Safety Preserving Control Synthesis for Sampled Data Systems

Ian M. Mitchell*

Department of Computer Science, University of British Columbia

Shahab Kaynama, Mo Chen

Department of Electrical Engineering & Computer Science, University of California, Berkeley

Meeko Oishi

Department of Electrical & Computer Engineering, University of New Mexico

---

Abstract

In sampled data systems the controller receives periodically sampled state feedback about the evolution of a continuous time plant, and must choose a constant control signal to apply between these updates; however, unlike purely discrete time models the evolution of the plant between updates is important. In this paper we describe an abstract algorithm for approximating the discriminating kernel (also known as the maximal robust control invariant set) for a sampled data system with continuous state space, and then use this operator to construct a switched, set-valued feedback control policy which ensures safety. We show that the approximation is conservative for sampled data systems. We then demonstrate that the key operations—the tensor products of two sets, invariance kernels, and a pair of projections—can be implemented in two formulations: One based on the Hamilton-Jacobi partial differential equation which can handle nonlinear dynamics but which

*Corresponding author. Mailing address: 2366 Main Mall, Vancouver, BC, Canada, V6T 1Z4. Phone: 604-822-2317. Fax: 604-822-5485.

Email addresses: `mitchell@cs.ubc.ca` (Ian M. Mitchell), `kaynama@eecs.berkeley.edu` (Shahab Kaynama), `mochen72@eecs.berkeley.edu` (Mo Chen), `moishi@ece.unm.edu` (Meeko Oishi)

URL: `http://www.cs.ubc.ca/~mitchell` (Ian M. Mitchell)

scales poorly with state space dimension, and one based on ellipsoids which scales well with state space dimension but which is restricted to linear dynamics. Each version of the algorithm is demonstrated numerically on a simple example.

Keywords: nonlinear systems, sampled data, control synthesis, continuous reachability, Hamilton-Jacobi equations, viability, ellipsoids

1. Introduction

A wide variety of reachability and viability algorithms for continuous and hybrid systems have been proposed in the literature over the last decade, but they have for the most part been driven by safety verification problems; for example, given initial and terminal sets in the state space, do there exist trajectories leading from the former to the latter? For the purposes of system design and debugging, this boolean decision problem is often augmented by a request for counterexamples if the system is unsafe (for example, see [1]). When the system has inputs, however, there is a much less well-studied challenge: Given a particular state, how can those inputs be chosen to maintain safety?

Here we study that problem in the context of sampled data systems. A common design pattern in cyber-physical systems consists of a digital controller receiving periodically sampled state feedback about the continuous time evolution of a continuous (or hybrid) state plant, and then generating a control signal (typically constant) to use until the next sample time. Feedback controllers for such systems are often designed using discrete time approaches, but that treatment ignores the states through which the plant evolves between sample times. Sampled data control takes the continuous time trajectories of the plant into account.

In this paper we propose an algorithm for synthesizing a permissive but safe control policy (also known as a feedback control law) for continuous state sampled data systems. It is safe in the sense that if the system is in a state which is not identified as inevitably unsafe and control signals are chosen from this policy at the sample times, then the system will not leave the constraint set over the safety horizon. It is permissive in the sense that it is set-valued when possible, so that other criteria can be taken into account in choosing the final control signal while still maintaining safety; for example, minimum control effort in an energy constrained situation, or proximity to the human operator's input in a collaborative control scenario.

Viability theory [2] defines a number of constructs for exploring the safe subset of a constraint set. Perhaps the most familiar is the invariance kernel: the set of states from which all trajectories remain inside the constraint no matter what disturbance input signal is applied. The viability kernel is dual to the invariance kernel and is also known as the maximal controlled invariant set: the set of states from which at least one control input signal gives rise to a trajectory which remains inside the constraint set. Finally, the discriminating kernel combines both concepts and could be thought of as a robust version of the viability kernel or maximal controlled invariant set: the set of states from which a least one control signal gives rise to a trajectory which remains inside the constraint set despite the actions of disturbance inputs.

We formulate our algorithm in terms of finite horizon versions of the discriminating and invariance kernels, although it could just as easily be formulated in terms of backward reach tubes. In fact, this algorithm is a generalization of the algorithm presented in [3], which was itself an extension of the algorithm proposed in [4]; both of those algorithms were formulated using backward reachability. Relative to those papers, the key new contributions of this paper are:

- Reformulation of the algorithm in terms of discriminating and invariance kernels.
- An abstract version of the algorithm which does not depend on the Hamilton-Jacobi (HJ) partial differential equation (PDE).
- An instantiation of the abstract algorithm's operators using ellipsoidal reachability constructs; although this version is restricted to systems with linear dynamics, it scales much better with state space dimension than the HJ PDE version.

We also replicate in a viability theory context several contributions from [3]:

- Demonstration that the computed sampled data discriminating kernel is a conservative estimate of the true sampled data discriminating kernel.
- Partition of the state space into regions where the full control authority can be used safely or where only a subset may be used while maintaining safety.
- An instantiation of the abstract algorithm's operators using HJ PDEs which can be applied to systems with nonlinear dynamics.

The remainder of the paper is organized as follows. Section 2 formalizes the problem, while section 3 discusses related work. Section 4 outlines the abstract sampled data invariance kernel algorithm, proves its conservativeness, and shows how it can be used to synthesize a permissive but safe control policy. Sections 5 and 6 respectively provide a Hamilton-Jacobi formulation and an ellipsoidal formulation of the abstract algorithm, discuss practical implementation details and provide simple examples.

This paper is an extended version of [3], which was presented at the 4^{th} IFAC Conference on the Analysis and Design of Hybrid Systems (Eindhoven, the Netherlands, June 6–8, 2012).

2. Problem Definition

Consider a system whose evolution is modelled by the ordinary differential equation (ODE)

$$\dot{x} = f(x, u, v) \tag{1}$$

with initial condition $x(0) = x_0$, where $x \in \Omega$ is the state, $\Omega \subset \mathbb{R}^{d_x}$ (or some similar vector space of dimension d_x) is the state space, $u \in \mathcal{U}$ is the control input, $v \in \mathcal{V}$ is the disturbance input, $\mathcal{U} \subset \mathbb{R}^{d_u}$ and $\mathcal{V} \subset \mathbb{R}^{d_v}$ are assumed to be compact, and the dynamics $f : \Omega \times \mathcal{U} \times \mathcal{V} \to \Omega$ are assumed to be Lipschitz continuous in x and continuous in u and v. Additional assumptions may be necessary for particular versions of the abstract algorithm; for example, f must be linear and $\mathcal{U}$ and $\mathcal{V}$ must be ellipsoids for the ellipsoidal formulation in section 6. Input u is used to keep the system within the imposed state constraints. Input v seeks to drive the system outside the state constraints, and can be used to model the effects of potentially adversarial agents on system evolution, to treat uncertainty in the dynamics in a worst case fashion, to improve the robustness of the results, or it can be omitted for deterministic scenarios.

We will assume that for feedback control purposes the state is sampled at times $t_k \triangleq k\delta$ for some fixed $\delta > 0$ and integer k, and that the control signal is constant between sample times. As a consequence, the actual dynamics are of the form

$$\dot{x}(t) = f(x(t), u_{\text{pw}}(t), v(t)) \tag{2}$$

where the piecewise constant input signal $u_{\text{pw}}(\cdot)$ is chosen according to

$$u_{\text{pw}}(t) = u_{\text{fb}}(x(t_k)) \text{ for } t_k \le t < t_{k+1} \tag{3}$$

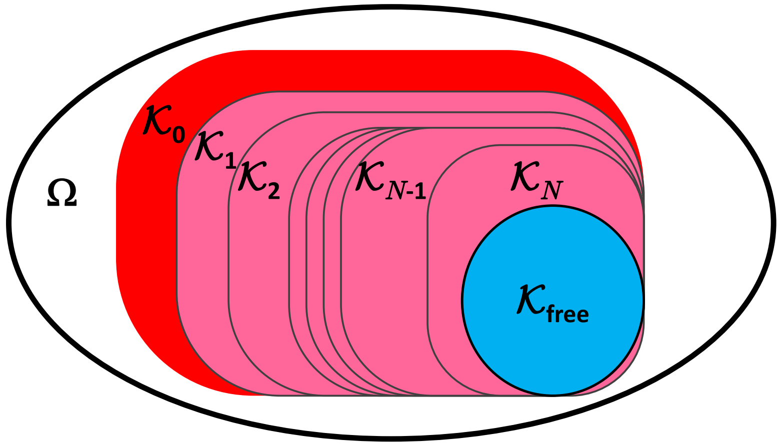

Figure 1: The subdivision of the state space. The constraint set $\mathcal{K}_0$ and the state space Ω are specified in the problem definition. The finite horizon safe sets $\mathcal{K}_k$ for horizons $k > 0$, the free control set $\mathcal{K}_{\text{free}}$ and the mandatory control set $\mathcal{K}_{\text{ctrl}} = \mathcal{K}_0 \setminus \mathcal{K}_{\text{free}}$ (not shown explicitly, but it is the union of the red and all of the pink sets) are determined by the algorithms proposed in this paper.

and $u_{\text{fb}} : \Omega \to \mathcal{U}$ is a feedback control policy. It was shown in [5] that there exists a control policy which renders the system safe if and only if there exists a feedback control policy which renders the system safe, so we restrict ourselves to feedback control policies without loss of generality. Input signal $v(\cdot)$ is not constrained to be piecewise constant, but is merely assumed to be measurable. Note that because the feedback control policy is time sampled, the dynamics (2) *cannot* be written in the form $\dot{x} = f(x, v)$.

