kg)
g	gravity ($g = 9.81$ m/s^2)
$L(\alpha, V)$	aircraft lift
$D(\alpha, V)$	aircraft drag

Table 3.3: Variable definitions for the model of a landing aircraft.

mode	flaps	gear	C_{L_0}	C_{L_α}	C_{D_0}	K
flare	30	down	0.8212	5.105	0.025455	0.04831
TOGA-max	20	down	0.4225	5.105	0.024847	0.04831
TOGA-up	20	up	0.4225	5.105	0.019704	0.04589

Table 3.4: Aerodynamic constants for various aircraft geometries.

The state variables are speed V, flightpath angle γ and altitude z. We assume a sufficiently long runway and thus do not model horizontal distance traveled. The control inputs (player I, denoted by input a in previous sections) are engine thrust T and angle of attack α. The model does not include any disturbance inputs (player II, previously denoted by input b). The notation we use is illustrated in figure 3.13, while the variables and parameters of (3.5) are summarized in table 3.3.

The lift $L(\alpha, V)$ and drag $D(\alpha, V)$ functions are modeled based on empirical data [83] and Prandtl's lifting line theory [4, 118]:

$$\begin{aligned} L(\alpha, V) &= \tfrac{1}{2}\rho S V^2 C_L(\alpha), \\ D(\alpha, V) &= \tfrac{1}{2}\rho S V^2 C_D(\alpha), \end{aligned} \tag{3.6}$$

where $\rho = 1.225$ kg/m^3 is the density of air and $S = 427.80$ m^2 is the wing surface area. The dimensionless lift $C_L(\alpha)$ and drag $C_D(\alpha)$ coefficients depend on the geometry and flight configuration of the aircraft. We model two geometry changes that occur during the final stages of landing: extension of the wing flaps and deployment of the landing gear. The flaps increase lift and lower the speed at which the aircraft will stall, while both flaps and landing gear increase the drag. The effects of geometry changes are incorporated by changing the coefficients in

$$\begin{aligned} C_L(\alpha) &= C_{L_0} + C_{L_\alpha}\alpha, \\ C_D(\alpha) &= C_{D_0} + K C_L^2(\alpha). \end{aligned} \tag{3.7}$$

We estimated these values, which are typical for large civil aircraft, from data in [21, 122, 54, 121, 77]. Table 3.4 summarizes the coefficient values for several different geometries.

3.3.2 The Problem

Our goal is a safety study of the procedures that a pilot follows when attempting an automated landing. We would like to examine how the aircraft's automatic systems interact with the pilot through the cockpit interface—including visual displays, audible warnings and tactile feedback from the control column—as well as the protocols that the pilot learns during training and from the manuals written by the aircraft manufacturer. We are looking for instances of *mode confusion*, which occurs when the aircraft automation does not respond as a pilot expects. Mode confusion is a leading cause of *incidents*, which are small problems or irregularities that do not by themselves cause accidents, but which pilots report after they land. Since they can be precursors to accidents, the study of incidents forms an important part of accident prevention research.

To study the automated landing system, we deduced from flight manuals and the comments of pilots a typical landing protocol. During the final stage of approach (called *flare*) the engines are at idle, the flaps are fully extended, and the landing gear are down. Under normal circumstances touchdown will occur, followed by deceleration to a taxi speed (called *rollout*). If the pilot detects some danger, such as debris on the runway, she can initiate a *take off / go around* (TOGA) by pressing a button on the control column. If TOGA occurs, the engines are increased to maximum thrust and the flaps are retracted in order to avoid a landing (*TOGA-max*). Once the aircraft has begun to climb, the landing gear are retracted and the engine thrust is allowed to vary again (*TOGA-up*). The aircraft will then climb to a predefined missed approach altitude and await instructions from the control tower. Note that during this sequence the autoland system has direct control of the T and α inputs, while the pilot manually adjusts the flaps and landing gear according to the procedure outlined above.*

We study the safety of this protocol by considering whether it allows the autoland system to keep the aircraft within its *safe flight envelope*. The envelope is a range of state variables and inputs which the aircraft manufacturer and Federal Aviation Administration (FAA) have declared to be safe; for example, the aircraft must always stay above its stall speed to avoid loss of control. Table 3.5 summarizes the bounds on the state variables and inputs

*In fact, the pilot's flap and gear controls are not direct mechanical links but rather computer mediated hydraulic or electric actuators; however, the autoland system cannot directly control these actuators and must depend on the pilot to do so.

mode	V (m/s) min	V (m/s) max	γ (degrees) min	γ (degrees) max	z (m) min	α (degrees) min	α (degrees) max	T (N) range
flare	55.57	87.46	-6.0	0.0	0	-9	+15	T_{idle}
TOGA-max	63.79	97.74	-6.0	0.0	0	-8	+12	$T_{\max}$
TOGA-up	63.79	97.74	0.0	+13.3	0	-8	+12	$[\, T_{\text{idle}}, T_{\max}\,]$

Table 3.5: Safe flight envelope bounds for various flight modes.

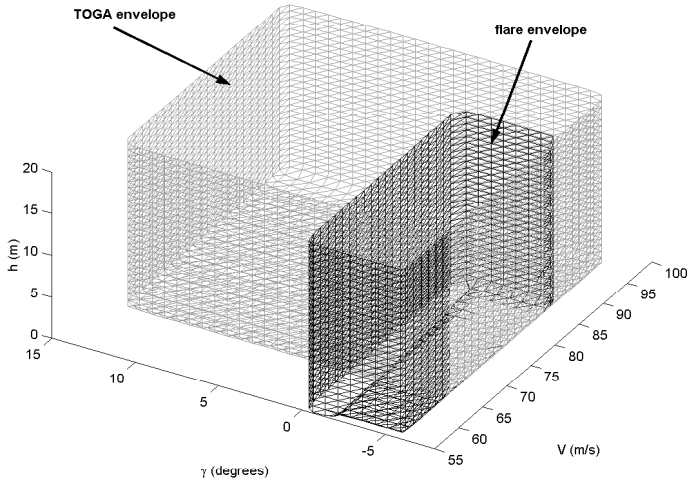

Figure 3.14: Safe flight envelopes of flare and TOGA modes compared.

for the various modes. At touchdown ($z = 0$) the envelope also requires $\theta \in [0^\circ, 12.9^\circ]$ to prevent the aircraft's tail from hitting the runway, and $\dot{z} \geq -1.829$ m/s to avoid damaging the landing gear. We do not model the dynamics after touchdown.

To analyze the aircraft's behavior, we compute the *maximally controllable subset* of the safe flight envelope; in other words, the largest set of states within the envelope for which the autolander can use its input authority over T and α to keep the aircraft within the envelope. We also call this subset the controllable envelope. We compute two controllable envelopes, one for the flare and one for the combination of the two TOGA modes. We combine the two TOGA modes because the switch between them is automatic and occurs when $\dot{z} = 0 \iff \gamma = 0$. Consequently, when computing the reachable set for the combined TOGA mode we use the dynamics for TOGA-max when $\gamma \leq 0$ and those for TOGA-up when $\gamma > 0$. Figure 3.14 shows the flare and combined TOGA envelopes as defined by the

parameters in table 3.5. The flare mode is much narrower in the γ direction because flare does not allow the aircraft to climb ($\gamma > 0$), while TOGA-up does. On the other hand, the flare mode does allow for lower speeds because the flaps are extended and hence the stall speed is lower in flare than in either of the TOGA modes.

Determining the maximally controllable subset of an envelope with a reachable set requires the same calculation but a different visualization than those used in the previous two examples. In this case, the unsafe set $\mathcal{G}_0$ is everything outside the safe flight envelope (so the exteriors of the sets in figure 3.14). As the calculation proceeds, $\mathcal{G}(\tau)$ grows and hence the controllable envelope shrinks.

The optimal Hamiltonian, as determined from (2.8) and (3.5), is complicated and we will not include all the details here. The process is the same as that presented in previous sections, except that finding the optimal value of the inputs T and α is complicated by the fact that they enter nonlinearly into the dynamics—a quadratic in α term in $D(\alpha, V)$ and the trigonometric terms $T \sin\alpha$ and $T \cos\alpha$. It can be shown that the optimal values always occur at endpoints in the range of T and almost always at endpoints in the range of α. For those few cases where the optimal occurs at an intermediate value of α, we use a quadratic approximation of the true transcendental equation for the optimal α; experimentally we have determined that the quadratic is accurate to within 1%. For details on this optimization, see [23]

3.3.3 The Reachable Set Analysis

The converged controllable envelopes for the flare and TOGA modes are shown in figure 3.15. Notice that the flare envelope extends beyond the TOGA envelope at low speeds, indicating a potential problem with the standard procedure outlined in the aircraft manuals. We tested this scenario in a commercial flight simulator by initiating a TOGA from a very low speed; the result was a stall warning. When he was asked to perform the maneuver, the pilot anticipated that flap retraction might lead to such behavior; however, there is cause for concern when any aircraft behavior during such a high tempo, low altitude period of flight is not well specified in advance. More details of how these results can be applied to formal methods in interface and procedure design can be found in [105, 106].

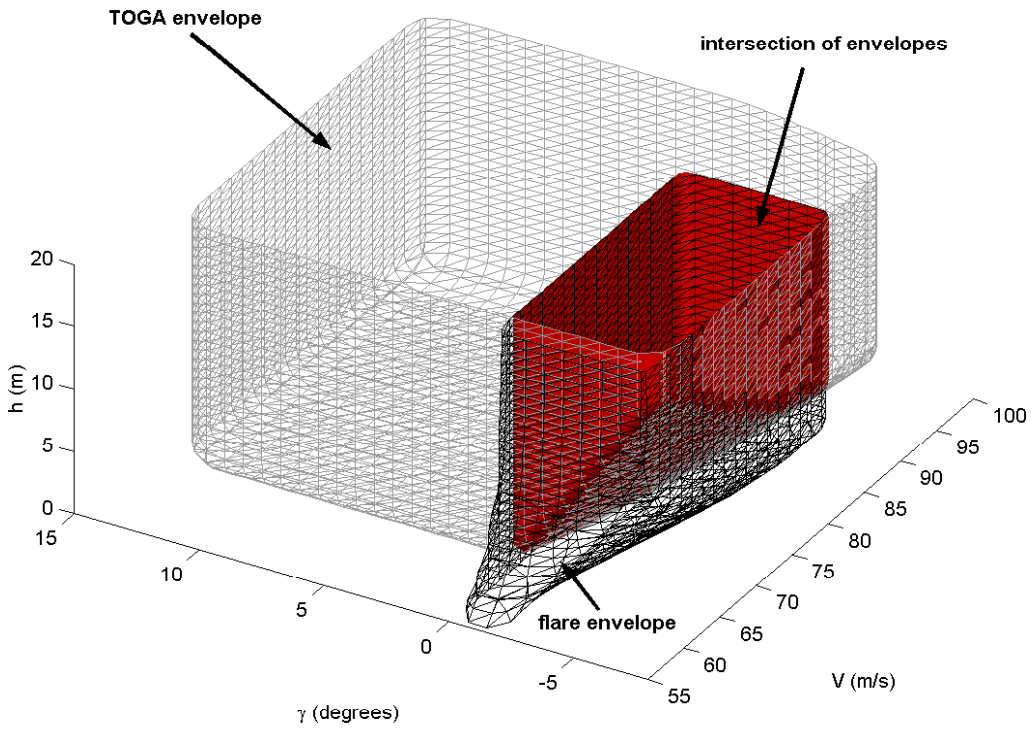

Figure 3.15: Maximally controllable envelopes of flare and TOGA modes compared

Chapter 4

Projective Overapproximation of Reachable Sets

The primary shortcoming of our formulation of reachable sets—or, in fact, of any of the convergent formulations discussed in section 2.3—is the exponential growth of computational cost with respect to the dimension of the system being analyzed. In this chapter we discuss a method that might reduce this computational burden. The bulk of this chapter is taken from [103]. It was inspired by the projection based ideas of [63, 64] for continuous spaces, and that of [59, 62, 61, 60] for discrete state spaces, as well as research which uses intersections and projections of level set functions to treat curves [36] and geometric optics [108]. We believe this method could be equally well applied (and indeed specialized) to other methods for determining reachable sets, including d/dt, Checkmate, and the methods from viability theory mentioned in section 2.3.1. In addition, this technique may be applied to reduce computational complexity in overapproximation of boundary propagation in other disciplines.

4.1 Computing the Reachable Set in a Projection

The Hamilton-Jacobi-Isaacs formulation and level set solution described in chapter 2 provides a computationally elegant method to determine the set of reachable states of a continuous dynamic game. Treating uncertainties in the system as the adversarial disturbance

inputs, this game formulation allows one to compute reachable sets conservatively: in section 3.1 the computed set represents the set of states for which there exists a disturbance action which, even if the best possible control action is played, the system state may be pushed into the target set.

The main problem with this procedure is the expense of computing the full reachable set. Instead, we wish to represent a high dimensional reachable set as the intersection of a collection of reachable sets computed in some lower dimensional subspaces. If we can formulate some way to evolve the lower dimensional reachable sets—called the projections—such that they are each an overapproximation of the full reachable set, then their intersection will also be an overapproximation. The key is to evolve the projections without referring explicitly to the full dimensional reachable set. It turns out that the HJI formulation provides this for free: in any projection, we simply augment the space of disturbance inputs with the unmodeled dimensions and form a new HJI PDE in a lower dimensional space.

Throughout the remainder of this chapter, we consider for clarity the specific case in which the true reachable set is of dimension three, and work with a set of projections in two dimensional spaces spanned by subsets of the coordinate axes. The generalization both to higher dimension, as well as to projections of different dimension, is not theoretically difficult, yet issues regarding the selection of projective subspaces are important, and will be discussed following the presentation of some examples.

4.1.1 Subspaces and Projections

We consider the state space $\mathbb{R}^3$ spanned by its coordinate axes e_1, e_2 and e_3. Let $\mathbb{Y}_i$ be the subspace spanned by coordinate axis e_i, and $\mathbb{Y}_{ij}$ the subspace spanned by coordinate axes e_i and e_j. Note that $\mathbb{Y}_{123} = \mathbb{R}^3$.

Define the *projection operators*:

- $\mathfrak{p}_i[x]$, which projects a point $x \in \mathbb{R}^3$ into the subspace $\mathbb{Y}_i$, defined as:

$$\mathfrak{p}_i[x] = x_i.$$

- $\mathfrak{p}_{ij}[x]$, which projects a point $x \in \mathbb{R}^3$ into the subspace $\mathbb{Y}_{ij}$, defined as:

$$\mathfrak{p}_{ij}[x] = \begin{bmatrix} x_i \\ x_j \end{bmatrix}.$$

  We sometimes write the pair $\begin{bmatrix} x_i & x_j \end{bmatrix}^T$ as x_{ij}.

- $\mathfrak{p}_{ij}^{-1}[y_{ij}]$, which represents the back projection of the point $y_{ij} \in \mathbb{Y}_{ij}$ into $\mathbb{R}^3$, defined as:

$$\mathfrak{p}_{ij}^{-1}[y_{ij}] = \{x \in \mathbb{R}^3 | \mathfrak{p}_{ij}[x] = y_{ij}\}.$$

  Note that $\mathfrak{p}_{ij}^{-1}[y_{ij}]$ is a subset of $\mathbb{R}^3$.

We sometimes abuse notation by applying these operators to sets instead of points. For example, if $\mathcal{X} \subset \mathbb{R}^3$, then the projection of $\mathcal{X}$ into $\mathbb{Y}_{ij}$ is represented as

$$\mathfrak{p}_{ij}[\mathcal{X}] = \{y_{ij} \in \mathbb{Y}_{ij} | \exists x \in \mathcal{X} \text{ with } \mathfrak{p}_{ij}[x] = y_{ij}\}.$$

As defined in (2.9), we represent the true, full dimensional reachable set $\mathcal{G}(\tau)$ as the zero sublevel set of the scalar function $\phi(x,t)$ (remembering that usually $\tau = -t$). In subsequent discussions we will have reason to refer to sublevel sets other than the zero sublevel set. In those cases we use a superscript to denote the particular sublevel set in which we are interested. For some set $\mathcal{M}$ represented by the signed distance function $\phi_{\mathcal{M}} : \mathbb{R}^3 \to \mathbb{R}$, and some constant $d \in \mathbb{R}$,

$$\mathcal{M}^d = \left\{x \in \mathbb{R}^3 | \phi_{\mathcal{M}}(x) \leq d\right\}.$$

The projections' reachable sets are represented by implicit surface functions defined in their respective subspaces

$$\mathcal{Y}_{ij}(\tau) = \{y_{ij} \in \mathbb{Y}_{ij} | \phi_{ij}(y_{ij}, t) \leq 0\},$$

where $\phi_{ij} : \mathbb{Y}_{ij} \times \mathbb{R} \to \mathbb{R}$. The intersection of the projections is given by

$$\begin{aligned}\mathcal{X}(\tau) &= \bigcap_{i=1}^{3} \bigcap_{j=i+1}^{3} \mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}(\tau)\right] \\ &= \{x \in \mathbb{X} | \mathfrak{p}_{ij}\left[x\right] \in \mathcal{Y}_{ij}(\tau) \text{ for } i, j \in \{1,2,3\}, j > i\}, \\ &= \{x \in \mathbb{X} | \phi_{ij}(\mathfrak{p}_{ij}\left[x\right], t) \leq 0 \text{ for } i, j \in \{1,2,3\}, j > i\}.\end{aligned} \tag{4.1}$$

Notice that $\mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}(\tau)\right]$ will be a prism in $\mathbb{R}^3$ whose cross section is $\mathcal{Y}_{ij}(\tau)$; for example, $\mathfrak{p}_{12}^{-1}\left[\mathcal{Y}_{12}(\tau)\right]$ is a prism aligned with the e_3 axis whose cross section in the e_1-e_2 plane is $\mathcal{Y}_{12}(\tau)$. Therefore, $\mathcal{X}(\tau)$ from (4.1) is simply the intersection of three orthogonal prisms.

