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 $\mathcal{A} \subset \mathbb{R}^{m_a}$ and $\mathcal{B} \subset \mathbb{R}^{m_b}$ 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

$$b_3(s) = b_1(s - h) \qquad \text{for } s \in [t + h, 0], h > 0$$

$$b_4(s) = \begin{cases} b_1(s), & \text{for } s \in [t, r]; \\ b_2(s), & \text{for } s \in]r, 0]; \end{cases} \quad \text{for any } r \in [t, 0], s \in [t, 0].$$

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

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

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.

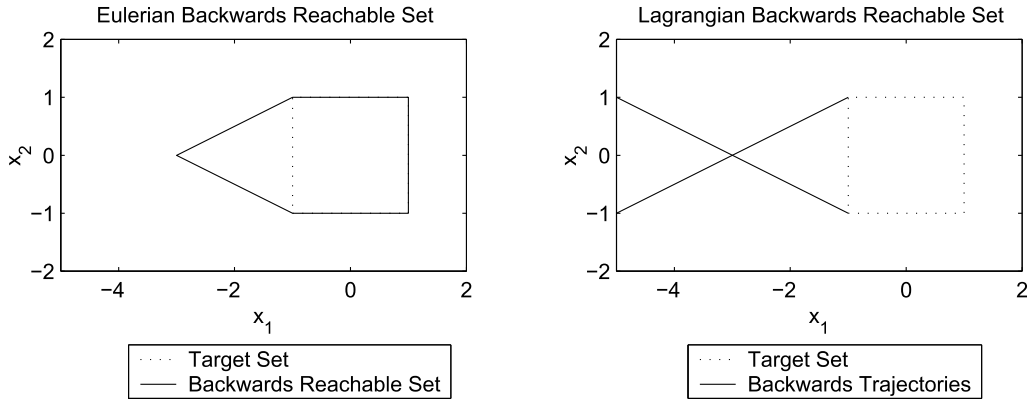

Figure 2.1: Comparing Eulerian and Lagrangian backwards reachable sets.

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). □