The state constraint $\mathcal{K}_0 \subset \Omega$ that we seek to maintain for safety is assumed to be the complement of an open set [6]. We divide the state space Ω into nested subsets as shown in figure 1. The outermost is the safety constraint $\mathcal{K}_0$. The finite horizon safe sets $\mathcal{K}_k$ contain states which give rise to trajectories which satisfy the safety constraint for at least time $k\delta$ provided the correct u_{fb} is chosen. Finally, given a fixed horizon $k = N$ of interest, we determine a free control subset $\mathcal{K}_{\text{free}}$ within which any u_{fb} can be chosen at the next sampling instant. The complement of this free control set with respect to the safety constraint is the mandatory control set $\mathcal{K}_{\text{ctrl}} \triangleq \mathcal{K}_0 \setminus \mathcal{K}_{\text{free}}$ within which we will constrain u_{fb} in order to ensure safety. We will determine the sets $\mathcal{K}_k$ for $1 \leq k \leq N$, $\mathcal{K}_{\text{ctrl}}$ and $\mathcal{K}_{\text{free}}$ through a series of finite horizon invariance kernel calculations. In some cases it may be possible to achieve $N = \infty$ in a finite number of steps, and thereby ensure safety over an infinite horizon.

3. Related Work

Sampled data systems have a long history in control engineering, and in recent decades that research has broadened to include nonlinear as well as linear systems; however, the focus is typically on traditional control objectives such as stability (for example, see [7, 8] and the citations within).

In the context of verification, research on "sampled data systems" has focused on hybrid systems in which some subset of the mode switches can only occur at sampling times. In [9], a "sampled data hybrid automata" formalism was introduced and used to extend the CheckMate hybrid system verification tool to study a version of such a system with deterministic continuous dynamics. In [10] the authors study a "piecewise affine" version of such a system; in other words, the state space is partitioned into polyhedra which specify the modes, in each of which the continuous dynamics are affine with a control input. An algebraic condition is given which ensures the existence of a control input signal which drives the system from an initial set of states to a specific final state; however, the input is assumed to be piecewise continuous (not piecewise constant) and it is only the mode switching which is sampled. In [11], the authors consider hybrid systems with nondeterministic continuous dynamics and a controller which can enable and/or force mode switches at sampling times, but assume that trajectories of those dynamics are explicitly available. They then derive necessary and sufficient conditions for a predicate to be control invariant and show that there is always a supremal control invariant subpredicate for any predicate. Such a subpredicate corresponds conceptually to a (hybrid) discriminating kernel of the set defined by the predicate, although for their systems the control input can only influence the mode switching rules, not the continuous evolution. In [12] the authors consider a hybrid system whose continuous dynamics admit only a piecewise constant control input signal; however, they must restrict themselves to affine dynamics within each mode in order to determine an explicit representation of trajectories in terms of linear inequalities and thereby construct their "timed relational abstraction," which can then be composed with a controller and analyzed with discrete time verification tools. In contrast to these earlier works on verification of sampled data systems, our abstract algorithm handles nonlinear continuous dynamics with a piecewise constant control input signal and robustness provided by allowing for a measurable but bounded disturbance input signal. We do not assume availability of explicit solutions for the resulting trajectories. Our algorithm is constructive in that it yields a set-valued control law, although it is potentially conservative. At present our algorithm is restricted to systems with purely continuous state.

Our algorithm is closely related to previous reachability and viability algorithms. We broadly categorize reachability algorithms into Lagrangian (those which follow trajectories of the system) and Eulerian (those which operate on a fixed grid); see [13] for a more extensive discussion of types of reachability algorithms. Most algorithms for systems with nonlinear dynamics and adversarial inputs are currently Eulerian; for example, there are schemes based on viability theory [6, 2], static HJ PDEs [14, 15], or time-dependent HJ PDEs [5, 16, 17]. In all three cases it is possible to synthesize control laws that are optimally permissive: constraints are only placed upon the choice of control along the boundary of the safe or viable set. From a practical perspective, however, such policies are impossible to implement because they require information about the state at all times and the ability to change the control input signal at any time. In contrast, here we assume that state feedback and control signal modification only occur at the periodic sample times, and the control signal is held constant between sample times.

In [4] a time-dependent HJ PDE formulation of sampled data reachability is presented for hybrid automata using the tool [18]. In that case, the HJ PDE is used to find an implicit surface representation of the sampled data backward reach tube, where the piecewise continuous control input signal attempts to drive the trajectory to a terminal set without entering an avoid set, despite the actions of a measurable disturbance input signal. In [3] we modify that algorithm to study the case where the control input signal seeks to avoid the target set, and also examine the relationship between the resulting HJ PDE solutions and the desired reachability operators. As described above, in this paper we create an abstract version of that algorithm formulated in terms of discriminating kernels instead of reachability, and provide an ellipsoidal version of the abstract algorithm in addition to the HJ PDE version.

For systems with linear or affine continuous dynamics, there are a number of Lagrangian algorithms available for reachability; for example, see [19, 20] and the citations within. While these techniques have not traditionally been used for control synthesis, they are amenable to the abstract algorithm described below. We use the tool [20] to implement the ellipsoidal version of the algorithm in section 6. The techniques from [19] have been adapted to discrete time viability kernels in [21], but using them for the algorithm described below will require further modification to handle invariance kernels and continuous time.

An alternative approach to finding safe control policies is through sample based planning schemes, such as the rapidly-exploring random tree (RRT) and its descendants (see [22] and the citations within). Adaptations of RRTs to verification/falsification are proposed in [23, 24], but to synthesize permissive yet safe control policies requires a slightly different but still quite feasible modification of traditional RRTs (to collect sets of safe paths, rather than just the optimal or first path found). Like many sample based schemes RRTs appear to scale better in practice to high dimensional systems than do schemes based on grids, and unlike most Lagrangian approaches they do a good job of covering the state space given sufficient samples. On the other hand, the output of RRTs is not as easily or accurately interpolated into continuous spaces as are grid-based results, and there is no simple method of introducing worst-case disturbance inputs to make the results robust to uncertainty.

4. Abstract Algorithm

In this section we define the finite horizon sampled data discriminating kernel for dynamics (2)–(3), and then show how it can be computed through a sequence of finite horizon continuous time invariance kernels. This construct plus one additional invariance kernel calculation is sufficient to determine the sets $\mathcal{K}_k$, $\mathcal{K}_{\text{ctrl}}$ and $\mathcal{K}_{\text{free}}$. Given these sets, it is possible to define the permissive but safe control policy using a nondeterministic hybrid automaton. In subsequent sections we demonstrate two practical methods of approximating the invariance kernels and resulting control hybrid automaton.

4.1. Preliminary Definitions

The algorithms for constructing the sampled data discriminating kernel and a corresponding set-valued control policy depend upon a number of set-valued maps which we define here. The first map is simply the sampled data discriminating kernel that we seek:

$$\mathsf{Disc}_{\mathsf{sd}}([0,T],\mathcal{S}) \triangleq \{x_0 \in \mathcal{S} \mid \exists u_{\text{pw}}(\cdot), \forall v(\cdot), \forall t \in [0,T], x(t) \in \mathcal{S}\}, \qquad (4)$$

where $x(\cdot)$ solves (2) with initial condition $x(0) = x_0$. The key difference between (4) and continuous time discriminating kernels is that the input signal in (4) must be piecewise constant over each sampling interval.

To construct an approximation to (4) we will sometimes work in an augmented state space

$$\tilde{x} \triangleq \begin{bmatrix} x \\ u \end{bmatrix} \in \tilde{\Omega} \triangleq \Omega \times \mathbb{R}^{d_u}$$

with dynamics

$$\frac{d}{dt}\tilde{x} = \frac{d}{dt}\begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} f(x,u,v) \\ 0 \end{bmatrix} \triangleq \tilde{f}(\tilde{x}, v). \tag{5}$$

To move from the augmented state space back to the original state and control spaces, we need a projection operator from $\tilde{\Omega}$ back into Ω:

$$\mathsf{Proj}_x(\tilde{\mathcal{X}}) \triangleq \left\{ x \in \Omega \;\middle|\; \exists u, \begin{bmatrix} x \\ u \end{bmatrix} \in \tilde{\mathcal{X}} \right\} \quad \text{for } \tilde{\mathcal{X}} \subseteq \tilde{\Omega}, \tag{6}$$

and a projection operator from $\tilde{\Omega}$ into $\mathcal{U}$ for a particular value of x:

$$\mathsf{Proj}_u(\tilde{\mathcal{X}}, x) \triangleq \left\{ u \in \mathcal{U} \;\middle|\; \begin{bmatrix} x \\ u \end{bmatrix} \in \tilde{\mathcal{X}} \right\} \quad \text{for } \tilde{\mathcal{X}} \subseteq \tilde{\Omega} \text{ and } x \in \Omega.$$

From these definitions it is straightforward to show

$$x \in \mathsf{Proj}_x(\tilde{\mathcal{X}}) \implies \mathsf{Proj}_u(\tilde{\mathcal{X}}, x) \neq \emptyset \tag{7}$$

Remark. It may appear to be dangerous from a complexity perspective to advocate augmenting the state space with the control input dimensions when viability algorithms have a reputation for poor scaling with dimension. We do so in this section because the resulting algorithm is conceptually simple. Section 5 will implement this algorithm with a formulation that scales poorly with dimension, but we will show that the lack of motion in the u coordinates allow us to use very coarse sampling and independent calculations in those dimensions. Section 6 will implement the algorithm in a formulation that scales polynomially with dimension, so the added dimensions are not as much of a concern.