We overload the projection operators to apply them to implicit surface functions. First, define the *depth* of a point $y_{ij} \in \mathbb{Y}_{ij}$ as

$$D(y_{ij}, t) = \min_{x \in \mathfrak{p}_{ij}^{-1}[y_{ij}]} \phi(x, t).$$

There are a number of possible ways to define a projection of the full dimensional function ϕ, but we will use the depth operator:

$$\mathfrak{p}_{ij}\left[\phi\right] : \mathbb{Y}_{ij} \times \mathbb{R} \to \mathbb{R}, \qquad \mathfrak{p}_{ij}\left[\phi\right](y_{ij}, t) = D(y_{ij}, t). \tag{4.2}$$

With this definition,

$$\mathcal{G}(\tau) = \left\{x \in \mathbb{R}^3 | \phi(x, t) \leq 0\right\} \implies \mathfrak{p}_{ij}\left[\mathcal{G}(\tau)\right] = \left\{y_{ij} \in \mathbb{Y}_{ij} | \mathfrak{p}_{ij}\left[\phi\right](y_{ij}, t) \leq 0\right\}.$$

The inverse projection for the implicit surface function of a subspace is easier to define

$$\mathfrak{p}_{ij}^{-1}\left[\phi_{ij}\right] : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}, \qquad \mathfrak{p}_{ij}^{-1}\left[\phi_{ij}\right](x, t) = \phi_{ij}(\mathfrak{p}_{ij}\left[x\right], t). \tag{4.3}$$

Under this definition, $\mathfrak{p}_{ij}^{-1}\left[\phi_{ij}\right](x, t)$ is an implicit surface function in $\mathbb{R}^3$ for the prism $\mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}\right]$ aligned normal to the e_i-e_j plane whose cross section is $\mathcal{Y}_{ij}(\tau)$.

Finally, define the *set evolution operator* $S_\tau(\cdot)$, which computes the backwards reachable set over time τ of its set valued argument. For example, $\mathcal{G}(\tau) = S_\tau(\mathcal{G}(0)) = S_\tau(\mathcal{G}_0)$. This operator is normally implemented by the HJI PDE (2.7).

4.1.2 The Linear Rotation Example

To illustrate these definitions and the projection evolution procedure, we use a simple example involving purely rotational dynamics (about the e_3 axis) and no inputs. The dynamics are given by the linear rigid body rotation

$$\dot{x} = Ax = f(x), \tag{4.4}$$

with $x \in \mathbb{R}^3$ and $A \in \mathbb{R}^{3\times 3}$

$$A = \pi \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

For this example, we will compute the forward evolution of the initial set under the rotation rather than a forward or backward reachable set, because it is easier to visualize the progress of this evolution and its projections. The entire region swept out by this evolution would be the forward reachable set. If the initial set is represented implicitly by some $\phi_0(x)$, we can compute the evolution of this initial set by solving a regular HJ PDE forward in time (note that $t \geq 0$ in this case)

$$\begin{aligned} D_t\phi(x,t) + H(x, D_x\phi(x,t)) &= 0, \\ \phi(x,0) &= \phi_0(x), \\ H(x,p) &= p \cdot f(x). \end{aligned} \tag{4.5}$$

We can safely use an HJ PDE for this forward evolution because the dynamics $f(x)$ does not contain shocks, or even flow that might be mistaken for a shock when sampled on a Cartesian grid. The projection based overapproximation method outlined below will assume that $S_t(\cdot)$ set evolution is accomplished with the forward time PDE (4.5). The method can be directly adapted to the computation of regular backward reachable sets by instead using (2.7) for $S_\tau(\cdot)$ set evolution.

Because we are working in forward time on a system with no inputs, trajectories of the

system will be denoted by $\xi_f(\cdot; x, 0)$ where

$$\begin{aligned} \frac{d}{dt}\xi_f(t; x, 0) &= A\xi_f(t; x, 0) \text{ almost everywhere,} \\ \xi_f(0; x, 0) &= x. \end{aligned}$$

This notation is an input free version of the trajectory notation used in previous chapters, with the trajectories starting at time zero. When we return to backwards reachable sets, we will return to trajectories starting at negative times.

For the purposes of this example, let $\mathcal{G}_0$ be our initial set and $\mathcal{G}(\tau)$ be the same set rotated under (4.4) for time $t = \tau$ (in the future we will call $\mathcal{G}(\tau)$ a reachable set, even though it is only a forward time evolution in this particular example). The dynamics are scaled such that $\mathcal{G}(2) = \mathcal{G}(0) = \mathcal{G}_0$. Ideally, we would like $\mathcal{G}_0$ to be a sphere of radius $r = 0.30$ centered at the point $c = \begin{bmatrix} 0.00 & 0.55 & 0.00 \end{bmatrix}^T$. Solving for the viscosity solution $\phi(x, t)$ of (4.5) with $f(x)$ from (4.4) and

$$\phi_0(x) = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + (x_3 - c_3)^2} - r \tag{4.6}$$

would generate an implicit surface representation of $\mathcal{G}(\tau)$, but would require solving (4.5) over three spatial dimensions. To reduce computational costs, we will instead seek a method of overapproximating $\mathcal{G}_0$ and $\mathcal{G}(\tau)$ that requires solving PDEs in only two spatial dimensions.

We work on three separate two dimensional projections into the subspaces $\mathbb{Y}_{12}$, $\mathbb{Y}_{13}$, and $\mathbb{Y}_{23}$. The corresponding reachable sets are $\mathcal{Y}_{12}(\tau)$, $\mathcal{Y}_{13}(\tau)$, and $\mathcal{Y}_{23}(\tau)$. The initial sets $\mathcal{Y}_{ij}(0)$ for each of these subspace reachability problems are constructed by projecting the full dimensional initial sphere $\mathcal{G}_0$ down into the subspace as $\mathcal{Y}_{ij}(0) = \mathfrak{p}_{ij}[\mathcal{G}_0]$. These $\mathcal{Y}_{ij}(0)$ and their intersection $\mathcal{X}(0)$ are shown in figure 4.1. Since $\mathcal{X}(0)$ is restricted by our projective geometry to be the intersection of three axis aligned prisms, it is unavoidably an overapproximation of the initial sphere $\mathcal{G}_0$.

4.1.3 Evolving a Projection

Our goal in this section is to develop an HJI PDE which can be applied in a lower dimensional subspace to evolve an overapproximative projection of the true reachable set, thus

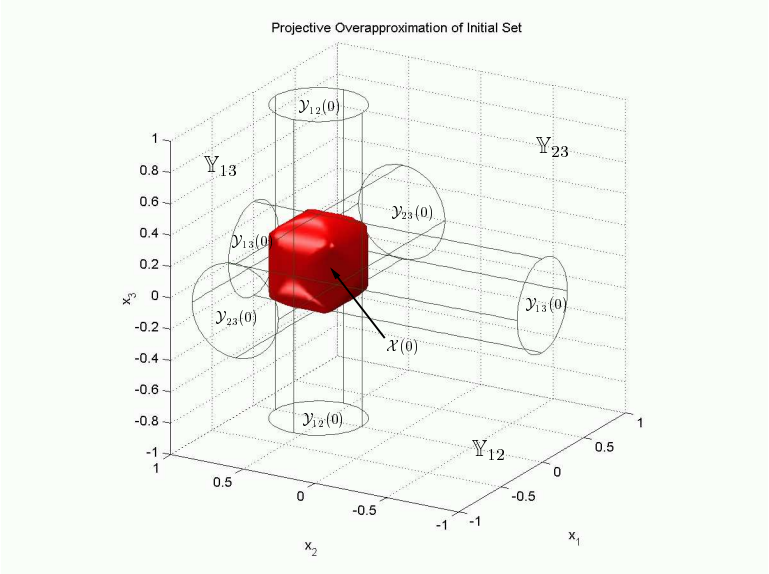

Figure 4.1: Initial projection sets for the linear rotation example.

avoiding the need to solve an expensive full dimensional PDE. First, however, we look at how to evolve an overapproximative projection using a full dimensional PDE.

Focus on a single projection whose index is ij, and denote the index of the unmodeled dimension as k. If $\mathcal{Y}_{ij}(\tau)$ is an overapproximative projection of $\mathcal{G}(\tau)$, then $\mathcal{G}(\tau) \subseteq \mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(\tau)]$. Conceptually, $\mathcal{Y}_{ij}(\tau)$ could be evolved by an inverse projection into $\mathbb{R}^3$, evolution by $\delta\tau$ and projection back down into $\mathbb{Y}_{ij}$, written as

$$\mathcal{Y}_{ij}(\tau+\delta\tau) = \mathfrak{p}_{ij}\left[S_{\delta\tau}\left(\mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}(\tau)\right]\right)\right] \tag{4.7}$$

Then

$$\begin{aligned}\mathcal{G}(\tau) \subseteq \mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}(\tau)\right] \implies & S_{\delta\tau}\left(\mathcal{G}(\tau)\right) \subseteq S_{\delta\tau}\left(\mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}(\tau)\right]\right), \\ \implies & \mathfrak{p}_{ij}\left[\mathcal{G}(\tau+\delta\tau)\right] \subseteq \mathcal{Y}_{ij}(\tau+\delta\tau).\end{aligned}$$

Consequently, we can ensure that $\mathcal{Y}_{ij}(\tau)$ remains an overapproximative projection of $\mathcal{G}(\tau)$ provided that we can perform the three steps of (4.7) on our implicit surface function representation $\phi_{ij}(x,t)$ of $\mathcal{Y}_{ij}(\tau)$. Projection is accomplished by (4.2) and inverse projection by (4.3). For this example $S_{\delta\tau}(\cdot)$ is accomplished in $\mathbb{R}^3$ by solving (4.5). Let $p(x) =$

$D_x \mathfrak{p}_{ij}^{-1}[\phi_{ij}](x,t)$. Since $\mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(\tau)]$ is a prism in $\mathbb{R}^3$, $\mathfrak{p}_k[p(x)] = p_k(x) = 0$ for all $x \in \mathbb{R}^3$; furthermore, $p_i(x)$ and $p_j(x)$ are independent of x_k. Examining the Hamiltonian of (4.5) more closely

$$\begin{aligned} H(x,p(x)) =& p(x)\cdot f(x), \\ =& p_i(x_i,x_j,x_k)f_i(x_i,x_j,x_k) + p_j(x_i,x_j,x_k)f_j(x_i,x_j,x_k) \\ &+ p_k(x_i,x_j,x_k)f_k(x_i,x_j,x_k), \\ =& p_i(x_i,x_j)f_i(x_i,x_j,x_k) + p_j(x_i,x_j)f_j(x_i,x_j,x_k). \end{aligned}$$

So the only dependence of the Hamiltonian (and thus the time evolution in general) on dimension k is through the x_k dependence in f_i and f_j. Geometrically, this dependence will manifest as a rotation of the prism $\mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(\tau)]$ so that it is no longer parallel to e_k. When this rotated prism is projected back down into $\mathbb{Y}_{ij}$, the projection's boundary will be determined by those parts of the prism that rotated the most. Maximum rotation occurs where the flow field is most closely aligned with the outward normal of the initial prism—precisely those states x where $p(x)\cdot f(x)$ is minimized (the gradient $p(x)$ points in the direction of the inward normal).

From this argument, we deduce that using the modified Hamiltonian

$$H'(x,p(x)) = \min_{x_k} p_i(x_i,x_j)f_i(x_i,x_j,x_k) + p_j(x_i,x_j)f_j(x_i,x_j,x_k) \tag{4.8}$$

in (4.5) for all $x \in \mathbb{R}^3$ would not modify the projection into $\mathbb{Y}_{ij}$ of the time evolved prism. Although the time evolved prism itself would not be the same, in the end we are only concerned with its projection.

The only reason we had for working with the projective overapproximation in $\mathbb{R}^3$ was the dependence of the time evolution operator $S_{\delta\tau}(\cdot)$ on the missing dimension x_k. After substituting the Hamiltonian (4.8) into the evolution PDE (4.5), $S_{\delta\tau}(\cdot)$ no longer has any dependence on x_k, and we can therefore work entirely in the lower dimensional $\mathbb{Y}_{ij}$.

The final concern is how to bound the range of x_k when minimizing in (4.8). We know that $x_k \in \mathbb{Y}_k$, but minimizing over such an unbounded range could lead to a negative value of arbitrarily large magnitude for (4.8). Fortunately, we have access to some sets within which all feasible reachable states should lie. If it were available, $\mathcal{G}(\tau)$ would provide a tight bound on possible values of x_k. In practice, we will have to settle for the overapproximation $\mathcal{X}(\tau)$;

however, expanding the interval of feasible x_k by using $\mathcal{X}(\tau)$ instead of $\mathcal{G}(\tau)$ can only cause the Hamiltonian (4.8) to be more negative and hence $\mathcal{Y}_{ij}(\tau)$ to grow more than necessary during the time evolution step. Since $Y_{ij}(\tau)$ was an overapproximative projection of $\mathcal{G}(\tau)$ to begin with, further excess growth cannot cause the overapproximation to fail.

To formalize the bounds on x_k, define the set valued *slice function* for some $\mathcal{M} \subset \mathbb{R}^3$ and $y_{ij} \in \mathbb{Y}_{ij}$ as

$$\begin{aligned} \mathcal{F}_k(\mathcal{M}, y_{ij}) &= \{y_k \in \mathbb{Y}_k | \exists x \in \mathcal{M} \text{ with } \mathfrak{p}_{ij}[x] = y_{ij} \text{ and } \mathfrak{p}_k[x] = y_k\}, \\ &= \{\mathfrak{p}_k[x] \in \mathbb{Y}_k | x \in \mathfrak{p}_{ij}^{-1}[y_{ij}] \cap \mathcal{M}\}. \end{aligned} \tag{4.9}$$

In words, $\mathcal{F}_k(\mathcal{M}, y_{ij})$ is a slice through $\mathcal{M}$ along the subspace $\mathbb{Y}_k$ at the point y_{ij}; its value will therefore be an interval in $\mathbb{Y}_k$. If $\mathcal{M}$ is described by the zero sublevel set of function $\phi_{\mathcal{M}} : \mathbb{R}^3 \to \mathbb{R}$, then we can write a mathematical description of $\mathcal{F}_k$

$$\mathcal{F}_k(\mathcal{M}, y_{ij}) = \{y_k \in \mathbb{Y}_k | \phi_{\mathcal{M}}(y_i, y_j, y_k) \leq 0\}. \tag{4.10}$$

Given this definition, we can formulate a time evolution HJI PDE operating entirely in $\mathbb{Y}_{ij}$ for the implicit surface function $\phi_{ij}(y_{ij}, t)$ of the overapproximative projection $\mathcal{Y}_{ij}(\tau)$. Instead of (4.5), use

$$\begin{aligned} D_t\phi_{ij}(y_{ij}, t) + H(y_{ij}, D_x\phi_{ij}(y_{ij}, t)) &= 0, \\ \phi_{ij}(y_{ij}, 0) &= \mathfrak{p}_{ij}[\phi_0](y_{ij}), \end{aligned} \tag{4.11}$$

for those $y_{ij} \in \mathfrak{p}_{ij}[\mathcal{X}(\tau)]$, with Hamiltonian

$$H(y_{ij}, p) = \min_{y_k \in \mathcal{F}_k(\mathcal{M}, y_{ij})} p_i f_i(y_i, y_j, y_k) + p_j f_j(y_i, y_j, y_k), \tag{4.12}$$

where $\mathcal{M}$ is either $\mathcal{G}(\tau)$ or $\mathcal{X}(\tau)$.

The derivation above is very informal, but its conclusion has a fascinating implication. Comparing (4.12) with (2.8), we see that the unmodeled dimension is in effect a disturbance input to the lower dimensional subspace's dynamics.

This observation leads to an alternative interpretation of (4.11) and (4.12). For the linear rotation example, $\mathcal{G}(\tau)$ is the set of trajectory points $\xi_f(t; x, 0)$ for those trajectories with

initial points $\xi_f(0; x, 0) \in \mathcal{G}_0$. If $\mathcal{Y}_{ij}(\tau)$ is to be a projective overapproximation of $\mathcal{G}(\tau)$, then $\mathcal{Y}_{ij}(\tau)$ must contain $\mathfrak{p}_{ij}[\xi_f(t; x, 0)]$ for all these trajectories (since it is an overapproximation, it may also contain other points). Consider any time $s \in [0, t]$ and the point $\xi_f(s; x, 0)$ along the full dimensional trajectory. By choosing the unmodeled dimension y_k from the set $\mathcal{F}_k(\mathcal{G}(s), y_{ij})$, we allow $y_k = \mathfrak{p}_k[\xi_f(s; x, 0)]$. Therefore

$$\mathfrak{p}_{ij}\left[\dot{\xi}_f(s; x, 0)\right] = \mathfrak{p}_{ij}\left[f\left(\mathfrak{p}_i[\xi_f(s; x, 0)], \mathfrak{p}_j[\xi_f(s; x, 0)], \mathfrak{p}_k[\xi_f(s; x, 0)]\right)\right]$$

will be among the possible flow fields for the subspace's dynamics. Since s was arbitrary, $\mathfrak{p}_{ij}[\xi(\cdot; x, 0)]$ is a feasible trajectory of the subspace's dynamic system, and so

$$\mathfrak{p}_{ij}[\xi_f(t; x, 0)] \in \mathcal{Y}_{ij}(\tau).$$

Conjecture 9. *Let $\mathcal{G}(\tau)$ be time evolved by some HJI PDE in $\mathbb{R}^3$ and $\mathcal{Y}_{ij}(\tau)$ by some HJI PDE in $\mathbb{Y}_{ij}$. If the unmodeled dimension $x_k \in \mathbb{Y}_k$ of the full dimensional system dynamics $\dot{x} = f(x)$ is treated as a disturbance input to the subspace's dynamics, then*

$$\mathfrak{p}_{ij}[\mathcal{G}(\tau)] \subseteq \mathcal{Y}_{ij}(\tau),$$

where that input x_k is drawn from a slice $\mathcal{F}_k(\mathcal{M}, y_{ij})$ of an appropriate $\mathcal{M}$ for points $y_{ij} \in \mathbb{Y}_{ij}$.

We initially formulated this conjecture based on our numerical success in computing overapproximating projections. Sections 4.1.4 and 4.2 showcase some of those results. In the remainder of this section we outline what might be required to prove the conjecture, and then discuss some implementation details.

If $\mathcal{M} = \mathcal{G}(\tau)$, proving the conjecture requires showing that $\mathcal{F}_k(\mathcal{G}(\tau), y_{ij})$ is a valid set from which to draw disturbance inputs such that the viscosity solution of the appropriate HJI PDE (either (4.11)–(4.12) or (2.7)) will still solve for the reachable set in which we are interested. The problem is that the input constraint set $\mathcal{F}_k(\mathcal{G}(\tau), y_{ij})$ depends on both time t and state y_{ij}. In chapter 2 we turned the computation of a backwards reachable set into a terminal payoff differential game and used results in [51] to show that the differential game could be solved with an HJI PDE; however, those results assumed that the control and disturbance input constraint sets were constant. State dependent input constraints were

investigated in [27], but only for the optimal control case (no disturbance inputs). It is not clear whether a differential game with time and state dependent input constraints would satisfy a dynamic programming principle. Without satisfying such a principle, it is unlikely that the viscosity solution of an HJI PDE would solve the differential game.

In practical terms, we do not have access to $\mathcal{G}(\tau)$ and must use $\mathcal{M} = \mathcal{X}(\tau)$. To prove the conjecture in this case would require the additional step of showing that

$$\mathcal{G}(\tau) \subseteq \mathcal{X}(\tau) \implies S_{\delta\tau}\left(\mathcal{G}(\tau)\right) \subseteq S_{\delta\tau}\left(\mathcal{X}(\tau)\right).$$

While we are investigating methods of proving or disproving the conjecture, we have decided to concentrate our effort on implementation of the projection technique rather than its theoretical aspects, in order to determine whether it can be applied to real problems. A number of implementation details arise when solving (4.11) and (4.12), of which we briefly describe the three most important.

- In practice, the unmodeled dimension should be chosen from a slightly bloated version of $\mathcal{X}(\tau)$ to avoid the chance that $\mathcal{F}_k(\mathcal{X}(\tau), y_{ij}) = \emptyset$ for some y_{ij} on the boundary of $\mathcal{Y}_{ij}(\tau)$. Choosing d as some small multiple of the grid spacing, we use $\mathcal{F}_k(\mathcal{X}^d(\tau), y_{ij})$ instead.

- The computational domain in $\mathbb{Y}_{ij}$ is always larger than $\mathcal{Y}_{ij}(\tau)$. Assuming that we keep d relatively small (to avoid excessive overapproximation), for those $y_{ij} \notin \mathfrak{p}_{ij}\left[\mathcal{X}^d(\tau)\right]$, we will still get $\mathcal{F}_k(\mathcal{X}^d(\tau), y_{ij}) = \emptyset$. One way of solving (4.11) and (4.12) in those cases is to use velocity extension [1] to extend the flow field artificially into $\mathcal{Y}_{ij}(\tau)^{\complement}$.

- Some projections approximate the reachable set better than others; however, each projection is individually an overapproximation of the reachable set, so if $\mathfrak{p}_i\left[\mathcal{Y}_{ij}(\tau)\right] \subset \mathfrak{p}_i\left[\mathcal{Y}_{ik}(\tau)\right]$, then we know that the extra range in $\mathfrak{p}_i\left[\mathcal{Y}_{ik}(\tau)\right]$ is not actually feasible. Thus, we can clip $\mathcal{Y}_{ik}(\tau)$ along dimension x_i until $\mathfrak{p}_i\left[\mathcal{Y}_{ij}(\tau)\right] = \mathfrak{p}_i\left[\mathcal{Y}_{ik}(\tau)\right]$. More generally, we can safely clip any portions of $\mathcal{Y}_{ij}(\tau)$ which lie outside of $\mathfrak{p}_{ij}\left[\mathcal{X}(\tau)\right]$. Without this clipping process, poorly behaved projections can quickly grow larger than practical computational domains.

4.1.4 Evolving the Linear Rotation Example's Projections

The presentation in the previous section was somewhat abstract, so in this section we will apply the algorithm to the example from section 4.1.2. Consider how to evolve the initial projective overapproximation $\mathcal{Y}_{13}(0)$. From (4.2) and (4.6)

$$\phi_{13}(x_1, x_3, 0) = \sqrt{(x_1 - c_1)^2 + (x_3 - c_3)^2} - r,$$

which is a circle in $\mathbb{Y}_{13}$. We can evolve $\mathcal{Y}_{13}(\tau)$ by solving the HJI PDE

$$D_t\phi_{13}(x_1, x_3, t) + H\left(x_1, x_3, D_{x_1}\phi_{13}(x_1, x_3, t) D_{x_3}\phi_{13}(x_1, x_3, t)\right) = 0, \tag{4.13}$$

with Hamiltonian (using the dynamics (4.4))

$$H(x_1, x_3, p_1, p_3) = \min_{x_2 \in \mathcal{F}_2(\mathcal{X}(\tau), x_1, x_3)} \pi(-p_1 x_2 + p_3 0). \tag{4.14}$$

While $\mathcal{F}_2(\mathcal{X}(\tau), x_1, x_3)$ is a set valued function of x_1 and x_3, for illustration we can describe its value (an interval of $\mathbb{Y}_2$) at a few points for $t = 0$ based upon (4.2) and (4.6)

$$\begin{aligned} \mathcal{F}_2(\mathcal{X}(0), 0, 0) &= [c_2 - r, c_2 + r], \\ \mathcal{F}_2(\mathcal{X}(0), r, 0) &= [c_2, c_2]. \end{aligned}$$

PDEs similar to (4.13)–(4.14) are used for $\mathcal{Y}_{12}(\tau)$ and $\mathcal{Y}_{23}(\tau)$.

Figure 4.2 shows the results of applying the projective evolution algorithm to the linear rotation example. The upper left figure shows the initial conditions and is the same as figure 4.1. The remaining subplots show the progress of the overapproximation through a half rotation of the dynamics. By $t = 1$, the projection $\mathcal{Y}_{13}(\tau)$ has grown from its initial circle to a square. This growth occurs because of the freedom in choosing x_2 in (4.14). Similar growth occurs in $\mathcal{Y}_{23}(\tau)$ because there is freedom in choosing x_1 for the dynamics in $\mathbb{Y}_{23}$. In contrast, $\mathcal{Y}_{12}(\tau)$ remains a circle, because the free dimension x_3 in $\mathbb{Y}_{12}$ has no effect on the dynamics (4.4). In fact, $\mathcal{Y}_{13}(\tau)$ and $\mathcal{Y}_{23}(\tau)$ would grow larger than the squares shown were it not for the clipping procedure mentioned in the previous section. Figure 4.3 compares $\mathcal{X}(\tau)$ with the true reachable set $\mathcal{G}(\tau)$ at a variety of times in a closer view. As advertised, $\mathcal{G}(\tau) \subseteq \mathcal{X}(\tau)$.

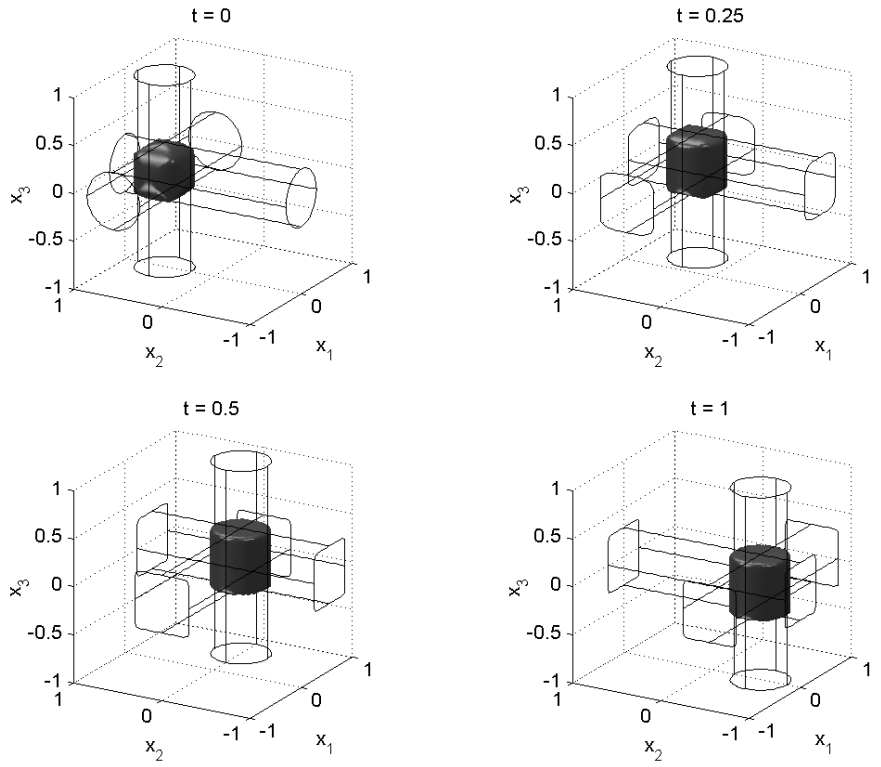

Figure 4.2: Evolution of the linear rotation example's projective overapproximations $\mathcal{Y}_{ij}(\tau)$ (contours on the walls) and $\mathcal{X}(\tau)$ (solid object).

4.2 Solving the Game of Two Identical Vehicles Projectively

The linear rotation was a toy example; in this section we examine the projective overapproximation algorithm's application to the real reachable set problem from section 3.1. We use the indexed state vector notation from this chapter in the analysis, so

$$z = \begin{bmatrix} x_r \\ y_r \\ \psi_r \end{bmatrix} \text{ becomes } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

We use a single projection into the relative location plane $\mathbb{Y}_{12}$. Because the unmodeled dimension—the relative heading x_3—is already restricted to $\mathcal{Y}_3 = [0, 2\pi]$, there is no need

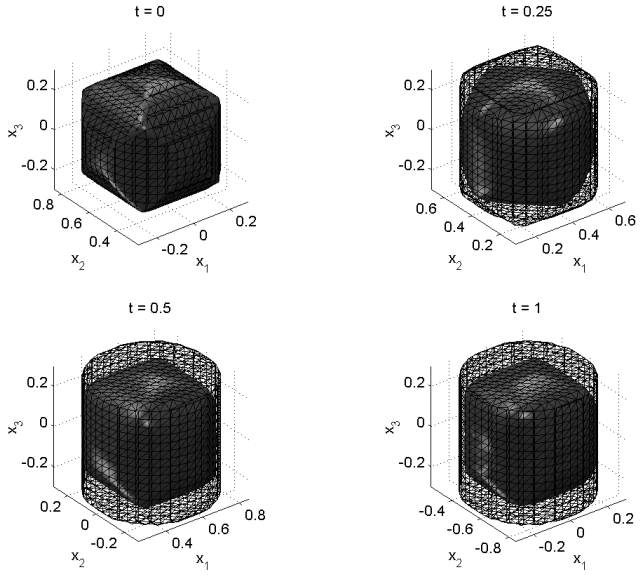

Figure 4.3: Comparing the projection based approximation $\mathcal{X}(\tau)$ (mesh) to the true reachable set $\mathcal{G}(\tau)$ (solid) at several times.

to keep track of any other projections. We simply solve

$$D_t\phi_{12}(x_1, x_2, t) + \min\left[0, H(x_1, x_2, D_{x_1}\phi_{12}(x_1, x_2, t), D_{x_2}\phi_{12}(x_1, x_2, t))\right] = 0,$$

with Hamiltonian

$$H(x_1, x_2, p_1, p_2) = \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} \min_{x_3\in[0,2\pi]} p_1 f_1(x_1, x_2, x_3, a, b) + p_2 f_2(x_1, x_2, x_3, a, b)$$

(where $f(x, a, b)$ is given by (3.1)) and terminal conditions

$$\phi(x_1, x_2, 0) = \sqrt{x_1^2 + x_2^2} - d.$$

The leftmost subplot of figure 4.4 shows the initial collision circle $\mathcal{Y}_{12}(0)$, while the remaining subplots show the growth of $\mathcal{Y}_{12}(\tau)$ until it converges to a fixed point $\mathcal{Y}_{12}$ in the rightmost for $t \gtrsim 2.6$. Figure 4.5 compares the overapproximation of the reachable set $\mathfrak{p}_{12}^{-1}[\mathcal{Y}_{12}]$ to the true reachable set $\mathcal{G}$ from two angles. Although $\mathfrak{p}_{12}^{-1}[\mathcal{Y}_{12}]$ is significantly larger than $\mathcal{G}$, in the left hand view it can be seen that to within grid resolution, $\mathcal{Y}_{12} = \mathfrak{p}_{12}[\mathcal{G}]$, which

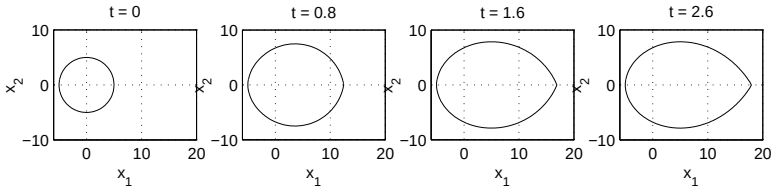

Figure 4.4: Growth of the projective reachable set $\mathcal{Y}_{12}(\tau)$ for the game of two identical vehicles.

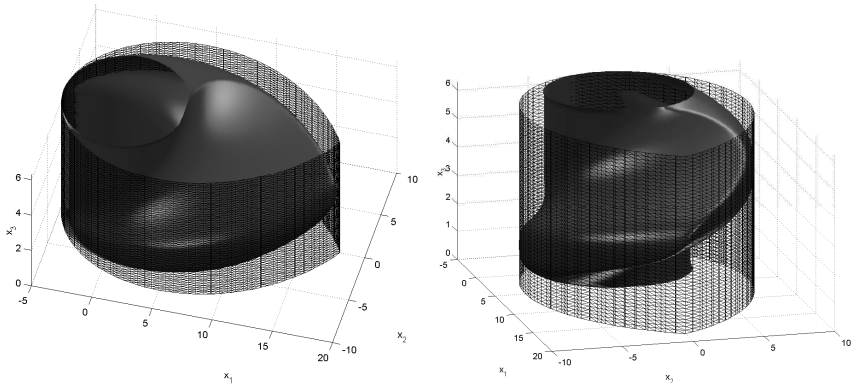

Figure 4.5: Two views comparing the true reachable set $\mathcal{G}$ (solid) with the back projection $\mathfrak{p}_{12}^{-1}[\mathcal{Y}_{12}]$ of the projective reachable set (mesh) for the game of two identical vehicles.

is the best that any projective representation could hope to achieve. The real payoff is computational time. While the full dimensional reachable set $\mathcal{G}$ takes about 20 minutes to compute on a three dimensional grid, the projective overapproximation $\mathcal{Y}_{12}$ takes less than one minute on a higher resolution two dimensional grid.

Figure 4.6 shows a series of frames from an animation of the collision avoidance scenario of section 3.1.5 when the evader uses the projective overapproximation $\mathcal{Y}_{12}$ of the backwards

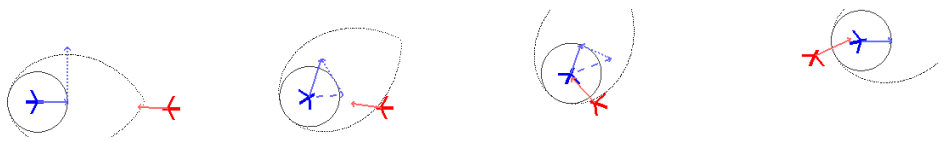

Figure 4.6: Evader keeps pursuer from entering the projective overapproximation $\mathcal{Y}_{12}$ of the reachable set, and hence conservatively avoids collision.

reachable set. When comparing figure 4.6 to figure 3.10, notice that the slice of reachable set in the frames of figure 4.6 does not depend on relative heading, since that is the unmodeled dimension in the projection. By construction, the evader can keep the pursuer from entering $\mathcal{Y}_{12}$, and as long as the pursuer does not enter, a collision is impossible. Using $\mathcal{Y}_{12}$ is a conservative strategy—it is an overapproximation of the true reachable set—but it is guaranteed to be safe and can be recomputed much more quickly than the true reachable set should model parameters change.

4.3 Discussion

While the outline of the projective overapproximation algorithm above was specific to projecting a three dimensional space into coordinate aligned two dimensional subspaces, the power of this HJI based approach is that it can be generalized so easily. Both the full space and the projection subspaces can be higher dimensional. The projection subspaces need not be aligned with the coordinate axes, nor need all subspaces be of the same dimension; in fact, there are systems in which it might be useful to allow the projection subspaces to change smoothly with time. In a projection with multiple unmodeled dimensions, all the unmodeled dimensions are just treated as a disturbance input vector constrained by the appropriate projection of $\mathcal{X}(\tau)$ into the subspace spanned by the unmodeled dimensions. There is no theoretical reason to constrain the number of projections—for example, we could add to the linear rotation problem a projection into the subspace whose coordinate axes are $e_1 + e_3$ and $e_1 - e_3$, if we thought that such a projection would help restrict excessive overapproximation in $\mathcal{X}(\tau)$. The only constraints are implementation complexity and computational resources.