Although algorithms exist to approximate both continuous and discrete time discriminating kernels directly, in this paper we will construct an approximation of the sampled data discriminating kernel (4) using a sequence of invariance kernels. In some cases these invariance kernels will be computed over the augmented dynamics (5) with only input v treated as a disturbance, while in other cases they will be computed over the original dynamics (1) with both inputs u and v treated as disturbances. For that reason, we define the invariance kernel in terms of a set of dummy variables: system dynamics $\dot{y} = g(y, w)$ with initial condition $y(0) = y_0$, solution $y(\cdot)$, and disturbance input w.

$$\mathsf{Inv}([0, T], \mathcal{S}, w, g) \triangleq \{y_0 \in \mathcal{S} \mid \forall w(\cdot), \forall t \in [0, T], y(t) \in \mathcal{S}\}, \tag{8}$$

Depending on the situation, dummy state vector y may be either x or $\tilde{x}$, dummy dynamics g may be either f or $\tilde{f}$, and dummy disturbance vector w may be either v or the concatenated vector $\begin{bmatrix} u & v \end{bmatrix}^T$. Note that the symbol "$w$" is included as a parameter of the invariance kernel simply to indicate over which inputs the kernel is invariant; the corresponding input signal $w(\cdot)$ is determined by the universal quantifier inside the definition and is not itself an argument to the invariance kernel.

4.2. Approximating the Sampled Data Discriminating Kernel through Iterated Invariance Kernels

We start by examining a single sample period. Let the single step sampled data discriminating kernel be defined as

$$\mathsf{Disc}_1(\mathcal{S}) \triangleq \mathsf{Disc}_{\mathsf{sd}}([0, \delta], \mathcal{S}). \tag{9}$$

This discriminating kernel can be determined through an invariance kernel in the augmented state space. For notational convenience we define

$$\mathsf{Inv}_1(\mathcal{S}) \triangleq \mathsf{Inv}([0, \delta], \mathcal{S} \times \mathcal{U}, v, \tilde{f}) \tag{10}$$

Lemma 1. *The single step sampled data discriminating kernel is the projection of a δ-horizon invariance kernel in the augmented state space*

$$\mathsf{Disc}_1(\mathcal{S}) = \mathsf{Proj}_x(\mathsf{Inv}_1(\mathcal{S})) \tag{11}$$

Proof. We seek to show

$$x_0 \in \mathsf{Disc}_1(\mathcal{S}) \iff x_0 \in \mathsf{Proj}_x(\mathsf{Inv}_1(\mathcal{S})).$$

To show the rightward implication, assume that $x_0 \in \mathsf{Disc}_1(\mathcal{S})$. By (4) there exists a $u_{\mathrm{pw}}(\cdot)$ such that for all $v(\cdot)$ and $t \in [0, \delta]$, $x(t) \in \mathcal{S}$ where $x(\cdot)$ solves (2) with initial condition $x(0) = x_0$. But for $t \in [0, \delta]$, $u_0 \triangleq u_{\mathrm{pw}}(t)$ is a constant by (3), so the augmented trajectory $\tilde{x}(\cdot) = \begin{bmatrix} x(\cdot) & u_0 \end{bmatrix}^T$ satisfies (5). Since $u_0 \in \mathcal{U}$ by (3) and for all $v(\cdot)$, $x(\cdot) \in \mathcal{S}$ over the same time interval, it must be that for all $v(\cdot)$, $\tilde{x}(\cdot) \in \mathcal{S} \times \mathcal{U}$. By (8) we have that $\begin{bmatrix} x_0 & u_0 \end{bmatrix}^T \in \mathsf{Inv}_1(\mathcal{S})$, and hence by (6) that $x_0 \in \mathsf{Proj}_x(\mathsf{Inv}_1(\mathcal{S}))$.

To show the leftward implication, assume that $x_0 \in \mathsf{Proj}_x(\mathsf{Inv}_1(\mathcal{S}))$. By (6) there exists $u_0 \in \mathcal{U}$ such that $\tilde{x}_0 \triangleq \begin{bmatrix} x_0 & u_0 \end{bmatrix}^T \in \mathsf{Inv}_1(\mathcal{S})$. Let $\tilde{x}(\cdot)$ solve (5) with initial condition $\tilde{x}(0) = \tilde{x}_0$ for $t \in [0, \delta]$, and let $x(\cdot)$ be the corresponding state space component of $\tilde{x}(\cdot)$, which by (5) solves (2) with constant input u_0. By (8), $\tilde{x}(t) \in \mathcal{S} \times \mathcal{U}$ for all $v(\cdot)$ and $t \in [0, \delta]$; consequently, $x(t) \in \mathcal{S}$ for all $v(\cdot)$ and $t \in [0, \delta]$. Since $u_{\text{pw}}(\cdot) = u_0$ is a feasible piecewise constant input for $x(\cdot)$ over time interval $[0, \delta]$, by (4) $x_0 \in \mathsf{Disc}_1(\mathcal{S})$. □

Approximation of the sampled data discriminating kernel over longer horizons is then performed recursively

$$\mathsf{Disc}_{k+1}(\mathcal{S}) \triangleq \mathsf{Disc}_1(\mathsf{Disc}_k(\mathcal{S})) \tag{12}$$

4.3. Conservatism of the Approximation

Proposition 2. *The true sampled data discriminating kernel over multiple sample periods is a superset of the recursive approximation*

$$\mathsf{Disc}_k(\mathcal{S}) \subseteq \mathsf{Disc}_{\mathsf{sd}}([0, k\delta], \mathcal{S}).$$

It may be a strict superset for $k > 1$.

Proof. We start by using induction to show containment:

$$x_0 \in \mathsf{Disc}_k(\mathcal{S}) \implies x_0 \in \mathsf{Disc}_{\mathsf{sd}}([0, k\delta], \mathcal{S}).$$

Assume that $x_0 \in \mathsf{Disc}_k(\mathcal{S})$. The implication holds true in the base case $k = 1$ by definition (9). For $k > 1$, $x_0 \in \mathsf{Disc}_k(\mathcal{S})$ implies by (12), (9) and (4) that for all $v(\cdot)$ and $t \in [0, \delta]$ there exists $u^a_{\text{pw}}(\cdot)$ and $x^a(\cdot)$ solving (2) such that $x^a(0) = x_0$ and $x^a(t) \in \mathsf{Disc}_{k-1}(\mathcal{S}) \subseteq \mathcal{S}$; in particular, $x_1 \triangleq x^a(\delta) \in \mathsf{Disc}_{k-1}(\mathcal{S})$. Make the inductive hypothesis that $x_1 \in \mathsf{Disc}_{k-1}(\mathcal{S})$ implies $x_1 \in \mathsf{Disc}_{\mathsf{sd}}([0, (k-1)\delta], \mathcal{S})$. If $x_1 \in \mathsf{Disc}_{\mathsf{sd}}([0, (k-1)\delta], \mathcal{S})$ then for the time independent dynamics (1) we can shift time to show by (4) that for all $v(\cdot)$ and $t \in [\delta, k\delta]$ there exists $u^b_{\text{pw}}(\cdot)$ and $x^b(\cdot)$ solving (2) such that $x^b(\delta) = x_1$ and $x^b(t) \in \mathcal{S}$. Now define

$$x(t) = \begin{cases} x^a(t) & 0 \le t < \delta \\ x^b(t) & \delta \le t \le k\delta \end{cases} \qquad \text{and} \qquad u_{\text{pw}}(t) = \begin{cases} u^a_{\text{pw}}(t) & 0 \le t < \delta \\ u^b_{\text{pw}}(t) & \delta \le t < k\delta. \end{cases}$$

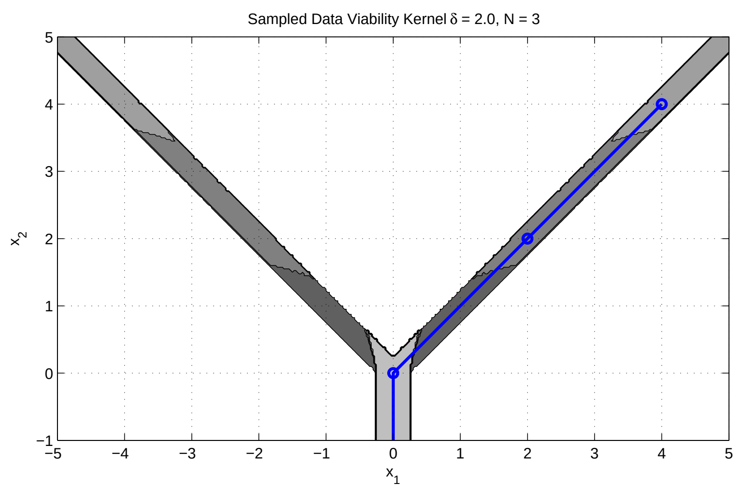

Figure 2: A demonstration that $\mathsf{Disc}_k(\mathcal{S})$ may exclude states which can remain inside $\mathcal{S}$ over horizon $k\delta$. In this case $\mathcal{S}$ is the Y-shaped shaded region. The states in $\mathsf{Disc}_k(\mathcal{S})$ for $k = 0, 1, 2, 3$ are shown darkest to lightest (darker colored sets also contain all lighter colored states). The solid blue line shows a trajectory starting within $\mathsf{Disc}_2(\mathcal{S})$ which nonetheless stays within $\mathcal{S}$ for all time. The input for this trajectory is sampled at the points marked by small circles. Note that the states within the lightest shaded region at the bottom are actually in $\mathsf{Disc}_\infty(\mathcal{S})$, although in this case the computation is performed only up to $k = 3$.