All of this flexibility in the choice of projections leads to the question of how to choose appropriate projections for a particular system. For the linear rotation example, the natural coordinate axis projections turned out to be very effective (see section 4.1.4). In particular, the $\mathbb{Y}_{12}$ projection captured the relevant system dynamics and thus constrained the other two less effective projections through the clipping procedure. We can simulate the effect of poorly chosen projections by using the same three coordinate axis aligned projections,

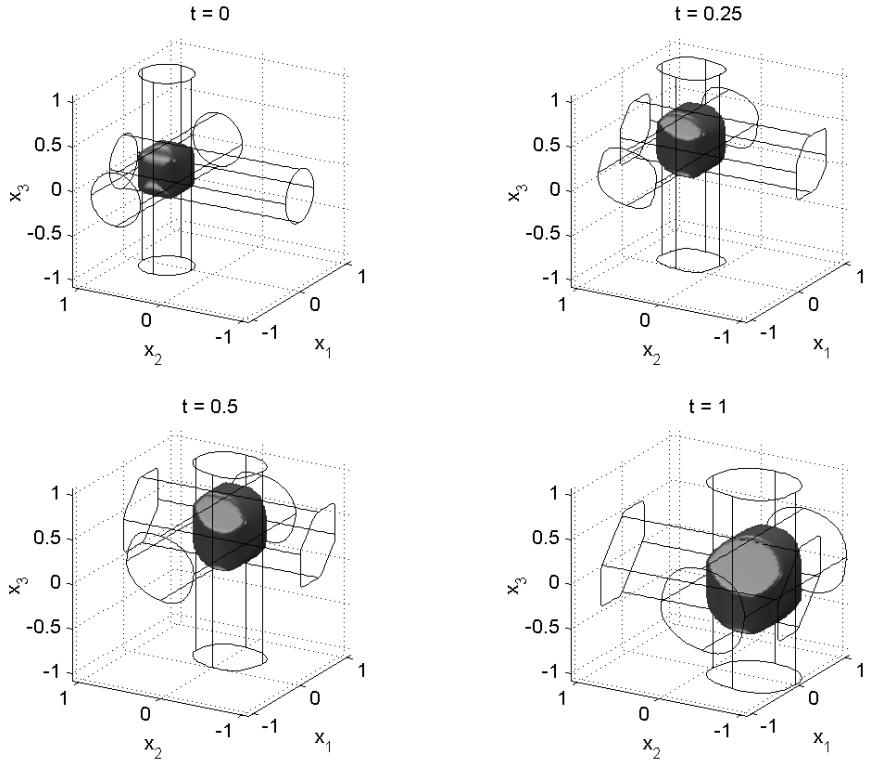

Figure 4.7: Evolution of the linear rotation example with poorly chosen projections.

but rotating the system dynamics counterclockwise by 45° around the e_1 axis. To do that, replace the matrix A in (4.4) by

$$A' = GAG^T,$$

where

$$G = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix},$$

and $\theta = \pi/4$. Figure 4.7 shows the growth of the projective reachable set $\mathcal{X}(\tau)$ for this version of the linear rotation example. Comparing it with figure 4.2 we can see how much greater the overapproximation becomes when none of the projections capture the system's dynamics. We are still investigating techniques for identifying appropriate projections for

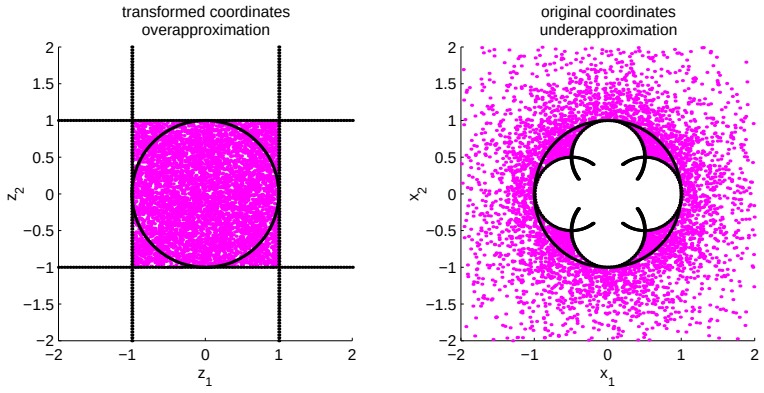

Figure 4.8: One avenue toward projective underapproximations: a square overapproximating the unit circle in z space becomes a clover leaf underapproximating the same circle in x space.

more general systems.

The idea of subspace projections works well when we are trying to overapproximate reachable sets, because inverting these projections back up into the full dimensional space generates a prism overapproximating the true reachable set. There are problems in which we wish to underapproximate a set; for example, in aircraft envelope protection (sections 3.3 and 6.2), safety requires that we stay within the flight envelope. If we are going to approximate that envelope we need an underapproximation, since an overapproximation would incorrectly mark as safe some states outside the true envelope. Safe flight envelopes are just one example of *controlled invariant sets*, and to compute these sets we need underapproximations of the true reachable set.

The projection scheme outlined above cannot directly compute underapproximations, since the back projected prisms are unbounded in the projection's unmodeled dimensions; thus, those prisms could never represent underapproximations of the true reachable set. We are instead investigating a coordinate inversion that could turn overapproximations into underapproximations. Consider underapproximating a circle centered at the origin in $\mathbb{R}^2$ by a pair of one dimensional projections (intervals of $\mathbb{R}$). Map $x \in \mathbb{R}^2$ to $z \in \mathbb{R}^2$ through the transformation

$$z_i = \frac{x_i}{\|x\|_2^2}. \tag{4.15}$$

While the circle stays a circle, this transformation could be applied to more general shapes by transformation of their implicit surface function representation, provided that the coordinate origin did not lie on the boundary of the shape (we could shift the origin if it did). System dynamics can likewise be mapped through this nonlinear transformation, so that reachability could be calculated in the transformed coordinates. Now build a projective overapproximation of the circle in z space, using projections onto the coordinate axes. The left side of figure 4.8 shows the slabs that are the inverse projections of the two overapproximating intervals. The intersection of these two slabs is a square overapproximating the circle. The key observation is that we can invert (4.15) back into the original coordinate frame, and in the process the overapproximation in z space becomes an underapproximation in x space—the square that was an intersection of slabs becomes a clover leaf made from a union of circles. The right side of figure 4.8 shows this underapproximation of the circle. The gray points on the left side map to the gray points on the right, and lie in the region of each state space that would be considered unsafe in an envelope protection problem. If the circle represents the true safe flight envelope, notice that the projective safe region (no gray points) on the left is an underapproximation of the true safe region.

Projective schemes based on Hamilton-Jacobi-Isaacs equations are a powerful way to tackle Bellman's "curse of dimensionality" and calculate approximations to reachable sets for systems larger than dimension two or three. In this chapter we have presented the basic ideas behind projective approximation algorithms. We continue to work on the many remaining theoretical and implementation details.

Chapter 5

Reachable Sets for Hybrid Systems

In a hybrid system, the interaction between continuous and discrete behaviors plays an important role in the evolution of the system's state. In the small time scale limit, there are few macroscopic systems which could not be modeled entirely by continuous differential equations; however, we frequently do not want to invest the effort to do so. The evolution of many systems' states occur on different timescales—some states switch between equilibria quickly while others change much more gradually.

As an example, consider an autonomous robot approaching a wall. Once the obstacle is detected, the robot's microcontroller may decide to turn left after only a few milliseconds, but the response time of the motors which move the robot will be hundreds of times longer. It would be overkill to model the continuous dynamics of the transistors within the microcontroller; on the other hand, modeling the robot's position discretely as either "free" or "collided" would make path planning difficult. Consequently, we would like to model the robot's command state as discrete (either "move straight" or "turn left") but its planar location and heading as continuous variables. The result is a hybrid system.

In previous chapters we have demonstrated how reachable sets can be used for safety verification and the synthesis of safe controllers for purely continuous systems. It should come as no surprise that reachable sets can be put to the same uses in hybrid systems. In addition, because the behavior of these systems can be quite complex, we frequently would like to summarize it in a qualitative discrete form. A reachable set analysis can divide the state space into subsets which facilitate such a summary.

In this chapter we show how an algorithm for continuous reachable sets can be extended to the hybrid domain. The next chapter demonstrates the hybrid reachable set algorithm on several examples. The presentation in this chapter is very informal, although work is under way on bringing these results up to the level of rigor demonstrated in chapter 2. Our focus here is on reachable set determination, but there are many other aspects to the field of hybrid systems. We refer the interested reader to the special issue [7] or the workshop proceedings [136, 26, 82, 140, 72, 5, 6, 3, 8, 65].

5.1 Related Work

Hybrid systems are clearly widespread throughout engineering—any system subject to computer control and interacting with the external world could be classified as hybrid. The interest in creating a formal framework for such systems has brought together researchers in both the traditional control engineering community and the computer science verification community. In a quest for methods to prove that hybrid systems operate correctly, the latter community has generated a number of tools for examining the reachable sets of hybrid systems.

The classification of hybrid systems is usually based on the types of continuous evolution allowed, and to a lesser extent the types of events that can generate discrete transitions. The simplest hybrid systems are the *timed automata*, which basically add continuous timer variables to a finite automata model. In *linear hybrid automata* the continuous variables may evolve according to polyhedral differential inclusions, which translates into allowing timers of different rates. The next step up in complexity allows for linear dynamics in the traditional controls sense; typically $\dot{x} = Ax + Ba$ for some matrices A and B and some input a, but sometimes including a constant term and/or a second, adversarial input. The most general models allow for fully nonlinear dynamics, in some cases with one or even two adversarial input signals.

For all but certain restrictive classes of hybrid systems, finding the reachable set exactly is formally undecidable (for example, see [127]). As a result, all available tools use approximation methods of various types. The algorithm discussed below was motivated by the work of [94, 11] for reachability computation and controller synthesis on timed automata, and

that of [141] for controller synthesis on linear hybrid automata. Early tools for analyzing reachable sets include *Uppaal* [89] and *Kronos* [142] for timed automata and HyTech [69, 70] for linear hybrid automata.

For more general hybrid automata, the most complicated part of the hybrid reachable set determination is the continuous reachable sets; consequently, newer tools for hybrid systems have developed around related continuous reachable set solvers. An overview of the most mature implementations—those being used by people other than their authors—is given in [130]. In addition to Uppaal and HyTech, the paper discusses d/dt [10, 47], and CheckMate [41, 131], which both use polyhedral representations. Other implementations include [25], which studies piecewise affine systems, modifications of the HyTech methods in [71] and [119], and the VeriSHIFT tool [28], which uses ellipsoidal representations. These algorithms are all of the overapproximative, forward reachable set variety. Some can be used for general nonlinear dynamics by employing nonlinear optimization in the continuous reachable set computation.

So far, no tools have emerged for computing hybrid reachable sets using convergent continuous reachable set methods, but research has been undertaken. This thesis describes a procedure that uses the time-dependent Hamilton-Jacobi-Isaacs formulation and level set methods. Theoretical and experimental work has also extended concepts from viability theory to hybrid systems [15, 16].

Finally, Lyapunov theory has inspired some algorithms for linear or piecewise (switched) linear hybrid systems, including [67, 66, 32, 80]. To our knowledge, these have not yet been implemented in generally available tools.

5.2 A Reachable Set Algorithm for Hybrid Systems

A hybrid reachable set algorithm was developed in [135, 93, 138, 92], and we briefly describe the algorithm in this section for background purposes (with a few notational modifications for consistency with the rest of the thesis). A key component of the algorithm is the reach-avoid operator. Section 5.3 describes the accurate numerical implementation of the reach-avoid operator, which constitutes the novel contribution of this thesis to the hybrid algorithm.

5.2.1 Hybrid Automata

A *hybrid automaton* H is defined as

$$H \triangleq ((\mathbb{Q} \times \mathbb{X}), (\mathcal{A} \times \mathcal{B}), (\Sigma_A \times \Sigma_B), f, \delta, \text{Inv}) \tag{5.1}$$

where

- $\mathbb{Q}$ is a finite set of discrete modes,
- $\mathbb{X} = \mathbb{R}^n$ is the continuous state space,
- $\mathcal{A} \subseteq \mathbb{R}^{n_a}$ is the set of continuous inputs for player I,
- $\mathcal{B} \subseteq \mathbb{R}^{n_b}$ is the set of continuous inputs for player II,
- Σ_A is the finite set of discrete inputs for player I,
- Σ_B is the finite set of discrete inputs for player II,
- $f : \mathbb{Q} \times \mathbb{X} \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}^n$ defines the flow of continuous trajectories,
- $\delta : \mathbb{Q} \times \mathbb{X} \times \Sigma_A \times \Sigma_B \to 2^{\mathbb{Q} \times \mathbb{X}}$ is the discrete transition function, which encodes both the guard conditions that enable discrete transitions as well as any resets of the continuous state that take place upon a discrete transition,
- $\text{Inv} \subseteq \mathbb{Q} \times \mathbb{X}$ is the invariant associated to each discrete state.

In order to distinguish between continuous and discrete, we use the term *mode* to refer to a discrete state and *switch* or *transition* to refer to a discrete change of state. We apply Assumption 1 to our continuous input signals $a(\cdot)$ and $b(\cdot)$ and Assumption 2 to the flow field $f(q, \cdot, \cdot, \cdot)$ in each mode $q \in \mathbb{Q}$.

Our target set $\mathcal{G}_0 \subseteq \mathbb{Q} \times \mathbb{X}$ may now include a (possibly different) subset of the state space for each discrete mode. In general, its level set function representation $g : \mathbb{Q} \times \mathbb{X} \to \mathbb{R}$ will likewise depend on mode as well:

$$\mathcal{G}_0 = \{(q, x) \in \mathbb{Q} \times \mathbb{X} | g(q, x) \leq 0\}.$$

We apply Assumption 3 to $\mathcal{G}_0$ and $g(\cdot,\cdot)$. Like the purely continuous case, player I will try to keep the state away from $\mathcal{G}_0$, while player II will drive the state toward $\mathcal{G}_0$. In all of our examples $\mathcal{G}_0$ is the set of unsafe states, so we can consider player I to be the control and player II the disturbance.

Our definition of hybrid automata (5.1) is very general. It allows for nonlinear continuous dynamics, two adversarial inputs which can affect both the continuous and discrete evolution of the system, invariant and guard conditions whose intersection has an interior—discrete switches may be enabled without being forced—and general nonlinear resets of the continuous state after a discrete transition. While Assumption 2 yields deterministic continuous evolution of the state within a particular mode, the general nature of δ and Inv allow for nondeterministic transitions and resets. Our definition also allows for some ill-posed hybrid automata [91]. Informally, a hybrid automaton is *blocking* if there exist states from which no further evolution is possible; for example, a trajectory which is outside a mode's invariant but for which no discrete transition is enabled. A hybrid automaton is *Zeno** if it allows for trajectories containing an infinite number of discrete switches in a finite time. We will restrict ourselves to non-blocking and non-Zeno hybrid automata. Chapter 6 gives some examples of hybrid automata, although by no means do they cover the broad range of behaviors possible under (5.1) for which the algorithm outlined in section 5.2.2 will work.

Because of the nondeterminism allowed by (5.1), trajectories of H can be somewhat tricky to define. Luckily, however, the results presented here require us to study only trajectories within a single discrete mode. For some fixed $q \in \mathbb{Q}$, let a trajectory be denoted by

$$\xi_f(s; q, x, t, a(\cdot), b(\cdot)) : [t, 0] \to \mathbb{X},$$

where for all $s \in [t, 0]$,

$$\begin{aligned}
\frac{d}{ds}\xi_f(s; q, x, t, a(\cdot), b(\cdot)) &= f(q, \xi_f(s; q, x, t, a(\cdot), b(\cdot)), a(s), b(s)) \text{ almost everywhere,} \\
\xi_f(t; q, x, t, a(\cdot), b(\cdot)) &= x, \\
(q, \xi_f(s; q, x, t, a(\cdot), b(\cdot))) &\in \text{Inv}\,.
\end{aligned}$$

The final condition ensures that the trajectory is not forced out of the current mode q during the time span of interest.

*Named after the Greek philosopher Zeno of Elea, famous for posing the paradox of Achilles and the tortoise [91].

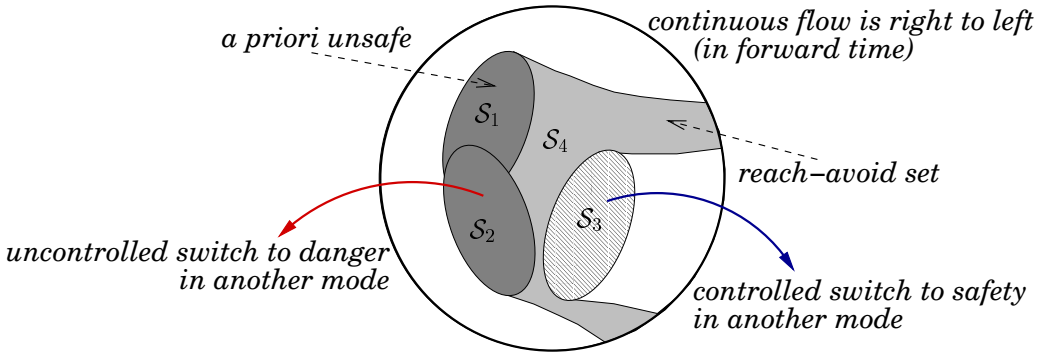

Figure 5.1: Hybrid reachable set algorithm's definitions for a single discrete mode. For iteration i, $\mathrm{Pre}_{\mathrm{B}}(\mathcal{W}^i) = \mathcal{S}_1 \cup \mathcal{S}_2$, $\mathrm{Pre}_{\mathrm{A}}(\mathcal{W}^i) = \mathcal{S}_3$ and $\mathrm{RA}(\mathrm{Pre}_{\mathrm{B}}(\mathcal{W}^i), \mathrm{Pre}_{\mathrm{A}}(\mathcal{W}^i)) = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_4$. In words, states in $\mathcal{S}_1$ are in the known unsafe set $\mathcal{G}_0$, states in $\mathcal{S}_2$ can be forced to make a discrete transition into an unsafe state in another mode, states in $\mathcal{S}_3$ can choose to make a discrete transition into a safe state in another mode, and states in $\mathcal{S}_4$ can reach $\mathcal{S}_1 \cup \mathcal{S}_2$ without going through $\mathcal{S}_3$.