By the arguments above, $x(t) \in \mathcal{S}$ for all $v(\cdot)$ and $t \in [0, k\delta]$, and $u_{\mathrm{pw}}(\cdot)$ is a valid piecewise constant input signal, so $x_0 = x(0) \in \mathsf{Disc}_{\mathsf{sd}}([0, k\delta], \mathcal{S})$.

We demonstrate that strict conservatism is possible through an example. Let $f(x, u, v) = \begin{bmatrix} u & -1 \end{bmatrix}^T$ in (2) with $\mathcal{U} = [-1, +1]$ (there is no input v). Let $\mathcal{S}$ be the Y-shaped shaded region shown in figure 2 (the arms and leg of the Y are assumed to extend outward to infinity). Notice that the upper arms of the Y are chosen to have constant width and a 45° slope. The vertical leg of $\mathcal{S}$ is viable for all $\delta > 0$, but for $\delta = 2$ there are regions of the upper arms which give rise to sampled data trajectories which inevitably leave $\mathcal{S}$; for example, a trajectory starting at $\begin{bmatrix} +1 & +1 \end{bmatrix}^T$ must choose $u \approx -1$ to avoid leaving the lower edge of the right arm of $\mathcal{S}$ almost immediately, but such a choice results in leaving the left edge of the vertical leg of $\mathcal{S}$ at some $t < \delta$. On the other hand, there are states along the upper arms which give rise to trajectories which remain viable for all time; for example, the trajectory

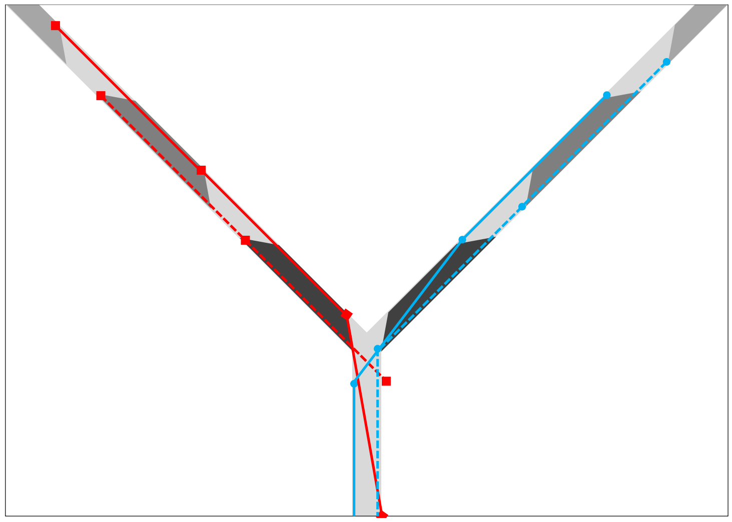

Figure 3: A sketch of the actual $\mathsf{Disc}_{\mathsf{sd}}([0, k\delta], \mathcal{S})$ for $k = 0, 1, 2, 3$ and some sample trajectories for the example in figure 2. The lightest shaded regions (including the periodic gaps between the darker regions on the arms of the Y) are actually within $\mathsf{Disc}_{\mathsf{sd}}([0, \infty], \mathcal{S})$. Two trajectories just at the boundary of safety (both blue, one solid and one dashed) are shown beginning in the right arm of the Y, where samples occur at the circles. Two trajectories just at the boundary of unsafety (both red, one solid and one dashed) are shown beginning in the left arm of the Y, where samples occur at the boxes and trajectories exit the Y just before the final (lowest) sample time. Note that perturbing the unsafe trajectories either up or down will lead to an earlier failure time.

shown in figure 2 starts at $\begin{bmatrix}+4 & +4\end{bmatrix}^T$ and uses input signal

$$u_{\text{pw}}(t) = \begin{cases} -1 & 0 \leq t < 4; \\ 0 & t \geq 4. \end{cases}$$

Despite the existence of these viable patches in the arms of the Y, the set $\mathsf{Disc}_k(\mathcal{S})$ for $k > 2$ completely excludes the arms up to some k-dependent level as shown in figure 2. A sketch of the actual sampled data discriminating kernel $\mathsf{Disc}_{\mathsf{sd}}([0, k\delta], \mathcal{S})$ (which includes the viable patches in the arms) is shown in figure 3. $\square$

Note that this proof and counter-example are different from the ones in [25]: this proof uses the viability formulation and this counter-example's control set is convex.

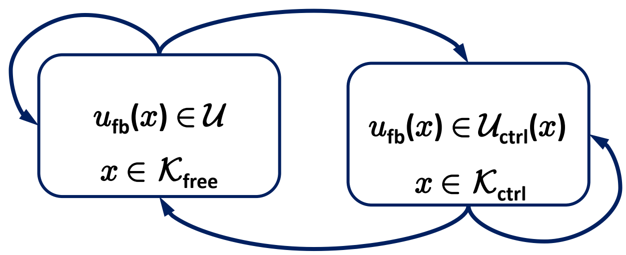

Figure 4: The general form of the switched sampled data control policy. Arrows show transitions which are possible under the policy.

4.4. Subdivision of the Constraint Set

Using the operators defined above, we determine the subdivision of the constraint set $\mathcal{K}_0$ shown in figure 1. The finite horizon safe sets $\mathcal{K}_k$ are (conservatively) approximated using the sampled data discriminating kernel

$$\mathcal{K}_k = \mathsf{Disc}_k(\mathcal{K}_0). \tag{13}$$

The final safe set $\mathcal{K}_N$ is partitioned using one last invariant set calculation, this time under the original dynamics (1) but treating *both* the control u and disturbance v in a worst-case fashion

$$\mathcal{K}_{\text{free}} = \mathsf{Inv}([0,\delta], \mathcal{K}_N, (u,v), f), \tag{14}$$

In other words, $\mathcal{K}_{\text{free}}$ is the set of states which will remain within $\mathcal{K}_N$ for at least time δ no matter what inputs $u(\cdot)$ and $v(\cdot)$ are chosen. Note that in the calculation of $\mathcal{K}_{\text{free}}$ the control input signal $u(\cdot)$ is drawn from the set of measurable functions, so $\mathcal{K}_{\text{free}}$ is also determined in a conservative fashion.

4.5. Control Policy Synthesis

Our permissive but safe control policy takes the form of a hybrid automaton as shown in figure 4. The policy guarantees that states which start in $\mathcal{K}_{\text{free}}$ do not leave $\mathcal{K}_0$ during the time interval $[0, N\delta]$. We do not synthesize a policy for $x \notin \mathcal{K}_0$, since the system has already failed the safety criterion in such states.

In order to be permissive, the policy is often set-valued. In subsequent sections we will examine reasons why one input might be favored over another based on additional information available from specific computational algorithms—for example, an approximation of how deep within a set the future trajectory will stay—but at this stage we treat equally all control signals for which we can guarantee safety.

For $x \in \mathcal{K}_{\text{free}}$, there are no constraints on the input $u_{\text{fb}}(x) \in \mathcal{U}$. For $x \in \mathcal{K}_{\text{ctrl}} = \mathcal{K}_0 \setminus \mathcal{K}_{\text{free}}$, define the safety horizon of x as

$$n(x) \triangleq \begin{cases} N, & \text{if } x \in \mathcal{K}_N \setminus \mathcal{K}_{\text{free}}, \\ k, & \text{if } x \in \mathcal{K}_k \setminus \mathcal{K}_{k+1}. \end{cases} \tag{15}$$

The control policy is given by

$$\mathcal{U}_{\text{ctrl}}(x) \triangleq \mathsf{Proj}_u(\mathsf{Inv}_1(\mathcal{K}_{n(x)-1}), x); \tag{16}$$

in other words, $\mathcal{U}_{\text{ctrl}}(x)$ is the set of constant control values which keeps $\mathcal{K}_{n(x)-1}$ invariant over a single sample period and hence allows x to be part of $\mathcal{K}_{n(x)}$.