5.2.2 The Algorithm

Define three operators:

$$
\begin{aligned}
\mathrm{Pre}_{\mathrm{A}}(\mathcal{K}) &\triangleq \{(q,x) \in \mathbb{Q} \times \mathbb{X} | \exists \sigma_a \in \Sigma_A,\ \forall \sigma_b \in \Sigma_B,\ \delta(q,x,\sigma_a,\sigma_a) \subseteq \mathcal{K}\} \cap \mathcal{K}, \\
\mathrm{Pre}_{\mathrm{B}}(\mathcal{K}) &\triangleq \{(q,x) \in \mathbb{Q} \times \mathbb{X} | \forall \sigma_a \in \Sigma_A,\ \exists \sigma_b \in \Sigma_B,\ \delta(q,x,\sigma_a,\sigma_b) \cap \mathcal{K}^c \neq \emptyset\} \cup \mathcal{K}^c, \\
\mathrm{RA}(\mathcal{G},\mathcal{E}) &\triangleq \{(q,x) \in \mathbb{Q} \times \mathbb{X} | \forall a(\cdot) \in \mathfrak{A}(t),\ \exists b(\cdot) \in \mathcal{B}(t), \exists t \leq 0 \\
&\qquad \text{such that } (q, \xi_f(0;q,x,a(\cdot),b(\cdot))) \in \mathcal{G} \\
&\qquad \text{and } \forall s \in [t,0],\ (q, \xi_f(s;q,x,a(\cdot),b(\cdot))) \in \mathrm{Inv} \setminus \mathcal{E}\},
\end{aligned}
\tag{5.2}
$$

where $\mathcal{K} \subseteq \mathbb{Q} \times \mathbb{X}$, $\mathcal{G} \subseteq \mathbb{X}$, and $\mathcal{E} \subseteq \mathbb{X}$. The set $\mathrm{RA}(\mathcal{G},\mathcal{E})$ describes those states from which there exists a $b(\cdot) \in \mathfrak{B}(t)$ such that the state trajectory $\xi_f(\cdot;q,x,t,a(\cdot),b(\cdot))$ can be driven into the *reach set* $\mathcal{G}$ while not entering the *avoid set* $\mathcal{E}$ for all $a(\cdot) \in \mathfrak{A}(t)$. We call $\mathrm{RA}(\cdot,\cdot)$ the *reach-avoid* operator. These definitions are illustrated in figure 5.1.

With these definitions in place, the following algorithm computes the hybrid backward

reachable set for a non-blocking, non-Zeno hybrid automaton. Initialize with

$$\begin{aligned} \mathcal{W}^{+1} &= \emptyset, \\ \mathcal{W}^{0} &= \mathcal{G}_0^{\complement}, \\ i &= 0. \end{aligned}$$

Then perform the loop

$$\begin{aligned} &\text{while}(\mathcal{W}^i \neq \mathcal{W}^{i+1}) \\ &\quad \mathcal{W}^{i-1} = \mathcal{W}^i \setminus \mathrm{RA}(\mathrm{Pre}_{\mathrm{B}}(\mathcal{W}^i), \mathrm{Pre}_{\mathrm{A}}(\mathcal{W}^i)), \\ &\quad i = i - 1. \end{aligned}$$

If the algorithm terminates after a finite number of steps, then the fixed point $\mathcal{W}^*$ is the largest set of states for which player I $(a(\cdot), \sigma_a(\cdot))$ can guarantee that the state of the hybrid system remains outside $\mathcal{G}_0$ despite the action of player II $(b(\cdot), \sigma_b(\cdot))$. In order to implement this algorithm, $\mathrm{Pre}_{\mathrm{A}}(\cdot)$, $\mathrm{Pre}_{\mathrm{B}}(\cdot)$, and $\mathrm{RA}(\cdot,\cdot)$ need to be computed. The calculation of $\mathrm{Pre}_{\mathrm{A}}(\cdot)$ and $\mathrm{Pre}_{\mathrm{B}}(\cdot)$ requires inversion of the transition relation δ subject to the existential and universal quantifiers; at present this procedure is performed by hand in our examples, since they all involve relatively simple discrete transition logic. The computation of $\mathrm{RA}(\cdot,\cdot)$ requires an algorithm for determining the set of initial conditions from which trajectories can reach one set avoiding a second set along the way.

5.3 Implementing the Reach-Avoid Operator

Although (5.2) defined the reach-avoid operator over the entire set of discrete modes, in practice it is computed one mode at a time. In the remainder of this section, we focus on a single mode $q \in \mathbb{Q}$, but we omit this parameter from the equations in order to simplify them; for example, we use $f(\cdot,\cdot,\cdot)$ to refer to the flow field $f(q,\cdot,\cdot,\cdot)$ for mode q. We also generically refer to the parameters of the reach-avoid operator as $\mathcal{G}$ and $\mathcal{E}$ rather than $\mathrm{Pre}_{\mathrm{B}}(\cdot)$ and $\mathrm{Pre}_{\mathrm{A}}(\cdot)$.

5.3.1 Previous Reach-Avoid Formulations

The first Hamilton-Jacobi-Isaacs based formulation for the reach-avoid operator [135, 138] featured two separate level set functions, one for the reach set ϕ_G and one for the avoid set ϕ_E. The functions were evolved according to the coupled terminal value HJI PDEs

$$\begin{aligned} -D_t\phi_G(x,t) &= \begin{cases} H_G(x, D_x\phi_G(x,t)), & \text{for } \{x \in \mathbb{X} | \phi_G(x,t) > 0\}; \\ \min\left[0, H_G(x, D_x\phi_G(x,t))\right], & \text{for } \{x \in \mathbb{X} | \phi_G(x,t) \leq 0\}; \end{cases} \\ -D_t\phi_E(x,t) &= \begin{cases} H_E(x, D_x\phi_E(x,t)), & \text{for } \{x \in \mathbb{X} | \phi_E(x,t) > 0\}; \\ \min\left[0, H_E(x, D_x\phi_E(x,t))\right], & \text{for } \{x \in \mathbb{X} | \phi_E(x,t) \leq 0\}; \end{cases} \end{aligned} \tag{5.3}$$

where $\phi_G(x,0)$ and $\phi_E(x,0)$ were level set representations of the input sets $\mathcal{G}$ and $\mathcal{E}$ of $\mathrm{RA}(\mathcal{G},\mathcal{E})$ and

$$\begin{aligned} H_G(x,p) &= \begin{cases} 0, & \text{for } \{x \in \mathbb{X} | \phi_E(x,t) \leq 0\}; \\ \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} p^T f(x,a,b), & \text{otherwise}; \end{cases} \\ H_E(x,p) &= \begin{cases} 0, & \text{for } \{x \in \mathbb{X} | \phi_G(x,t) \leq 0\}; \\ \min_{a\in\mathcal{A}} \max_{b\in\mathcal{B}} p^T f(x,a,b), & \text{otherwise}. \end{cases} \end{aligned} \tag{5.4}$$

Note that in the evolution of the avoid set, player I is trying to drive the state into $\mathcal{E}$ and player II is trying to keep it out; therefore, the role of the inputs in the maximization and minimization in H_E is swapped from the role they play in H_G (and in our continuous reachable set formulation's Hamiltonian (2.8)). From (5.3) and (5.4) we can see that the evolution of ϕ_G or ϕ_E is frozen when the state is either in the reach or in the avoid set already. The value of the reach-avoid operator in this formulation is given by

$$\mathrm{RA}(\mathcal{G},\mathcal{E}) = \left\{ x \in \mathbb{X} \,\middle|\, \lim_{t\to\infty} \phi_G(x,t) \leq 0 \right\}.$$

While this formulation may work to determine the reach and avoid sets in theory, its practical implementation is complicated by the multiple case structure of (5.3) and (5.4). Because the right hand side of (5.3) could be drastically different for two states arbitrarily close together, the functions ϕ_G and ϕ_E could develop discontinuities. In the presence of

discontinuities, numerical level set methods and, in fact, the basic viscosity solution theory of Hamilton-Jacobi equations breaks down.

To avoid these discontinuities, a second formulation was developed in [102] based on the HJI PDEs

$$\begin{aligned} D_t\phi_G(x,t) + H_G(x, D_x\phi_G(x,t)) &= 0, \\ D_t\phi_E(x,t) + H_E(x, D_x\phi_E(x,t)) &= 0, \end{aligned} \tag{5.5}$$

where $\phi_G(x,0)$ is a level set function for $\mathcal{G}$, $\phi_E(x,0)$ is a level set function for $\mathcal{E}$, and

$$\begin{aligned} H_G(x,p) &= \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} p^T f(x,a,b), \\ H_E(x,p) &= \min_{a\in\mathcal{A}} \max_{b\in\mathcal{B}} p^T f(x,a,b). \end{aligned} \tag{5.6}$$

Letting

$$\begin{aligned} \phi_G^{\min}(x,t) &= \min_{s\in[t,0]} \phi_G(x,s), \\ \phi_E^{\min}(x,t) &= \min_{s\in[t,0]} \phi_E(x,s), \end{aligned} \tag{5.7}$$

we put an additional constraint on the solutions ϕ_G and ϕ_E of (5.5)

$$\begin{aligned} \phi_G(x,t) &\geq -\phi_E^{\min}(x,t), \\ \phi_E(x,t) &\geq -\phi_G^{\min}(x,t). \end{aligned} \tag{5.8}$$

The value of the reach-avoid operator is

$$\mathrm{RA}(\mathcal{G},\mathcal{E}) = \left\{ x \in \mathbb{X} \,\middle|\, \lim_{t\to\infty} \phi_G^{\min}(x,t) \leq 0 \right\}.$$

We assumed that the purely continuous reachable set could be described by (remember that $\tau = -t$)

$$\mathcal{G}(\tau) = \{x \in \mathbb{X} | \phi^{\min}(x,t) \leq 0\},$$

where

$$\begin{aligned} D_t\phi(x,t) + H(x, D_x\phi(x,t)) &= 0, \\ H(x,p) &= \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} p^T f(x,a,b), \\ \phi^{\min}(x,t) &= \min_{s\in[t,0]} \phi(x,s). \end{aligned} \tag{5.9}$$

Under this assumption, it is possible to show that the two formulations (5.3)–(5.4) and (5.5)–(5.8) generate equivalent reach-avoid sets. Unfortunately, (5.9) fails to correctly compute the continuous backwards reachable set in some cases with nonconvex target sets and/or Hamiltonians.

5.3.2 The Current Reach-Avoid Formulation

Our current reach-avoid implementation makes use of the provably correct HJI formulation given in section 2.1.3 for the continuous backwards reachable set. The evolution equation for the level set representation of the reach set ϕ_G is

$$D_t\phi_G(x,t) + \min[0, H(x, D_x\phi_G(x,t))] = 0, \tag{5.10}$$

where the Hamiltonian is

$$H(x,p) = \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} p^T f(x,a,b). \tag{5.11}$$

The level set function for the avoid set is not evolved

$$\phi_E(x) \triangleq \phi_E(x,0), \tag{5.12}$$

but the evolution of the reach set is constrained by

$$\phi_G(x,t) \geq -\phi_E(x). \tag{5.13}$$

This formulation (5.10)–(5.13) turns out to be equivalent to a variational inequality [19], and from that theory we can show that $\phi_G(x,t)$ is bounded and continuous if $\phi_E(x)$ is bounded and continuous. This formulation is considerably simpler than the previous two, in part

because there is no attempt to evolve the avoid set. Section 5.3.3 presents a simplistic argument that avoid set evolution is unnecessary. We are working on a more robust general proof that this formulation implements the reach-avoid operator.

In practical terms, we construct $\phi_G(x, 0)$ and $\phi_E(x)$ using the union, intersection and complement set operations (2.15) for level set functions. We solve (5.10) using the level set methods described in section 2.2.2. The constraint (5.13) is enforced at each grid node at each timestep, a process frequently called *masking* the level set [125].

5.3.3 No Evolution for the Avoid Set

In this section we argue that it is unnecessary to evolve the avoid set as part of the reach-avoid calculation; only the initial avoid set $\mathcal{E}(0) \triangleq \mathcal{E}$ is needed for masking purposes. The concern is that if $\mathcal{E}(\tau)$ is not propagated, points in the state space may be swallowed by $\mathcal{G}(\tau)$ when $\mathcal{E}(\tau)$ should have reached them first and thus made them unavailable to $\mathcal{G}(\tau)$. We show that if a point is part of $\mathcal{E}(\tau)$, then it is either part of $\mathcal{E}(0)$ or may never be part of $\mathcal{G}(\tau)$. The weakness in this argument is the assumption that there exist unique normals for the boundaries of $\mathcal{G}(\tau)$ and $\mathcal{E}(\tau)$; because these boundaries need not be smooth, unique normals need not exist. We are currently working on a more general proof.

Start by assuming that we propagate both $\mathcal{E}(\tau)$ and $\mathcal{G}(\tau)$. Consider a point $x \in \partial\mathcal{G}(\tau) \cap \partial\mathcal{E}(\tau)$ such that $\partial\mathcal{E}(\tau)$ reached x at the same time or before $\partial\mathcal{G}(\tau)$. Remember that $\mathcal{G}(\tau)$ is open and $\mathcal{E}(\tau)$ is closed. Let n_E and n_G be the outward pointing normals at x to $\mathcal{E}(\tau)$ and $\mathcal{G}(\tau)$ respectively. Since the two sets are touching at x, $n_G = -n_E$.

We are concerned that if $\mathcal{E}(\tau)$ were not computed, then $\mathcal{G}(\tau)$ would be able to swallow x. If $x \notin \mathcal{E}(0)$ then by the construction of $\mathcal{E}(\tau)$

$$x \in \mathcal{E}(\tau) \implies \exists a \, \forall b, \; n_E \cdot f(x, a, b) \leq 0. \tag{5.14}$$

Furthermore, for some $\hat{x} \notin \mathcal{G}(0)$, we know that

$$\hat{x} \in \mathcal{G}(\tau) \implies \forall a \, \exists b, \; n_G \cdot f(\hat{x}, a, b) < 0. \tag{5.15}$$

What would happen if $\mathcal{E}(\tau)$ were not computed? Since we know $x \in \mathcal{E}(\tau)$, by (5.14)

$$\begin{aligned}
&\exists a\, \forall b,\, n_E \cdot f(x,a,b) \leq 0,\\
&\exists a\, \forall b,\, (-n_G) \cdot f(x,a,b) \leq 0,\\
&\exists a\, \forall b,\, n_G \cdot f(x,a,b) \geq 0,\\
&\exists a\, \forall b,\, \neg(n_G \cdot f(x,a,b) < 0),\\
\exists a\, &\neg(\exists b,\, n_G \cdot f(x,a,b) < 0),\\
\neg(\forall a\, &\exists b,\, n_G \cdot f(x,a,b) < 0),
\end{aligned}$$

which by the contrapositive of (5.15)

$$\neg(\forall a\ \exists b,\ n_G \cdot f(x,a,b) < 0) \implies x \notin \mathcal{G}(\tau).$$

Therefore, x cannot be swallowed by $\mathcal{G}(\tau)$ even if $\mathcal{E}(\tau)$ were not computed to act as a mask. For points $x \in \mathcal{E}(\tau) \backslash \partial\mathcal{E}(\tau)$, consider the same procedure using $\mathcal{E}(\tau_1)$ and $\mathcal{G}(\tau_2)$ where times $t_1 \leq t_2$ are chosen so that $x \in \partial\mathcal{G}(\tau_2) \cap \partial\mathcal{E}(\tau_1)$.

Note that for $x \in \mathcal{E}(0)$, none of this analysis applies; consequently, $\mathcal{E}(0)$ must still be used as a mask to halt the growth of $\mathcal{G}(\tau)$. Using such a fixed mask is much less of a concern numerically, however.

Chapter 6

Examples of Hybrid Reachable Sets

The first section analyzes the safety regions of some simple collision avoidance protocols for two cooperating vehicles. The primary purpose of the three mode scenario is a step by step demonstration of the hybrid reachable set algorithm and the reach-avoid operator. The seven mode scenario is a little more realistic, and also shows how the results of a reachability analysis can be used to create a discrete abstraction of the hybrid system. The second section looks at a landing aircraft example involving discrete decisions on how to set the flaps.

6.1 Multimode Collision Avoidance

Free flight is a concept under consideration as a way of increasing the efficiency and reducing the flight path congestion for aircraft cruising at high altitudes [120]. At the present time, pilots are usually instructed by air traffic control (ATC) to follow predefined "highways in the sky", and therefore follow a crooked route to their destination [104]. Under free flight, pilots would be allowed to choose their cruising altitude flight paths almost arbitrarily in order to maximize fuel efficiency or minimize flight time. Of course, such freedom in aircraft trajectories would make it difficult for ATC to intervene and resolve the potential collision should two flight paths intersect.

With new technologies allowing aircraft to determine each other's location and speed [117, 76, 81, 57], it becomes possible for the vehicles involved to resolve the conflict independently. Even in this situation, however, it is necessary to define a clear conflict resolution procedure in advance—given the slow response time of a commercial passenger aircraft, it is important to avoid the case where both vehicles turn one way and maintain the conflict, realize their mistake, then both turn the other way and maintain the conflict again, all while approaching one another at more than one thousand miles per hour.

If we are to define collision avoidance protocols, we must identify and describe the situations under which they can be applied safely; reachable sets can be used to perform both tasks. The examples below investigate two simple protocols with the tools developed in chapters 2 and 5. The description of both scenarios is taken from [138], and they were both analyzed in [137]; the three mode version also appeared in [102].

6.1.1 The Model

Since we are again studying a collision avoidance problem, we use relative coordinates with dynamics (3.1) and a circular unsafe set centered at the planar origin representing a collision. While the dynamics may seem simplistic, the resulting trajectories are intuitively easy to understand. In [58], a controller that can track the inertial trajectories generated by (3.1) was derived for an accurate nonlinear model of the lateral flight dynamics of a commercial passenger jet.

In the scenarios below the two vehicles are cooperating, so we assume that their continuous behavior is known and constant, and that their mode switches are synchronized. Removing these assumptions (apart from the deterministic dynamics) would increase the complexity of the computation and visualization, but would not change the analysis procedure.

The aircraft will always be flying in one of two modes:

- Straight flight: both aircraft are flying at constant linear velocities and zero angular velocities. The dynamics are

$$\dot{z} = \frac{d}{dt}\begin{bmatrix} x_r \\ y_r \\ \psi_r \end{bmatrix} = \begin{bmatrix} -v_a + v_b \cos\psi_r \\ v_a \sin\psi_r \\ 0 \end{bmatrix} = f_s(z), \tag{6.1}$$

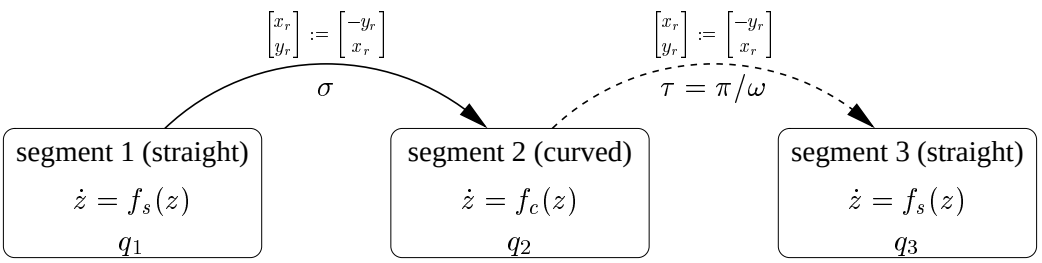

Figure 6.1: Hybrid automaton for the three mode protocol.

where v_a and v_b are fixed (although not necessarily equal).

- Curved flight: both aircraft are flying at constant linear velocities and a constant, equal angular velocity. The dynamics are

$$\dot{z} = \frac{d}{dt}\begin{bmatrix} x_r \\ y_r \\ \psi_r \end{bmatrix} = \begin{bmatrix} -v_a + v_b \cos\psi_r + \omega y_r \\ v_a \sin\psi_r - \omega x_r \\ 0 \end{bmatrix} = f_c(z), \tag{6.2}$$

where v_a, v_b and ω are fixed.

Note that in both modes the relative heading of the aircraft is fixed by the initial conditions, since $\dot{\psi}_r = 0$. Consequently, we can perform the analyses in two dimensions (x_r and y_r) with fixed ψ_r.

The protocols we examine consist of a sequence of straight and curved flight segments, and therefore the system has no continuous inputs. Instead, we will look at a single discrete input σ which initiates the protocol by causing the first mode switch. Subsequent mode switches are timed, and the two aircraft are assumed to switch modes synchronously. Using a hybrid reachable set analysis, we can identify the set of states where no collision will occur, the set of states where the collision avoidance protocol can safely be initiated, the set of states where initiating the protocol will cause a collision, and the set of states where a collision is inevitable whether that particular protocol is invoked or not.

6.1.2 Three Mode Scenario

With this simple scenario we illustrate the hybrid reachable set algorithm. The protocol is outlined in figure 6.1 and demonstrated in figure 6.2, where the continuous dynamics $f_s(z)$

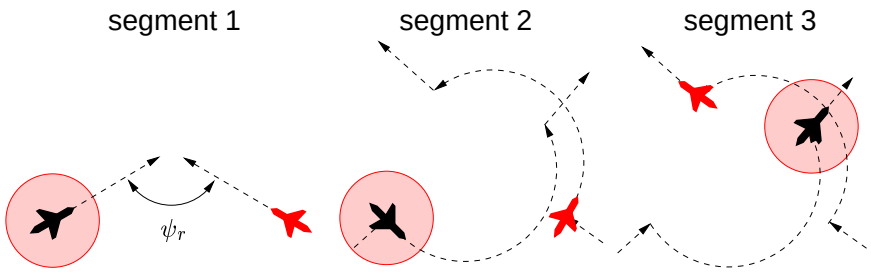

Figure 6.2: Example aircraft trajectories in the three mode protocol.

variable	meaning
x_r	relative position in flight direction of fixed aircraft
y_r	relative position perpendicular to flight direction of fixed aircraft
z	state vector ($z = \begin{bmatrix} x_r & y_r \end{bmatrix}^T$)
ψ_r	constant relative heading ($\psi_r = 120°$)
ω	angular velocity of both aircraft in segment 2 ($\omega = 1$)
v_a	speed of fixed aircraft ($v_a = 3$)
v_b	speed of second aircraft ($v_b = 4$)
d	minimum safe separation distance ($d = 5$)

Table 6.1: Parameters for the three mode protocol.

and $f_c(z)$ are given by (6.1) and (6.2) respectively. The first mode transition is caused by the discrete input σ, while the second is timed and occurs when the aircraft have completed a half circle (in this case, at $\tau = \pi$). Both planes perform an (unrealistic) instantaneous 90° clockwise turn at both mode switches, shown by the reset relations on top of the transition arrows in figure 6.1. The combination of resets and mode q_2 result in the aircraft returning to their original flight direction in the final mode.

The iterations of the hybrid reachable set algorithm are shown in figure 6.3, using the parameters in table 6.1. The *fixed* aircraft is the one lying at the origin of the relative coordinate system. Reach sets are shown in dark grey and avoid sets in light grey. The initial conditions are just the collision set in each mode. Only the analysis of segment one includes an avoid set, because only q_1 has a outward controlled transition to introduce an avoid set. That avoid set is related to the complement of the reach set of q_2 (the subsequent mode for the outward transition) in the previous iteration. The outward transition for q_2 is timed, so it merely adds to the initial reach set for this mode (as demonstrated by the

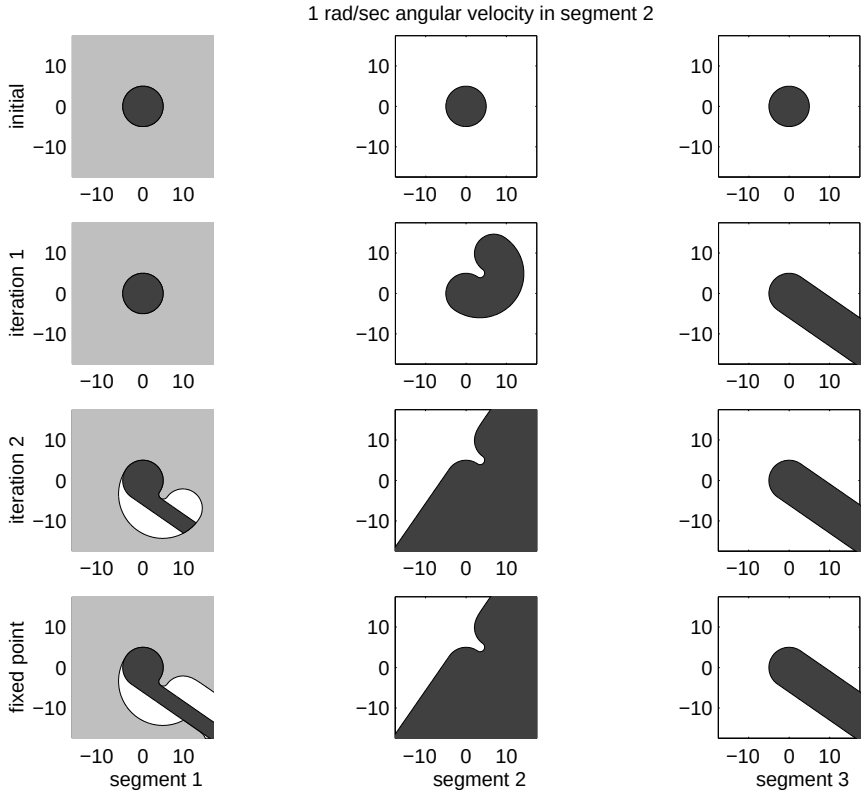

Figure 6.3: Iterations of the hybrid reachable set analysis for the three mode protocol (reach sets in dark grey, avoid sets in light grey).

difference between the reach sets for iterations one and two of segment two). Because the transition out of q_2 is timed, the reach set shown is a projection onto the state space plane of the true three dimensional reach set (the third dimension is an unmodeled timer). There is no outward transition for q_3, so the analysis for segment three is just a standard continuous reachable set for the straight flight dynamics.

After three iterations, the algorithm reaches the fixed point shown in the final row of figure 6.3. The subplot in the lower left corner summarizes the information we can deduce from this analysis whether the collision avoidance protocol should be invoked. If the current state of the system lies in a dark region of this subplot, a collision will occur regardless of whether the protocol is invoked or not. In the white regions, a collision will occur if the protocol is invoked, but the aircraft can safely continue to fly straight without a collision. In the light grey regions, the protocol can be safely invoked.

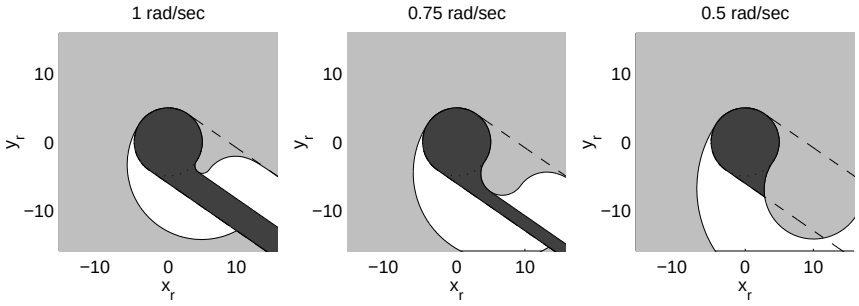

Figure 6.4: Comparing hybrid reachable set analysis results for three different angular velocities ω in the second segment q_2.

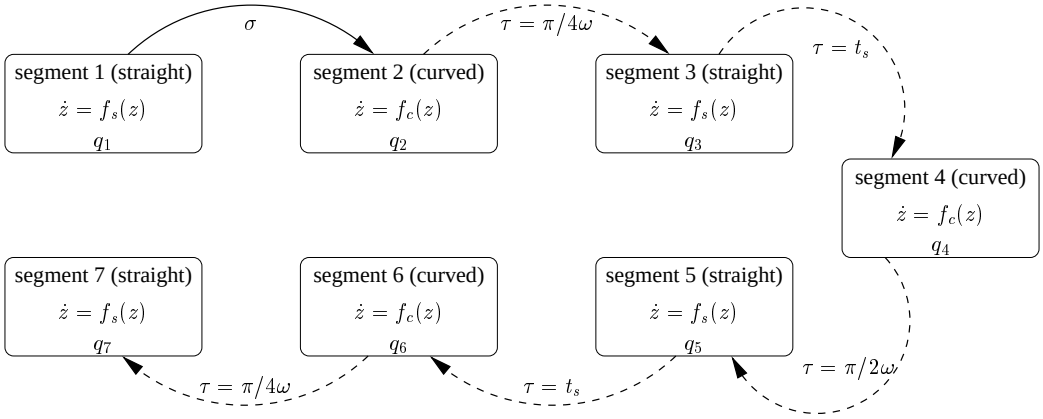

Figure 6.5: Hybrid automaton for the seven mode protocol.

Notice the white region on the right side of this subplot. In this region invoking the protocol will cause a collision, but continued straight flight will lead the system up and to the left into the light grey notch. The protocol can now be initiated, and must be initiated before the system continues into the collision set if safety is to be maintained. Figure 6.4 demonstrates that this notch of safety can be enlarged by increasing the turn radius (reducing ω) in segment two. If it is enlarged sufficiently, the unsafe region of infinite extent (leading down and to the right from the collision set) can be removed.

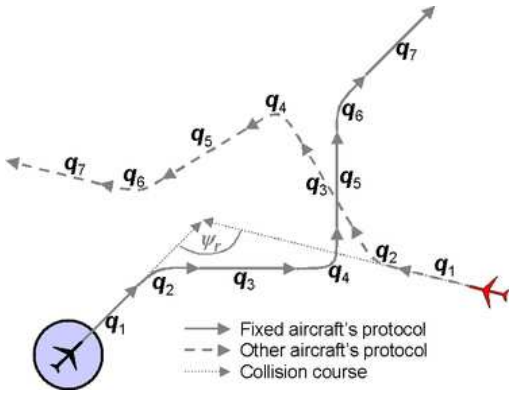

Figure 6.6: Example aircraft trajectories in the seven mode protocol.

variable	meaning
x_r	relative position in flight direction of fixed aircraft
y_r	relative position perpendicular to flight direction of fixed aircraft
z	state vector ($z = \begin{bmatrix} x_r & y_r \end{bmatrix}^T$)
ψ_r	constant relative heading ($\psi_r = 120°$)
ω	angular velocity of both aircraft in segments 2, 4 and 6
t_s	length of time in segments 3 and 5
v_a	speed of fixed aircraft ($v_a = 3$)
v_b	speed of second aircraft ($v_b = 3$)
d	minimum safe separation distance ($d = 5$)

Table 6.2: Parameters for the seven mode protocol.

6.1.3 Seven Mode Scenario

The three mode protocol featured instantaneous heading changes before and after segment two, in order for the aircraft to return to their original heading at the end of the protocol. The hybrid automaton in figure 6.5 describes a more realistic protocol involving seven flight segments, where the continuous dynamics $f_s(z)$ and $f_c(z)$ are given by (6.1) and (6.2) respectively. The protocol's flight path sequence is demonstrated in figure 6.6, which shows that the aircraft return to their original heading at the conclusion of the protocol. There is no need to make instantaneous heading changes, because modes q_2 and q_6 each involve a 45° clockwise turn, which is cancelled by a 90° counter clockwise turn in mode q_4.

We perform an analysis of the hybrid reachable set for this automaton, in the same manner

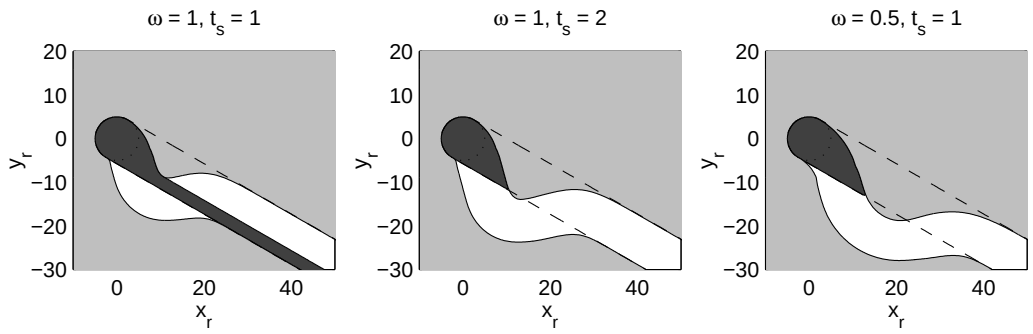

Figure 6.7: Reachable set analysis for the seven mode protocol, comparing three different parameter choices.

as that performed in the previous section and using the parameters in table 6.2. Observing that the set of discrete transitions in figure 6.5 forms a directed acyclic graph, we run the mode iteration backwards through this graph and achieve convergence after one pass.* All transitions in the protocol are timed except the first, so the reach sets of modes q_2 through q_7 are straightforward to calculate, but not particularly enlightening to visualize.

The real information from the analysis is the reach and avoid sets of mode q_1, which are shown in figure 6.7 for three different choices of protocol parameters ω and t_s. The dashed lines are the boundary of the unsafe set if the protocol is not invoked. By choosing parameters wisely, the unsafe set can be shrunk until it is only slightly larger than the original collision set.

Figure 6.8 shows how the information from the hybrid reachable set analysis can be used to divide the continuous state space into subsets, and then to construct a discrete automaton on the subsets which describes the qualitative behavior of the full hybrid automaton. The discrete automaton has two types of transitions: those that can be taken by invoking the discrete input σ, and those that will occur automatically after some time passes. For most states, an appropriate combination of these transitions will lead to safety. For example, if the system starts in set s_1, then invoking σ would be dangerous but waiting will lead to set s_3. In set s_3, waiting is dangerous but invoking σ will lead to safety.

*The same observation applies to the three mode example, but we ran the modes forward in figure 6.3 to demonstrate the general iterative algorithm.

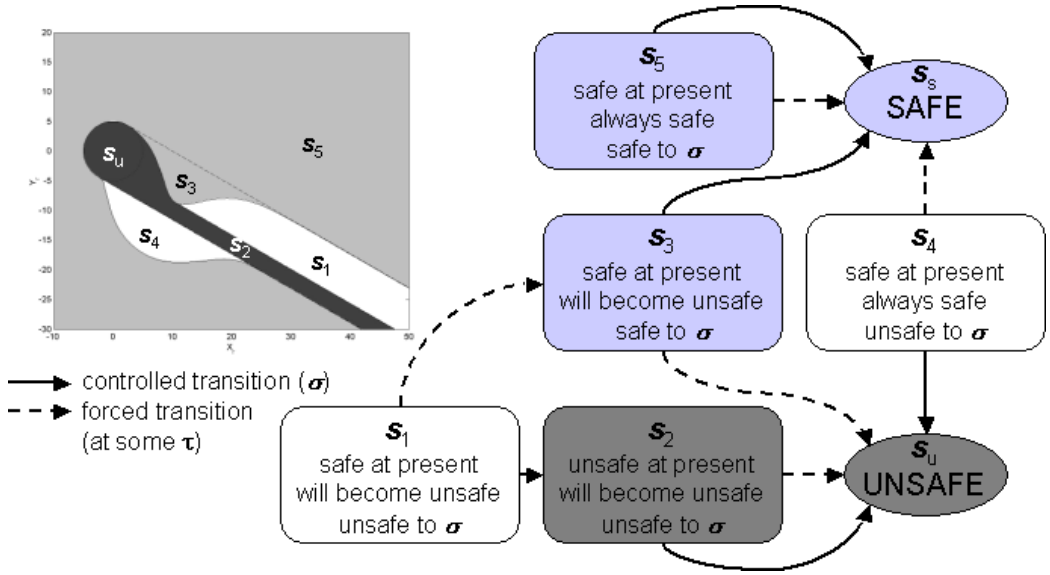

Figure 6.8: Abstracting the seven mode hybrid automaton into a discrete automaton.

6.2 Flap Deflection in a Landing Aircraft

In this section we again examine a landing aircraft, but this time we focus our attention on the flap setting choices available to the pilot. While flap extension and retraction are continuous operations at the lowest level (they are generally actuated by an electric screw drive in the wing), we choose to model their setting as a discrete variable for two reasons. First, in modern commercial airliners the choice of flap deflections given to the pilot is discrete, and is chosen either by pressing a button or moving a lever into one of several fixed settings; a low level automatic controller handles the continuous actuation. Second, the dynamic effect of deflecting flaps is generally assumed to be relatively minor, so we believe that a quasi-static analysis will not adversely affect our fairly simple model. The results in this section are taken from [100].