Lemma 3. *For all $x \in \mathcal{K}_{ctrl}$, if $n(x) > 0$ then $\mathcal{U}_{ctrl}(x) \neq \emptyset$.*

Proof. Let $x \in \mathcal{K}_{\text{ctrl}}$ such that $n(x) > 0$. By (15), $x \in \mathcal{K}_{n(x)}$, which by (13), (12) and (11) implies that $x \in \mathsf{Proj}_x(\mathsf{Inv}_1(\mathcal{K}_{n(x)-1}))$, which in turn implies by (7) that $\mathsf{Proj}_u(\mathsf{Inv}_1(\mathcal{K}_{n(x)-1}), x) = \mathcal{U}_{\text{ctrl}}(x) \neq \emptyset$. $\square$

4.6. Safety of the Policy

Theorem 4. *Let trajectory $x(\cdot)$ solve (2)–(3) with initial condition $x(0) = x_0$ and sampled feedback control policy*

$$u_{fb}(x) \in \begin{cases} \mathcal{U}_{ctrl}(x), & \text{for } x \in \mathcal{K}_{ctrl}; \\ \mathcal{U}, & \text{for } x \in \mathcal{K}_{free}. \end{cases} \tag{17}$$

If $x_0 \in \mathcal{K}_{free}$, then $x(t) \in \mathcal{K}_0$ for all $t \in [0, (N+1)\delta]$, where $N\delta$ is the horizon used in the computation (14) of $\mathcal{K}_{free}$. If $x_0 \in \mathcal{K}_{ctrl}$, then $x(t) \in \mathcal{K}_0$ for all $t \in [0, n(x_0)\delta]$.

Proof. Consider first the case $x_0 \in \mathcal{K}_{\text{ctrl}}$. By (13) $x_0 \in \mathcal{K}_{n(x_0)}$, which implies by (12) and (9) that $x_0 \in \mathsf{Proj}_x(\mathsf{Inv}_1(\mathcal{K}_{n(x_0)-1}))$ and by (16) that for all $u_0 \in \mathcal{U}_{\text{ctrl}}(x_0)$, $v(\cdot)$ and $t \in [0, \delta]$, $x(t) \in \mathcal{K}_{n(x_0)-1} \subseteq \mathcal{K}_0$, where $x(\cdot)$ solves (2) with fixed input u_0 and initial conditions $x(0) = x_0$. Since $x(\delta) \in \mathcal{K}_{n(x_0)-1}$, use the same argument to construct a new constant input $u_j \in \mathcal{U}_{\text{ctrl}}(x(j\delta))$

and show that $x(t) \in \mathcal{K}_{n(x_0)-j} \subseteq \mathcal{K}_0$ for all $v(\cdot)$ and $t \in [j\delta, (j+1)\delta]$ for all $j = 1, 2, 3 \ldots, n(x_0) - 1$. Concatenating the u_j for $j = 0, 1, 2, \ldots, n(x_0) - 1$ together we arrive at a control signal which satisfies (17) and maintains $x(t) \in \mathcal{K}_0$ for all $t \in [0, n(x_0)\delta]$.

If $x_0 \in \mathcal{K}_{\text{free}}$, then by (14) for all $u(\cdot)$, $v(\cdot)$ and $t \in [0, \delta]$, $x(t) \in \mathcal{K}_N$. In particular, if $x(\delta) \in \mathcal{K}_N$, then by the argument above we can construct a sampled feedback control policy according to (17) such that $x(t) \in \mathcal{K}_0$ for all $t \in [0, (N+1)\delta]$. □

4.7. What About Infinite Horizon?

Corollary 5. *If at some point $\mathcal{K}_{k+1} = \mathcal{K}_k$, then for $x_0 \in \mathcal{K}_\infty \triangleq \mathcal{K}_k$, it is possible to guarantee $x(\cdot) \in \mathcal{K}_\infty$ for all $t > 0$.*

Proof. Let $x_0 \in \mathcal{K}_{k+1} = \mathcal{K}_k$. By (12) and (9), $x_0 \in \mathsf{Proj}_x(\mathsf{Inv}_1(\mathcal{K}_k))$ and by (16) for all $u_0 \in \mathcal{U}_{\text{ctrl}}(x_0)$, $v(\cdot)$ and $t \in [0, \delta]$, $x(t) \in \mathcal{K}_k$. In particular, $x(\delta) \in \mathcal{K}_k = \mathcal{K}_{k+1}$, so use the same argument to construct a new constant input $u_j \in \mathcal{U}_{\text{ctrl}}(x(j\delta))$ and show that $x(t) \in \mathcal{K}_k$ for all $v(\cdot)$ and $t \in [j\delta, (j+1)\delta]$ for all $j = 1, 2, 3, \ldots$. Concatenating the u_j together we arrive at a control signal which satisfies (17) and maintains $x(t) \in \mathcal{K}_k$ for all $t > 0$ (thus justifying the notational choice $\mathcal{K}_k = \mathcal{K}_\infty$). □

In general, there may not be an infinite horizon sampled data discriminating kernel for a given set of dynamics, input and state constraints. Furthermore, because of the conservative nature of $\mathsf{Disc}_k(\mathcal{K}_0)$, $\mathcal{K}_\infty$ may not exist even when a true infinite horizon sampled data discriminating kernel does. However, if a $\mathcal{K}_\infty$ is found and it is possible to guarantee $x_0 \in \mathcal{K}_\infty$, then the control policy shown in figure 4 can be implemented without the need to evaluate $n(x_0)$ or store $\mathcal{K}_k$ for finite k; only $\mathcal{K}_{\text{free}}$, $\mathcal{K}_\infty$ and the control policy for $x_0 \in \mathcal{K}_\infty \setminus \mathcal{K}_{\text{free}}$ need to be stored.

5. Hamilton-Jacobi Formulation

In this section we outline how to implement the abstract algorithm above using an HJ PDE formulation of invariance kernels. The main advantages of this formulation are that general nonlinear dynamics (1) can be handled and implementation of the key operators in the algorithm are straightforward. The main disadvantage is the computational cost: exponential in the number of dimensions in which the invariance kernel is calculated. We demonstrate how the constants involved can be kept small for the control dimensions, but there is at present no way to escape the curse of dimensionality for the state space dimensions.

5.1. Preliminaries: Implicit Surface Functions and the HJ PDE

In this formulation we represent sets $\mathcal{S} \subset \mathbb{R}^d$ using an implicit surface function $\psi_{\mathcal{S}} : \mathbb{R}^d \to \mathbb{R}$ such that

$$\mathcal{S} = \{x \in \Omega \mid \psi_{\mathcal{S}}(x) \leq 0\}.$$

The implicit surface function representation is very flexible; for example, it can represent nonconvex and disconnected sets. Its main restriction is that sets must have a nonempty interior and exterior. Analytic implicit surface functions for common geometric shapes (such as spheres, hyperplanes, prisms, etc.) are easily constructed. The constructive solid geometry operations of union, intersection and complement of sets are achieved through pointwise minimum, maximum and negation operations on their implicit surface functions. For example, consider a sphere of radius two $\mathcal{S}_1 = \{x \mid \|x\|_2 \leq 2\}$, the halfspace whose boundary has outward normal vector a and passes through the origin $\mathcal{S}_2 = \{x \mid a^T x \leq 0\}$ and the hemisphere that is their intersection $\mathcal{S}_3 = \mathcal{S}_1 \cap \mathcal{S}_2$. Implicit surface representations of these sets are given by $\psi_{\mathcal{S}_1}(x) = \|x\|_2 - 2$, $\psi_{\mathcal{S}_2}(x) = +a^T x$ and $\psi_{\mathcal{S}_3}(x) = \max[\psi_{\mathcal{S}_1}(x), \psi_{\mathcal{S}_2}(x)]$.

An HJ PDE whose solution is an implicit surface function for the reachable tube of a system with adversarial inputs was proven in [17]; the adaptation to invariance kernels that we outline here is straightforward. Given a constraint set $\mathcal{S}$ represented by the known implicit surface function $\psi_{\mathcal{S}}$ and system dynamics $\dot{y} = g(y, w)$ with input $w \in \mathcal{W}$, we can determine an implicit surface function for $\mathsf{Inv}([0, \delta], \mathcal{S}, w, g)$

$$\psi_{\mathsf{Inv}([0,\delta],\mathcal{S},w,g)}(y) = \phi(y, 0),$$

where ϕ is the viscosity solution of the terminal value, time-dependent HJ PDE

$$D_t \phi + \max\left[0, H(y, D_y \phi)\right] = 0$$

with Hamiltonian

$$H(y, p) = \max_{w \in \mathcal{W}} p^T g(y, w)$$

and terminal condition

$$\phi(y, \delta) = \psi_{\mathcal{S}}(y).$$

5.2. Hamilton-Jacobi Formulation of Operators

Using properties of the implicit surface function and the HJ PDE formulation of invariance kernels described above, we can implement the operators needed to approximate the sampled data discriminating kernel.