We use the same continuous model as that described in section 3.3.1, with the parameters modified to fit a DC9-30 landing at sea level. Instead of (3.7), we use lift and drag terms given by (measured in newtons)

$$\begin{aligned} L(\alpha, V) &= 68.6(h_\delta + 4.2\alpha)V^2, \\ D(\alpha, V) &= (2.81 + 3.09(h_\delta + 4.2\alpha)^2)V^2, \end{aligned}$$

flaps (δ)	h_δ	V (m/s) min	V (m/s) max	γ (degrees) min	γ (degrees) max	z (m) min
0u	0.2	78	82	-3	0	0
25d	0.8	61	82	-3	0	0
50d	1.2	58	82	-3	0	0

Table 6.3: Lift parameter and safe flight envelope bounds for various flap settings.

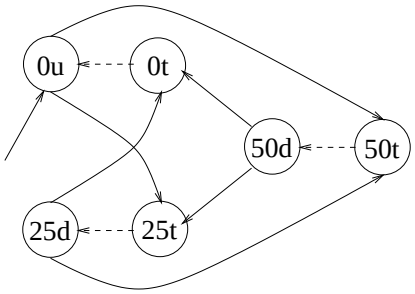

Figure 6.9: Hybrid automaton for the multimode landing aircraft (solid arrows are controlled switches, dashed arrows are timed switches).

We consider three different flap settings denoted by δ: clean wing $\delta = 0$u, partially deflected $\delta = 25$d, and fully deflected $\delta = 50$d. The letter "u" or "d" denotes whether the *slats* on the front edge of the wing are deflected or not—slat deflection increases the maximum angle of α possible without stalling the aircraft, and so slats will always be deflected in any high lift (flap deflected) wing configuration.

The lift parameter h_δ and the safe flight envelopes for each flap setting are summarized in table 6.3. We are not considering a TOGA maneuver in this analysis, so the aircraft may never climb ($\gamma \leq 0$). Upon touchdown ($z = 0$), there is no tail strike restriction on θ, but $\dot{z} \geq -0.9144$ m/s is required to avoid landing gear damage. The continuous inputs are

$$\begin{aligned} T &\in [0 \text{ kN}, 160 \text{ kN}], \\ \alpha &\in [0^\circ, 10^\circ], \end{aligned} \tag{6.3}$$

where we have placed a tighter bound on α for passenger comfort.

The hybrid automaton that we study is shown in figure 6.9. Ideally, we would like to model just the three flap modes 0u, 25d and 50d, but connecting these modes together with

controllable switches leads to a Zeno automaton—the model can switch modes infinitely fast, and hence can maintain safety by stopping the progression of continuous time in an infinitely long switching sequence [79, 78]. To avoid this behavior we add timed delay modes 0t, 25t and 50t. In the resulting automaton, the pilot can switch to a new flap setting instantaneously, but cannot switch out of that setting for some finite time period.

The results of the hybrid reachable set analysis are shown in figure 6.10. Note that we include some states $z \leq 0$ (below ground) in order to ensure that the envelope restriction at $z = 0$ is satisfied. For these states we use dynamics $\dot{x} = 0$.

The top row of the figure shows the initial envelopes from table 6.3. The second row shows the maximally controllable subset of the envelope for each mode individually, as determined by a continuous reachable set computation. The clean wing configuration 0u becomes completely uncontrollable (the remaining stub in the figure lies in the dummy states below ground), while the remaining modes are partially controllable. The subset of the envelope that cannot be controlled in these high lift / high drag configurations can be divided into two components. For low speeds, the aircraft will tend to stall. For values of z near zero and low flight path angles γ, the aircraft cannot pull up in time to avoid landing gear damage at touchdown.

The third row shows the results for the hybrid reachable set computation. Now both modes 0u and 25d are almost completely controllable, since they can switch instantaneously to the fully deflected mode 50d. However, no mode can control the states z near zero and low γ, because no mode can pull up in time to avoid landing gear damage. The fourth row shows a slice through the reach and avoid sets for the hybrid analysis at a fixed height $z = 3$ m. In this case, the reach set is light grey and the avoid set dark grey (the opposite of the convention in this chapter's previous figures).

Since it allows instantaneous flap deflections and changes to continuous inputs T and α, this model is not very realistic; consequently, we do not advocate that these results be applied to aircraft autolander design in their current form. While we are in the process of increasing the fidelity of the model [23], we discuss these results here because they demonstrate a reachable set analysis procedure that we need not change to incorporate a more realistic and complex model.

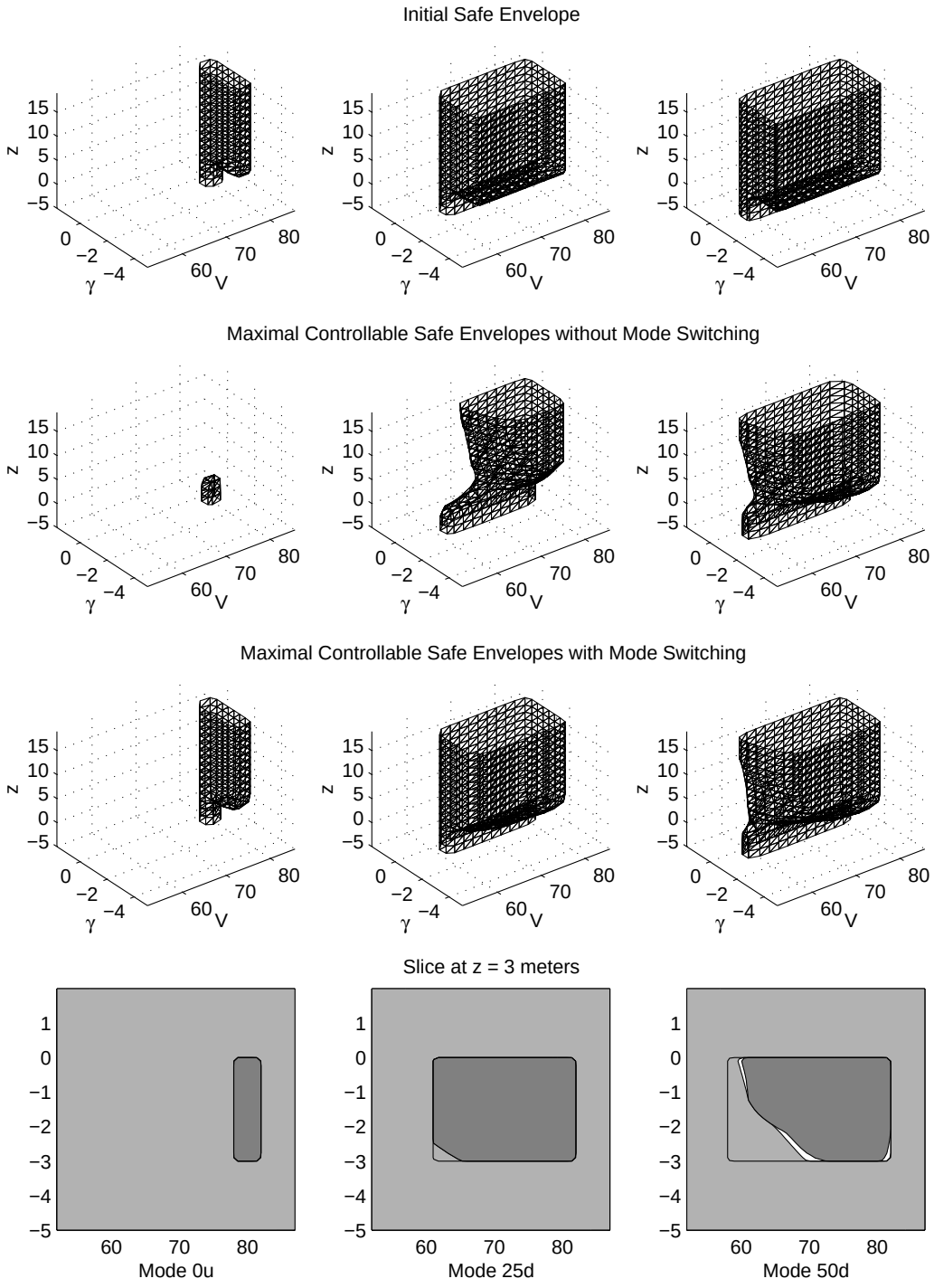

Figure 6.10: Maximally controllable envelopes for the multimode landing example.

Chapter 7

Conclusions

We summarize the results of the previous chapters, make some observations regarding them, and then discuss some possible extensions of this research.

7.1 What Has Been Accomplished

We have presented an algorithm which can numerically compute the backwards reachable set for a two player, continuous nonlinear differential game with a general target set. The algorithm is based on a formulation of reachability in terms of the viscosity solution of a Hamilton-Jacobi-Isaacs PDE, and we have proven that the analytic solution of this equation is the exact backwards reachable set. By adopting this two input HJI formulation, we can conservatively handle model uncertainties as well as traditional control inputs, and we can develop very accurate approximations of the boundary of the reachable set for systems with complex dynamics.

Our implementation is based on level set methods, and its accuracy and convergence have been demonstrated on a three dimensional pursuit-evasion example. This example also served to illustrate the use of reachable sets in synthesizing controllers which are guaranteed to act safely. A second example used a reachable set analysis to detect potential inconsistencies in the protocol followed by a pilot during an autolanding in a commercial passenger aircraft, in the case where the pilot must execute a take-off / go-around.

The primary shortcoming of the HJI formulation is that its cost scales exponentially with the dimension of the system. We described how the idea of projective overapproximations can fit into the two input HJI formulation of reachable sets. Because these projections occupy lower dimensional subspaces, they are much less costly to compute. The dimensions missing from a particular subspace are treated as disturbance inputs to the evolution dynamics for that projection, which ensures that the projection is an overapproximation of the true reachable set. Because the reachable set is the unsafe set, overapproximations can be used to conservatively verify safety.

After presenting a previously developed algorithm for computing hybrid reachable sets, we informally described how the continuous reachable set method could be adapted to compute the reach-avoid operator from the hybrid algorithm. The result is a constrained HJI PDE which can be added to a level set implementation without any difficulty. The hybrid algorithm was illustrated with several simple aircraft collision avoidance protocols. The final example was a multimode aircraft autolanding system, where the discrete modes represented different wing flap settings.

In conclusion, we have demonstrated that accurate computation of backwards reachable sets is possible for nonlinear continuous and hybrid systems of low dimension. Extension to higher dimensions will be challenging, but may be possible.

7.2 Suggestions for Further Research

As with so much research, the act of putting these results to paper has made clear just how many directions are available for further work.

A user friendly implementation: The present implementation is still an alpha version; while the author can introduce a new model to the system in just a few hours, anybody else would face a significant struggle to do likewise. Now that the technical details of the implementation are stabilizing, we hope to improve its user interface to the point that others can perform reachable set analyses.

Guaranteed overapproximation: Because it uses an implicit surface representation, the time dependent HJI algorithm presented in sections 2.1.3 and 2.2 can determine the boundary of the reachable set much more accurately than the approaches based on viability theory from section 2.3.1. On the other hand, those approaches can guarantee an overapproximation of the true reachable set, which the time dependent HJI formulation cannot. We are investigating whether the two approaches could be used in a two step process: the viability method to develop a coarse but guaranteed overapproximation and the HJI formulation to find a representation of the overapproximation with subgrid accuracy.

Higher accuracy: While our approach can resolve the location of the reachable set's boundary with subgrid accuracy in smooth regions, any approach based on Eulerian grids will inevitably fail to resolve features of the reachable set in regions where the boundary displays high curvature. This behavior is a known and serious problem for other applications of level sets, including the modeling of fluid interfaces where it manifests itself as a failure to conserve volume. The particle level set method [49] combines trajectory based and Hamilton-Jacobi approaches to interface tracking, and thereby does a much better job at conserving volume. While its current form applies only to purely advective flows (no inputs allowed), we are working on ways to adapt it to the reachable set problem to improve our accuracy further.

Forwards reachable sets: As described in section 2.3, Hamilton-Jacobi and viability based algorithms are suitable for backwards reachable sets and trajectory based approaches are appropriate for forwards reachable sets. For the weakly stable and weakly unstable examples studied in this thesis, either type of reachable set can be used for safety verification. However, for strongly stable systems—such as digital circuits—the backwards formulation is strongly unstable and hence numerical approximations may become wildly inaccurate. With these types of systems in mind, we are looking at what kinds of restrictions on dynamics might make HJ formulations of forwards reachable sets possible, and how the particle level set method might be applied.

Safe controller synthesis: The algorithm outlined in section 3.1.5 is just a prototype of how an implicit surface representation of the reachable set might be used to filter potentially unsafe controllers so as to guarantee safety, and further investigation is planned.

Minimum time to reach function: We have formulated reachable set determination as a time-dependent HJI PDE in order to take advantage of high resolution level set methods. Representing the reachable set using the minimum time to reach function, which is the solution to the static HJI PDE, might make the synthesis of safe controllers more stable numerically. In most cases we cannot apply level set methods to approximate the minimum time to reach function, since they assume a continuous solution. Instead, it may be possible to apply the closely related high resolution methods for conservation laws (which may have discontinuous solutions) to get accurate approximate solutions to the static HJI PDE.

Probabilistic models: Treating noise as the input of player II is conservative, but may be overly so. There are close ties between second order Hamilton-Jacobi equations, stochastic differential equations [107], and Markov processes [55]. We intend to explore how these ties could be exploited to handle reachable set analysis for probabilistic models.

Projective overapproximation: We have just begun to study projective techniques. As mentioned in chapter 4, we would like to prove or disprove Conjecture 9, determine for what types of dynamics we can get reasonable overapproximations of the reachable set, and figure out how to choose the projections wisely.

The hybrid algorithm: The present hybrid reachable set algorithm (from section 5.2) depends on the reach-avoid operator achieving a fixed point in each mode, and cannot compute finite time reachable sets. With the continuous reachable set algorithm completed, we are now ready to update the hybrid algorithm to handle finite horizon reachable sets and to elucidate the connections between boundary conditions and forced discrete switches (as separate from switches that are chosen by player I or player II).

Application to autonomous vehicles: Hybrid systems would appear to be excellent models for mobile robots. We believe our analysis techniques will prove useful not only for safety verification and control synthesis, but also for high level behavioral programming in the style of [37] or [33].

Abstraction: Abstraction is a powerful tool for analyzing large, hierarchically constructed systems. While it has been used extensively on discrete systems, only recently have the

concepts been extended to continuous and hybrid systems [2, 112, 113, 133], and then just for restrictive classes of dynamics. The abstraction shown in section 6.1.3 was constructed in an ad hoc manner, but we would like to investigate how our reachability techniques might allow construction of abstractions for systems with general dynamics, if only approximately.

We are, of course, very interested in tackling new systems of all types, and look forward to extending our techniques to cover as wide a variety of examples as possible.

Bibliography

[1] D. Adalsteinsson and J. A. Sethian. The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148:2–22, 1999.

[2] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[3] R. Alur, T.A. Henzinger, and E.D. Sontag, editors. *Hybrid Systems III*. Number 1066 in Lecture Notes in Computer Science. Springer Verlag, 1996.

[4] J.D. Anderson. *Fundamentals of Aerodynamics*. McGraw Hill Inc., New York, 1991.

[5] P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors. *Hybrid Systems V*. Number 1567 in Lecture Notes in Computer Science. Springer Verlag, 1999.

[6] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems IV*. Number 1273 in Lecture Notes in Computer Science. Springer Verlag, 1997.

[7] P. J. Antsaklis. Special issue on hybrid systems: Theory and applications. *Proceedings of the IEEE*, 88(7), July 2000.

[8] Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors. *Hybrid Systems II*. Number 999 in Lecture Notes in Computer Science. Springer Verlag, 1995.

[9] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In N. Lynch and B.H. Krogh, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 21–31. Springer Verlag, 2000.

[10] E. Asarin, T. Dang, and O. Maler. d/dt: A verification tool for hybrid systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2893–2898, Orlando, FL, 2001.

[11] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Proceedings of Hybrid Systems II*, number 999 in Lecture Notes in Computer Science. Springer Verlag, Cambridge, 1995.

[12] J.-P. Aubin. *Viability Theory.* Systems & Control: Foundations & Applications. Birkhäuser, 1991.

[13] J.-P. Aubin. Systems of Hamilton-Jacobi-Bellman equations: A viability approach. Lecture Notes, June-July, 2001.

[14] J.-P. Aubin and H Frankowska. *Set Valued Analysis.* Birkhäuser, 1990.

[15] J.-P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Viability and invariance kernels of impulse differential inclusions. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1639–1644, Orlando, FL, 2001.

[16] J.-P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Impulse differential inclusions: A viability approach to hybrid systems. *IEEE Transactions on Automatic Control*, AC-47(1):2–20, 2002.

[17] T. Başar and G. J. Olsder. *Dynamic Non-cooperative Game Theory.* Academic Press, second edition, 1995.

[18] M. Bardi. Some applications of viscosity solutions to optimal control and differential games. In I. Capuzzo-Dolcetta and P. L. Lions, editors, *Viscosity Solutions and Applications*, number 1660 in Lecture Notes in Mathematics, pages 44–97. Springer, 1995.

[19] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations.* Birkhäuser, Boston, 1997.

[20] M. Bardi, M. Falcone, and P. Soravia. Numerical methods for pursuit-evasion games and viscosity solutions. In M. Bardi, T Parthasarathy, and T. E. S. Raghavan, editors, *Stochastic and Differential Games: Theory and Numerical Methods*, volume 4 of *Annals of International Society of Dynamic Games.* Birkhäuser, 1999.

[21] A. Bayen and C. Tomlin. Nonlinear hybrid automaton model for aircraft landing. Technical Report SUDAAR 737, Dept. of Aeronautics and Astronautics, Stanford University, Stanford, CA, 2001.