Given implicit surface representations $\psi_{\mathcal{S}}$ and $\psi_{\mathcal{U}}$ of $\mathcal{S}$ and $\mathcal{U}$ respectively, an implicit surface representation of $\mathcal{S} \times \mathcal{U}$ is given by

$$\psi_{\mathcal{S}\times\mathcal{U}}(\tilde{x}) = \max\left(\psi_{\mathcal{S}}(x), \psi_{\mathcal{U}}(u)\right)$$

where $\tilde{x} = \begin{bmatrix} x & u \end{bmatrix}^T$. To find the implicit surface representation $\psi_{\mathsf{Inv}_1(\mathcal{S})}$ of (10) we solve

$$\begin{aligned} D_t\phi + \max\left[0, H(\tilde{x}, D_{\tilde{x}}\phi)\right] = 0 \\ H(\tilde{x}, p) = \max_{v\in\mathcal{V}} p^T \tilde{f}(\tilde{x}, v) \\ \phi(\tilde{x}, \delta) = \psi_{\mathcal{S}\times\mathcal{U}}(\tilde{x}) \\ \psi_{\mathsf{Inv}_1(\mathcal{S})}(\tilde{x}) = \phi(\tilde{x}, 0). \end{aligned} \tag{18}$$

Projecting out the u dimension to accomplish (11) is easily done

$$\psi_{\mathsf{Disc}_1(\mathcal{S})}(x) = \min_u \psi_{\mathsf{Inv}_1(\mathcal{S})}(\tilde{x}). \tag{19}$$

By (12) and (13), this sequence of pointwise maximization, HJ PDE solution and pointwise minimization can be repeated to construct implicit surface representations $\psi_{\mathcal{K}_k}$ for $k = 1, 2, \dots, N$.

Once $\psi_{\mathcal{K}_N}$ is determined, we implement (14) by solving one last HJ PDE

$$\begin{aligned} D_t\phi + \max\left[0, H(x, D_x\phi)\right] = 0 \\ H(x, p) = \max_{v\in\mathcal{V}} \max_{u\in\mathcal{U}} p^T f(x, u, v) \\ \phi(x, \delta) = \psi_{\mathcal{K}_N}(x) \\ \psi_{\mathcal{K}_{\text{free}}}(x) = \phi(x, 0). \end{aligned} \tag{20}$$

to find the implicit surface representation $\psi_{\mathcal{K}_{\text{free}}}$.

5.3. Control Policy Synthesis

For $x_0 \in \mathcal{K}_{\text{ctrl}}$, an implicit surface function for $\mathcal{U}_{\text{ctrl}}(x_0)$ in (16) can be constructed

$$\psi_{\mathcal{U}_{\text{ctrl}}(x_0)}(u) = \psi_{\mathsf{Inv}_1(\mathcal{K}_{n(x_0)-1})}(\tilde{x}_0) \tag{21}$$

where $\tilde{x}_0 = \begin{bmatrix} x_0 & u \end{bmatrix}^T$. However, there is additional quantitative information in the implicit surface functions $\psi_{\mathcal{K}_k}$ which we can take advantage of to construct alternative representations of the control policy and even alternative control policies.

For $x_0 \in \mathcal{K}_{\text{ctrl}}$, define the value at the next sample time under fixed input $\bar{u} \in \mathcal{U}$ as

$$\psi_\delta^{\bar{u}}(x_0) \triangleq \max_{v(\cdot)} \psi_{\mathcal{K}_{n(x_0)-1}}(\bar{x}(\delta)), \tag{22}$$

where $\bar{x}(\cdot)$ solves (2) with fixed input $u = \bar{u}$ and initial condition $x(0) = x_0$. If the infinite horizon discriminating kernel $\mathcal{K}_\infty$ has been discovered, then for $x_0 \in \mathcal{K}_\infty$ use the alternative definition

$$\psi_\delta^{\bar{u}}(x_0) = \psi_{\mathcal{K}_\infty}(\bar{x}(\delta)).$$

With $\psi_\delta^{\bar{u}}$ defined, the policy (21) can also be represented as

$$\mathcal{U}_{\text{ctrl}}(x_0) \triangleq \{\bar{u} \in \mathcal{U} \mid \psi_\delta^{\bar{u}}(x_0) \leq 0\}, \tag{23}$$

while two alternative policies are given by

$$\begin{aligned} \mathcal{U}_{\text{ctrl}}^{\rightarrow}(x_0) &\triangleq \{\bar{u} \in \mathcal{U} \mid \psi_\delta^{\bar{u}}(x_0) \leq \psi_{\mathcal{K}_{n(x_0)}}(x_0)\}, \\ \mathcal{U}_{\text{ctrl}}^{\searrow}(x_0) &\triangleq \operatorname*{argmin}_{\bar{u} \in \mathcal{U}} \psi_\delta^{\bar{u}}(x_0). \end{aligned} \tag{24}$$

Note that all of these policies will be set-valued in general.

Proposition 6. *For all $x_0 \in \mathcal{K}_{ctrl}$, $\mathcal{U}_{ctrl}^{\rightarrow}(x_0) \neq \emptyset$.*

Proof. The HJ PDE (18) and minimization (19) imply that $\psi_{\mathsf{Disc}_1(\mathcal{S})}$ is the value function of a finite horizon terminal value differential game problem

$$\psi_{\mathsf{Disc}_1(\mathcal{S})}(x_0) = \max_{v(\cdot)} \max_{s \in [0,\delta]} \min_{\bar{u}} \psi_{\mathcal{S}}(\bar{x}(s)), \tag{25}$$

where $v(\cdot)$ is a measurable input signal but $\bar{u}$ is a constant input. Consider $x_0 \in \mathcal{K}_{\text{ctrl}}$, and let $\bar{n} = n(x_0)$. By (12) and (13), $\psi_{\mathsf{Disc}_1(\mathcal{K}_{\bar{n}-1})}(x_0) = \psi_{\mathcal{K}_{\bar{n}}}(x_0)$; consequently, by (25) there exists $\bar{u} \in \mathcal{U}$ such that

$$\max_{v(\cdot)} \max_{s \in [0,\delta]} \psi_{\mathcal{K}_{\bar{n}-1}}(\bar{x}(s)) = \psi_{\mathcal{K}_{\bar{n}}}(x_0).$$

By (22)

$$\psi_\delta^{\bar{u}}(x_0) = \max_{v(\cdot)} \psi_{\mathcal{K}_{\bar{n}-1}}(\bar{x}(\delta)) \leq \psi_{\mathcal{K}_{\bar{n}}}(x_0);$$

therefore, $\bar{u} \in \mathcal{U}_{\text{ctrl}}^{\rightarrow}(x_0)$. $\square$

Corollary 7. *For all* $x_0 \in \mathcal{K}_{ctrl}$, $\mathcal{U}^{\searrow}_{ctrl}(x_0) \neq \emptyset$. *For all* $\bar{u} \in \mathcal{U}^{\searrow}_{ctrl}(x_0)$, $\psi^{\bar{u}}_{\delta}(x_0) \leq \psi_{\mathcal{K}_{n(x_0)}}(x)$.

Corollary 8. *For* $x_0 \in \mathcal{K}_{ctrl}$, *the following containment property holds*

$$\mathcal{U}^{\searrow}_{ctrl}(x_0) \subseteq \mathcal{U}^{\rightarrow}_{ctrl}(x_0) \subseteq \mathcal{U}_{ctrl}(x_0),$$

The intuition behind these different policies is

- The most permissive policy $\mathcal{U}_{\text{ctrl}}(x_0)$ allows any control input which will keep $\bar{x}(\delta) \in \mathcal{K}_{n(x_0)-1}$; consequently, it ensures safety over the desired horizon but permits the system to get arbitrarily close to the boundary of $\mathcal{K}_{n(x_0)-1}$.
- The intermediate policy $\mathcal{U}^{\rightarrow}_{\text{ctrl}}(x_0)$ allows any control input which will keep $\bar{x}(\delta)$ at least as far away from the boundary of $\mathcal{K}_{n(x_0)-1}$ as x_0 is from the boundary of $\mathcal{K}_{n(x_0)}$ (where the distance metric is the implicit surface functions $\psi_{\mathcal{K}_k}$).
- The most aggressive policy $\mathcal{U}^{\searrow}_{\text{ctrl}}(x_0)$ chooses the control(s) which will drive $\bar{x}(\delta)$ as deep within $\mathcal{K}_{n(x_0)-1}$ as possible.

Why use anything other than the most permissive policy $\mathcal{U}_{\text{ctrl}}(x_0)$? The sampled data discriminating kernel algorithm from section 4.2 is inherently conservative (by Proposition 2), and models with disturbance inputs v are often used to construct robust discriminating kernels even though such models are also conservative with respect to safety. Consequently, it may be possible to drive $x(\cdot)$ back into $\mathcal{K}_{\text{free}}$ using the more aggressive policies described above even if $x_0 \in \mathcal{K}_{\text{ctrl}}$.

5.4. Practical Implementation

In this section we describe a particular approach to approximating the solution of the equations above for the common case where we do not have analytic solutions to those equations.

5.4.1. Approximating the Implicit Surface Functions

We use the Toolbox of Level Set Methods (ToolboxLS) as described in [18] to manipulate implicit surface functions. Implicit surface functions are represented by values sampled at nodes on a regular orthogonal grid. When values are needed away from grid points, interpolation is used (eg: `interpn` in Matlab). Maximum and minimum operations are done pointwise at each node in the grid.

In general, HJ PDEs (18) and (20) include an input and so a Lax-Friedrichs centered difference scheme is used to approximate the respective Hamiltonians. High order of accuracy finite difference approximations of the spatial and temporal derivatives are used to evolve the equation (for example, see [26]). If (21) is used to construct the control policy in $\mathcal{K}_{\text{ctrl}}$ then only the zero level set of the solutions of the PDEs are needed and so reinitialization and/or velocity extension techniques can be applied to improve the numerical results. If (23) or (24) are used to construct the control policy, then the value of the implicit surface functions $\psi_{\mathcal{K}_k}$, and not just their zero level sets, is used via (22) for all $x_0 \in \mathcal{K}_{\text{ctrl}}$, and so reinitialization and/or velocity extension cannot be applied when solving (18).