[22] A. M. Bayen, E. Crück, and C. J. Tomlin. Overapproximations of unsafe sets for continuous and hybrid systems: Solving the Hamilton-Jacobi equation using viability techniques. In C. J. Tomlin and M. R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, number 2289 in Lecture Notes in Computer Science, pages 90–104. Springer Verlag, 2002.

[23] Alexandre M. Bayen, Ian Mitchell, Meeko Oishi, and Claire J. Tomlin. Automatic envelope protection and cockpit interface analysis of an autoland system using hybrid system theory. Submitted to the AIAA Journal of Guidance, Control, and Dynamics, August 2002.

[24] G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, number 2034 in Lecture Notes in Computer Science, pages 147–161. Springer Verlag, 2001.

[25] A. Bemporad, F. D. Torrisi, and M. Morari. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 45–59. Springer Verlag, 2000.

[26] M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors. *Hybrid Systems: Computation and Control*. Number 2034 in Lecture Notes in Computer Science. Springer Verlag, Rome, Italy, 2001.

[27] S. Bortoletto. The Bellman equation for constrained deterministic optimal control problems. *Differential and Integral Equations*, 6(4):905–924, 1993.

[28] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 73–88. Springer Verlag, 2000.

[29] O. Bournez, O. Maler, and A. Pnueli. Orthogonal polyhedra: Representation and computation. In F. Vaandrager and J. van Schuppen, editors, *Hybrid Systems: Computation and Control*, number 1569 in Lecture Notes in Computer Science, pages 46–60. Springer Verlag, 1999.

[30] W. E. Boyce and R. C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. John Wiley & Sons, fourth edition, 1986.

[31] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM, Philadelphia, 1994.

[32] M. S. Branicky. Multiple Lyapunov functions and other tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, 1998.

[33] M. S. Branicky and G. Zhang. Solving hybrid control problems: Level sets and behavioral programming. In *Proceedings of the American Control Conference*, pages 1175–1180, Chicago, IL, 2000.

[34] M. Broucke, M. D. Di Benedetto, S. Di Gennaro, and A. Sangiovanni-Vincentelli. Optimal control using bisimulations: Implementation. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, number 2034 in Lecture Notes in Computer Science, pages 175–188. Springer Verlag, 2001.

[35] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[36] P. Burchard, L.-T. Cheng, B. Merriman, and S. Osher. Motion of curves in three spatial dimensions using a level set approach. *Journal of Computational Physics*, 170:720–741, 2001.

[37] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 18(6):534–555, June 1999.

[38] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Set-valued numerical analysis for optimal control and differential games. In M. Bardi, T Parthasarathy, and T. E. S. Raghavan, editors, *Stochastic and Differential Games: Theory and Numerical Methods*, volume 4 of *Annals of International Society of Dynamic Games*. Birkhäuser, 1999.

[39] D. Chopp. Computing minimal surfaces via level set curvature flow. *Journal of Computational Physics*, 106:77–91, 1993.

[40] A. Chutinan and B. H. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In F. Vaandrager and J. H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, number 1569 in Lecture Notes in Computer Science, pages 76–90. Springer Verlag, New York, 1999.

[41] A. Chutinan and B. H. Krogh. Verification of infinite-state dynamic systems using approximate quotient transition systems. *IEEE Transactions on Automatic Control*, 46(9):1401–1410, 2001.

[42] F. H. Clarke, Yu. S. Ledyaev, R. J. Stern, and P. R. Wolenski. *Nonsmooth Analysis and Control Theory*. Graduate Texts in Mathematics. Springer, 1998.

[43] M. G. Crandall, L. C. Evans, and P.-L. Lions. Some properties of viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 282(2):487–502, 1984.

[44] M. G. Crandall and P.-L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 277(1):1–42, 1983.

[45] M. G. Crandall and P.-L. Lions. Two approximations of solutions of Hamilton-Jacobi equations. *Mathematics of Computation*, 43(167):1–19, 1984.

[46] T. Dang and O. Maler. Reachability analysis via face lifting. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control*, number 1386 in Lecture Notes in Computer Science, pages 96–109. Springer Verlag, 1998.

[47] Thao Dang. *Vérification et synthèse des systèmes hybrides*. PhD thesis, Institut National Polytechnique de Grenoble (Verimag), 2000.

[48] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik 1*, pages 269–271, 1959.

[49] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, In Press.

[50] L. C. Evans. *Partial Differential Equations.* American Mathematical Society, Providence, Rhode Island, 1998.

[51] L. C. Evans and P. E. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana University Mathematics Journal*, 33(5):773–797, 1984.

[52] M. Falcone. Numerical solution of dynamic programming equations. In *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations.* Birkhäuser, 1997. Appendix A of [19].

[53] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics*, 152:457–492, 1999.

[54] A. Flaig and R. Hilbig. High-lift design for large civil aircraft. In *AGARD Conference Proceedings 515*, France, October 1992.

[55] W. H. Fleming and H. M. Soner. *Controlled Markov Processes and Viscosity Solutions.* Springer-Verlag, 1993.

[56] H. Frankowska and M Quincampoix. Viability kernels of differential inclusions with constraints: Algorithm and applications. *Mathematics of Systems, Estimation and Control*, 1(3):371–388, 1991.

[57] R. Y. Gazit. *Aircraft Surveillance and Collision Avoidance using GPS.* PhD thesis, Department of Aeronautics and Astronautics, Stanford University, 1996.

[58] R. Ghosh and C. J. Tomlin. Nonlinear inverse dynamic control for mode-based flight. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Denver, CO, August 2000.

[59] Shankar G. Govindaraju and David L. Dill. Verification by approximate forward and backward reachability. In *International Conference on Computer Aided Design*, pages 366–370, San Jose, CA, 1998.

[60] Shankar G. Govindaraju and David L. Dill. Counterexample-guided choice of projections in approximate symbolic model checking. In *International Conference on Computer Aided Design*, pages 115–119, San Jose, CA, 2000.

[61] Shankar G. Govindaraju, David L. Dill, and Jules P. Bergmann. Improved approximate reachability using auxiliary state variables. In *36th Design Automation Conference*, pages 312–316, New Orleans, LA, 1999.

[62] Shankar G. Govindaraju, David L. Dill, Alan J. Hu, and Mark A. Horowitz. Approximate reachability with BDDs using overlapping projections. In *35th Design Automation Conference*, pages 451–456, San Francisco, CA, 1998.

[63] M.R. Greenstreet and I. Mitchell. Integrating projections. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control*, number 1386 in Lecture Notes in Computer Science, pages 159–174. Springer Verlag, 1998.

[64] M.R. Greenstreet and I. Mitchell. Reachability analysis using polygonal projections. In F. Vaandrager and J. van Schuppen, editors, *Hybrid Systems: Computation and Control*, number 1569 in Lecture Notes in Computer Science, pages 103–116. Springer Verlag, 1999.

[65] Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors. *Hybrid Systems*. Number 736 in Lecture Notes in Computer Science. Springer Verlag, 1993.

[66] A. Hassibi and S. Boyd. Quadratic stabilization and control of piecewise-linear systems. In *Proceedings of the American Control Conference*, pages 3659–3664, Phildelphia, PA, 1998.

[67] A. Hassibi, S. Boyd, and J. P. How. A class of Lyapunov functionals for analyzing hybrid dynamical systems. In *Proceedings of the American Control Conference*, pages 2455–2460, San Diego, CA, 1999.

[68] T. A. Henzinger, P. H. Ho, and H. Wong-Toi. A user guide to HyTech. In E. Brinksma, W. Cleaveland, K. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, number 1019 in Lecture Notes in Computer Science, pages 41–71. Springer Verlag, 1995.

[69] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.

[70] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. In O. Grumberg, editor, *Proceedings of the* 9^{th} *International Conference on Computer Aided Verification*, number 1254 in Lecture Notes in Computer Science, pages 460–463. Springer Verlag, 1997.

[71] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 130–144. Springer Verlag, 2000.

[72] T. A. Henzinger and S. Sastry, editors. *Hybrid Systems: Computation and Control*. Number 1386 in Lecture Notes in Computer Science. Springer Verlag, Berkeley, CA, 1998.

[73] H. Hindi and S. Boyd. Analysis of linear systems with saturation using convex optimization. In *Proceedings of the IEEE Conference on Decision and Control*, pages 903–908, Tampa, FL, 1998.

[74] Haitham Hindi. *Local Analysis of Perturbed Linear Systems with Application to Saturating Control Systems*. PhD thesis, Department of Electrical Engineering, Stanford University, 2000.

[75] R. Isaacs. *Differential Games*. John Wiley, 1967.

[76] J. W. Jackson and S. M. Green. Control applications and challenges in air traffic management. In *Proceedings of the American Control Conference*, Philadelphia, PA, 1998.

[77] L. Jenkinson, P. Simpkin, and D. Rhodes. *Civil Jet Aircraft Design*. American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 1999. `http://www.bh.com/companions/aerodata`.

[78] K. H. Johansson, J. Lygeros, S. S. Sastry, and M. Egerstedt. Simulation of zeno hybrid automata. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3538–3543, Phoenix, AZ, 1999.

[79] K. J. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. On the regularization of zeno hybrid automata. *Systems and Control Letters*, 38:141–150, 1999.

[80] M. Johansson and A. Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):555–559, 1998.

[81] S. Kahne and I. Frolow. Air traffic management: Evolution with technology. *IEEE Control Systems Magazine*, 16(4):12–21, 1996.

[82] B. Krogh and N. Lynch, editors. *Hybrid Systems: Computation and Control*. Number 1790 in Lecture Notes in Computer Science. Springer Verlag, Pittsburgh, PA, 2000.

[83] I. M. Kroo. *Aircraft Design: Synthesis and Analysis*. Desktop Aeronautics Inc., Stanford, California, 1999.

[84] R. P. Kurshan and K. L. McMillan. Analysis of digital circuits through symbolic reduction. *IEEE Transactions on Computer-Aided Design*, 10(11):1356–1371, 1991.

[85] A. B. Kurzhanski and I. Vályi. *Ellipsoidal calculus for estimation and control*. Birkhäuser, Boston, 1997.

[86] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 202–214. Springer Verlag, 2000.

[87] G. Lafferriere, G. J. Pappas, and S. Sastry. Hybrid systems with finite bisimulations. In *Hybrid Systems V*, number 1567 in Lecture Notes in Computer Science. Springer Verlag, 1999.

[88] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computations for families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, 2001.

[89] K. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1, 1997.

[90] J. Lygeros. On the characterisation of invariant sets by partial differential equations. Unpublished Manuscript, October, 2001.

[91] J. Lygeros, S. S. Sastry, and C. J. Tomlin. Lecture notes for aa 278a (hybrid dynamical systems). 2002.

[92] J. Lygeros, C. Tomlin, and S. Sastry. On controller synthesis for nonlinear hybrid systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2101–2106, Tampa, FL, 1998.

[93] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), 1999.

[94] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In Ernst W. Mayr and Claude Puech, editors, *STACS 95: Theoretical Aspects of Computer Science*, number 900 in Lecture Notes in Computer Science, pages 229–242. Springer Verlag, Munich, 1995.

[95] A. W. Merz. The game of two identical cars. *Journal of Optimization Theory and Applications*, 9(5):324–343, 1972.

[96] `http://cherokee.stanford.edu/~mitchell`.

[97] `http://www-sccm.stanford.edu/~mitchell`.

[98] I. Mitchell. Games of two identical vehicles. Technical Report SUDAAR 740, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, July 2001.

[99] I. Mitchell, A. Bayen, and C. J. Tomlin. Computing reachable sets for continuous dynamic games using level set methods. Submitted January 2002 to *IEEE Transactions on Automatic Control*.

[100] I. Mitchell, A. Bayen, and C. J. Tomlin. Validating a Hamilton-Jacobi approximation to hybrid system reachable sets. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, number 2034 in Lecture Notes in Computer Science, pages 418–432. Springer Verlag, 2001.

[101] I. Mitchell and R. Fedkiw. A level set algorithm localized in time and space. In preparation.

[102] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 310–323. Springer Verlag, 2000.

[103] I. Mitchell and C. J. Tomlin. Overapproximating reachable sets by Hamilton-Jacobi projections. Submitted April 2002 to *Journal of Scientific Computing*.

[104] M. S. Nolan. *Fundamentals of Air Traffic Control*. Brooks/Cole Publishing, Pacific Grove, CA, third edition, 1999.

[105] M. Oishi, I. Mitchell, A. Bayen, C. Tomlin, and A Degani. Hybrid verification of an interface for an automatic landing. In *Proceedings of the IEEE Conference on Decision and Control*, Las Vegas, NV, 2002. To appear.

[106] Meeko Oishi, Claire J. Tomlin, and Asaf Degani. Verification of user interfaces for hybrid systems. Submitted as a NASA Technical Memorandum, Ames Research Center, August 2002.

[107] B. K. Øksendal. *Stochastic Differential Equations: an Introduction with Applications*. Springer, fifth edition, 1998.

[108] S. Osher, L.-T. Cheng, M. Kang, H. Shim, and Y.-H. Tsai. Geometric optics in a phase space based level set and Eulerian framework. preprint, submitted to *J. Computational Physics*, 2001.

[109] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[110] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[111] S. Osher and Chi-Wang Shu. High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations. *SIAM Journal on Numerical Analysis*, 28(4):907–922, 1991.

[112] G. J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, AC-45(6):1144–1160, 2000.

[113] G. J. Pappas and S. Simić. Consistent abstractions of affine control systems. *IEEE Transactions on Automatic Control*, AC-47(5):745–756, 2002.

[114] V. S. Patsko and V. L. Turova. Level sets of the value function in differential games with the homicidal chauffeur dynamics. *International Game Theory Review*, 3(1):67–112, 2001.

[115] V. S. Patsko and V. L. Turova. Semipermeable curves and level sets of the value function in differential games with the homicidal chauffer dynamics. In K. H. Hoffmann, I. Lasiecka, G. Leugering, and J. Sprekels, editors, *Optimal Control of Complex Structures*, number 139 in International Series of Numerical Mathematics, pages 191–202. Birkhäuser, 2002.

[116] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE based fast local level set method. *Journal of Computational Physics*, 165:410–438, 1999.

[117] T. S. Perry. In search of the future of air traffic control. *IEEE Spectrum*, 34(8):18–35, 1997.

[118] L. Prandtl and O.G. Tietjens. *Applied Hydro- and Aeromechanics*. Dover Publications (Eng. Societies Monographs), New York, 1957 (1934).

[119] J. Preußig and H. Wong-Toi. A procedure for reachability analysis of rectangular automata. In *Proceedings of the American Control Conference*, pages 1674–1678, Chicago, IL, 2000.

[120] Radio Technical Commission for Aeronautics. Minimum aviation system performance standards for Automatic Dependent Surveillance-Broadcast (ADS-B). Technical report, RTCA-186, February 1997. DRAFT 4.0.

[121] S. Rogers, K. Roth, H. Cao, J. Slotnick, M. Whitlock, S. Nash, and M. Baker. Computation of viscous flow for a Boeing 777 aircraft in landing configuration. In *AIAA Conference Proceedings*, number 2000-4221, October 1992.

[122] J. Roskam and C.-T. Lan. *Airplane Aerodynamics and Performance*. Design, Analysis, and Research Corporation, Lawrence, Kansas, 1997.

[123] P. Saint-Pierre. Approximation of the viability kernel. *Applied Mathematics and Optimization*, 29:187–209, 1994.

[124] Shankar Sastry. *Nonlinear Systems: Analysis, Stability and Control*. Springer-Verlag, New York, 1999.

[125] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, New York, 1999.

[126] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations. *Proceedings of the National Academy of Sciences, USA*, 98(20):11069–11074, 2001.

[127] O. Shakernia, G. J. Pappas, and S. S. Sastry. Decidable controller synthesis for classes of linear systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 407–420. Springer Verlag, 2000.

[128] Jong-Yeob Shin. Analysis of linear parameter varying system models based on reachable sets. Technical report, NASA/CR–2001–211231, ICASE report No. 2001-30, ICASE, NASA Langley Research Center, Hampton, VA, October 2001.

[129] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77:439–471, 1988.

[130] B. I. Silva, O. Stursberg, B. H. Krogh, and S. Engell. An assessment of the current status of algorithmic approaches to the verification of hybrid systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2867–2874, Orlando, FL, 2001.

[131] B.I. Silva and B. H. Krogh. Formal verification of hybrid systems using *CheckMate*: A case study. In *Proceedings of the American Control Conference*, pages 1679–1683, Chicago, IL, 2000.

[132] M. Sussman, P. Smereka, and S. Osher. An improved level set method for incompressible two-phase flow. *Journal of Computational Physics*, 114:145–159, 1994.

[133] P. Tabuada, G. J. Pappas, and P. Lima. Composing abstractions of hybrid systems. In C. J. Tomlin and M. R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, number 2289 in Lecture Notes in Computer Science, pages 436–450. Springer Verlag, 2002.

[134] A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In C. J. Tomlin and M. R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, number 2289 in Lecture Notes in Computer Science, pages 465–478. Springer Verlag, 2002.

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

[136] C. J. Tomlin and M. R. Greenstreet, editors. *Hybrid Systems: Computation and Control*. Number 2289 in Lecture Notes in Computer Science. Springer Verlag, Stanford, CA, 2002.

[137] C. J. Tomlin, I. Mitchell, and R. Ghosh. Safety verification of conflict resolution maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 2(2):110–120, 2001. June.

[138] Claire J. Tomlin. *Hybrid Control of Air Traffic Management Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1998.

[139] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, AC-40(9):1528–1538, 1995.

[140] F. Vaandrager and J. H. van Schuppen, editors. *Hybrid Systems: Computation and Control*. Number 1569 in Lecture Notes in Computer Science. Springer Verlag, Berg en Dal, the Netherlands, 1999.

[141] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proceedings of the IEEE Conference on Decision and Control*, San Diego, CA, 1997.

[142] S. Yovine. Kronos: A verification tool for real-time systems. *Software Tools for Technology Transfer*, 1:123–133, 1997.