5.4.2. Mitigating the Curse of Dimensionality

As mentioned previously, the primary weakness of this formulation is that the size of the grid needed to accurately approximate the solution of the HJ PDEs grows exponentially with dimension. Such cost is bad enough in the d_x dimensional state space, but (10) requires an invariant kernel computed in $d_x + d_u$ dimensions. Fortunately, without too much loss of accuracy those extra d_u dimensions can be treated with an arbitrarily coarse grid and each sample in those dimensions run separately, so the situation is not quite as dire as it might first appear.

When approximating the solution of an evolutionary PDE, one normally has to ensure a grid fine enough to resolve key features of the solution both in order to avoid error in those key features and also to avoid that error from destroying the accuracy of nearby features through numerical dissipation. This property holds true for the HJ PDEs above in the d_x state space dimensions, but does not apply to the d_u control input dimensions because the augmented dynamics in these dimensions are zero. A coarse sampling of the u dimensions may not capture the optimal u input value and hence may underestimate the true discriminating kernel, but it will accurately reflect the discriminating kernel for the sampled values of u. In fact, the algorithms for approximating sampled data reachability in [4, 3] can be interpreted as exactly such a coarse sampling of a reachable set calculation using the same augmented dynamics (5). To give some idea of the order of magnitude savings such a coarse sampling of u can provide, the HJ PDE based approximations in sections 5.5 and 6.5 used only 3–7 samples of u (further sampling in the u dimension had little effect on the outcome), but grids of 60–200 nodes in each of the x dimensions. Such a coarse sampling strategy can be very effective when the number of control input dimensions is low and/or the optimal samples for the control input can be guessed a priori.

Furthermore, the fact that the dynamics in the control input dimensions are zero imply that the results for separate input samples do not interact with one another during the invariant set calculation. Therefore, it is possible to run the invariant sets for each input sample separately (either serially on a single processor or in parallel on a cluster) so that the memory cost of the algorithm is exponential only in d_x. Separate runs for each input sample also ensures that there can be no numerical dissipation or issues with artificial boundary conditions in the u dimensions. Because this separated sampling approach reduces both memory cost and numerical error, we have not yet encountered any situation where it makes sense to directly approximate the HJ PDE formulation in the full augmented state space.

The coarse and separated sampling strategies described above are effective at reducing the computational cost of this formulation significantly—they made the difference between seconds and hours of computation time for the examples presented below—however, it must be admitted that they only postpone but do not overcome the scaling barrier created by the exponential growth of computational effort with respect to both state space and control input dimension for this formulation.

5.4.3. Constructing the Feedback Controller

For $x_0 \in \mathcal{K}_{\text{free}}$ the implementation is trivial. For $x_0 \in \mathcal{K}_{\text{ctrl}}$, there are two approaches to determine a set of safe control signals.

To construct an implicit surface representation of the set $\mathcal{U}_{\text{ctrl}}(x_0)$, create a grid of u values $\{u_j\}_j$ and then a grid of augmented state values $\{\tilde{x}_j\}_j$ such that $\tilde{x}_j = \begin{bmatrix} x_0 & u_j \end{bmatrix}^T$. Using numerical interpolation where necessary, evaluate $\psi_{\mathsf{Inv}_1(\mathcal{K}_{n(x_0)-1})}(\tilde{x}_j)$ on the grid $\{\tilde{x}_j\}_j$. Then (21) provides an approximation on the grid $\{u_j\}_j$ of an implicit surface function $\psi_{\mathcal{U}_{\text{ctrl}}(x_0)}(u)$ representing $\mathcal{U}_{\text{ctrl}}(x_0)$. Interpolation of $\psi_{\mathcal{U}_{\text{ctrl}}(x_0)}(u)$ (which is continuous) can be used to approximate the full set of safe inputs if the control input dimension is sufficiently well sampled.

Alternatively, again choose a set of input samples $\{u_j\}_j$ but this time compute $\psi_\delta^{\bar{u}}(x_0)$ through (22) with $\bar{u} = u_j$ for each u_j. A numerical ODE solver (eg: `ode45` in MATLAB) can be used to approximate the point $\bar{x}(\delta)$ and then numerical interpolation can provide an approximation of $\psi_{\mathcal{K}_{n(x_0)-1}}(\bar{x}(\delta))$. Either (23) or (24) can then be used to select a subset of $\{u_j\}_j$ which lie within

$\mathcal{U}_{\text{ctrl}}(x_0)$, $\mathcal{U}^{\rightarrow}_{\text{ctrl}}(x_0)$ or $\mathcal{U}^{\searrow}_{\text{ctrl}}(x_0)$ as desired. Interpolation might also be needed to approximate $\psi_{\mathcal{K}_{n(x_0)}}(x_0)$ if $\mathcal{U}^{\rightarrow}_{\text{ctrl}}(x_0)$ is being used.

5.4.4. Guaranteeing an Underapproximation

The combination of the algorithm from section 4 and the analytic HJ PDE formulation of the operators from section 5.2 guarantees safety, but the numerical implementation described above does not maintain that guarantee. The decision to use an unsound implementation was primarily driven by convenience, and also the empirical accuracy that the level set schemes have demonstrated in the past.

It is possible to use sound numerical implementations such as those described in [6] for the required invariance kernel calculations. These implementations use an indicator-like representation of sets, so it might not be possible to directly extract the control policies (24) but there are several approaches to reformulate HJ PDEs as viability kernels if necessary. The primary shortcoming of these sound algorithms is their relative inaccuracy when compared to the schemes implemented in ToolboxLS. It is possible that a combination of the two approaches might be able to achieve both sound and accurate approximations.

5.5. Example

Computations were done on an Intel Core2 Duo at 1.87 GHz with 4 GB RAM running 64-bit Windows 7 Professional (Service Pack 1), 64-bit Matlab version 7.11 (R2010b), and ToolboxLS version 1.1. Matlab code can be found at the first author's web site `http://www.cs.ubc.ca/~mitchell`

We demonstrate the algorithms using an envelope protection problem for a variation on the double integrator because it is much easier to visualize results in two dimensions. In the standard double integrator, once deceleration begins the optimal control stays constant until the system stops no matter what the state; consequently, the results are very similar in a sampled data environment to what they would be in a continuous time environment. Instead, we modify the double integrator so that the optimal choice of input depends on state (a "spatially varying double integrator"). The dynamics are given by

$$\dot{x} = \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \cos(2\pi x_1)u \end{bmatrix} = f(x,u)$$

with $|u| \leq 1$. Note that the effect of the input varies considerably over the domain, and the sign of the optimal input will switch every 0.5 units in the x_1

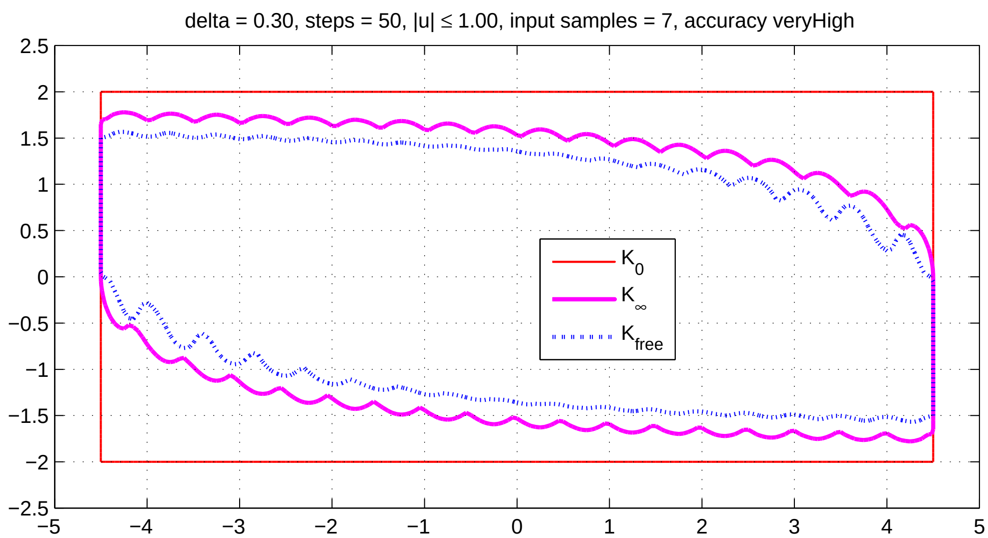

Figure 5: The partition of Ω for the spatially varying double integrator with $\delta = 0.3$ and horizon $N = 50$ (eg: $T = 15$, long enough for convergence). The state constraint $\mathcal{K}_0$ is the outermost thin red rectangle, $\mathcal{K}_\infty$ is inside the thick magenta contour, $\mathcal{K}_{\text{ctrl}}$ is *outside* the dotted blue contour, and $\mathcal{K}_{\text{free}}$ is inside that innermost contour (where the legend is).

direction. The constraint set is a rectangle $\mathcal{K}_0 = [-4.5, +4.5] \times [-2.0, +2.0]$. For the sampled data problem, we choose $\delta = 0.3$ and $N = 50$ (which is empirically sufficient time for convergence). As discussed in section 5.4.2, we choose a coarse sampling of the input set

$$\{u_j\}_{j=1}^{7} = \left\{-1, -\tfrac{2}{3}, -\tfrac{1}{3}, 0, +\tfrac{1}{3}, +\tfrac{2}{3}, +1\right\}.$$

Figure 5 shows the results for the above parameters. They were calculated on a state space grid of size 201×101 using a fifth order accurate spatial and a third order accurate temporal derivative approximation. Figure 6 shows results for the continuous time version, and also for versions with $\delta = 0.1$ and $\delta = 1.0$. Notice that the continuous time version has a much larger $\mathcal{K}_{\text{free}}$ because it can always choose an input that generates deceleration. Furthermore, $\mathcal{K}_{\text{ctrl}} = \mathcal{K}_\infty$ in this case, because $\delta = 0$. In contrast, as δ becomes large the envelope becomes increasingly uncontrollable.

Figures 7 and 8 show some sample trajectories generated using the policy (17) with $\mathcal{U}_{\text{ctrl}}^{\rightarrow}$ and $\mathcal{U}_{\text{ctrl}}^{\searrow}$ respectively. For illustrative purposes the control was chosen to drive the trajectory back toward $\mathcal{K}_{\text{ctrl}}$ for $x \in \mathcal{K}_{\text{free}}$, and was chosen for $\mathcal{U}_{\text{ctrl}}^{\rightarrow}$ to keep the trajectory as deeply within $\mathcal{K}_{\text{ctrl}}$ as possible, but

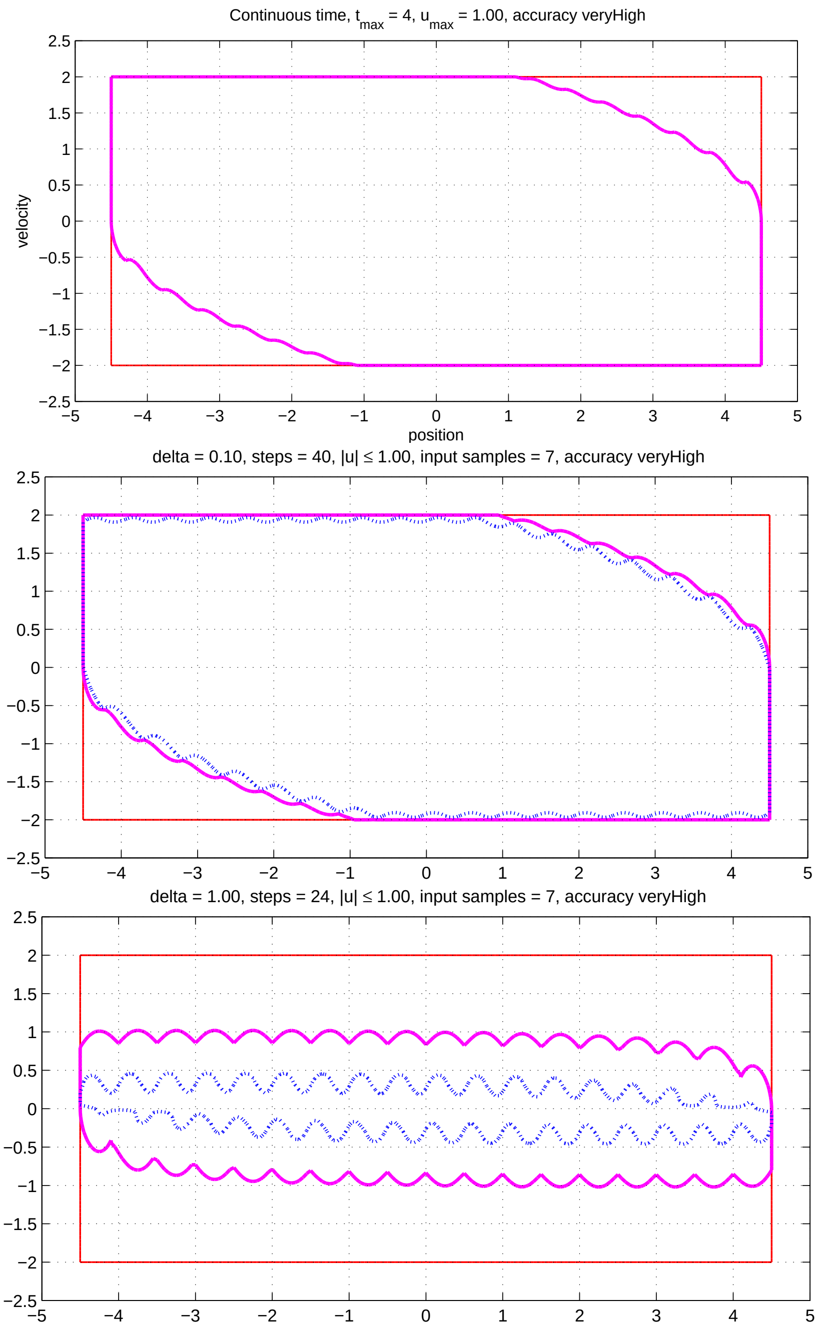

Figure 6: The effect of δ on the spatial partition. Top: Traditional reachability with continuous state feedback and measurable control signals ($T = 4$). Middle: Sampled data with $\delta = 0.1$, $N = 40$ ($T = 4$). Bottom: Sampled data with $\delta = 1.0$, $N = 24$ ($T = 24$).

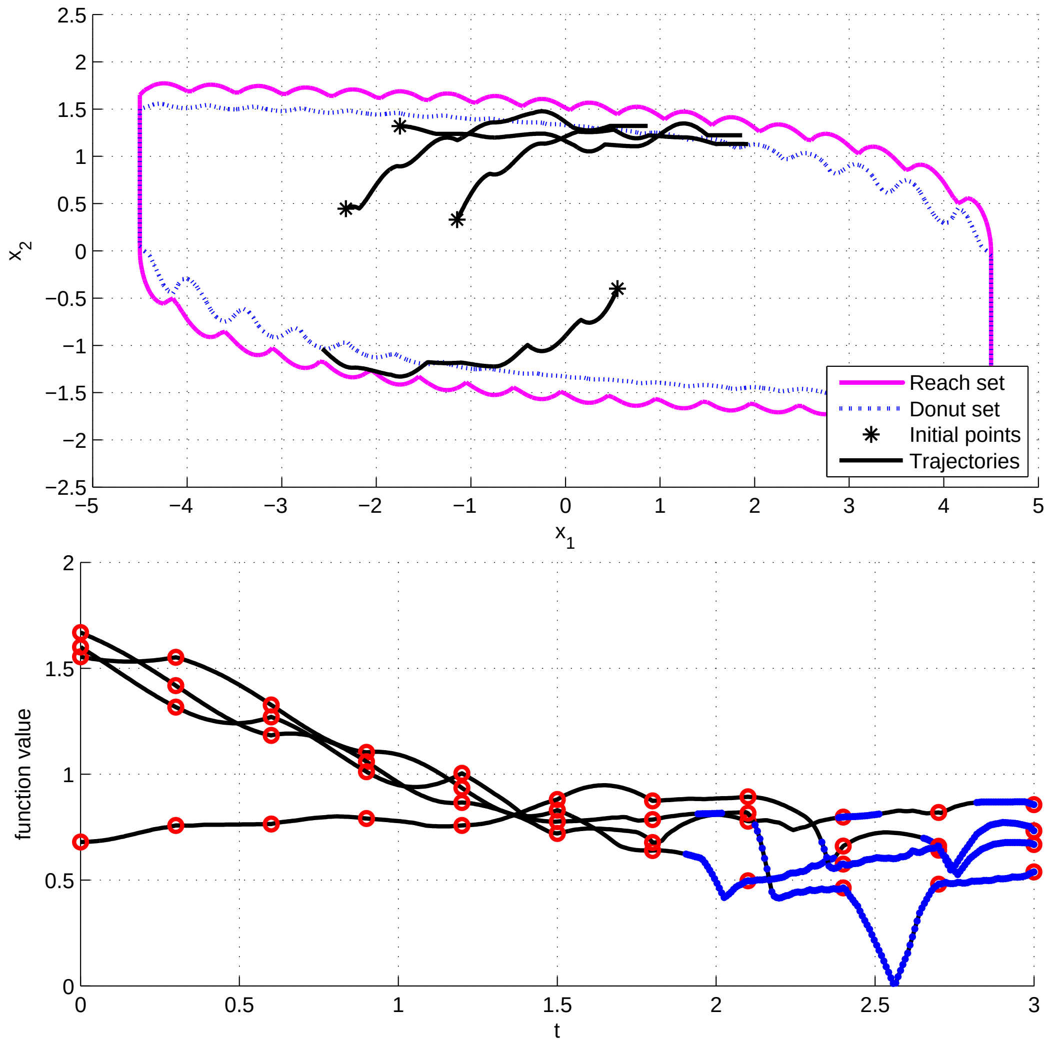

Figure 7: Sample trajectories using the intermediate safe policy $\mathcal{U}^{\rightarrow}_{\mathrm{ctrl}}$ for $\delta = 0.3$. Top: Trajectories $x(\cdot)$ in phase space overlaid on the state space partition. Bottom row: $\psi_{\mathcal{K}_\infty}(x(t))$ versus t. Sample times are shown as red circles, and periods during which $x(t) \in \mathcal{K}_{\mathrm{ctrl}}$ are shown with blue dots.

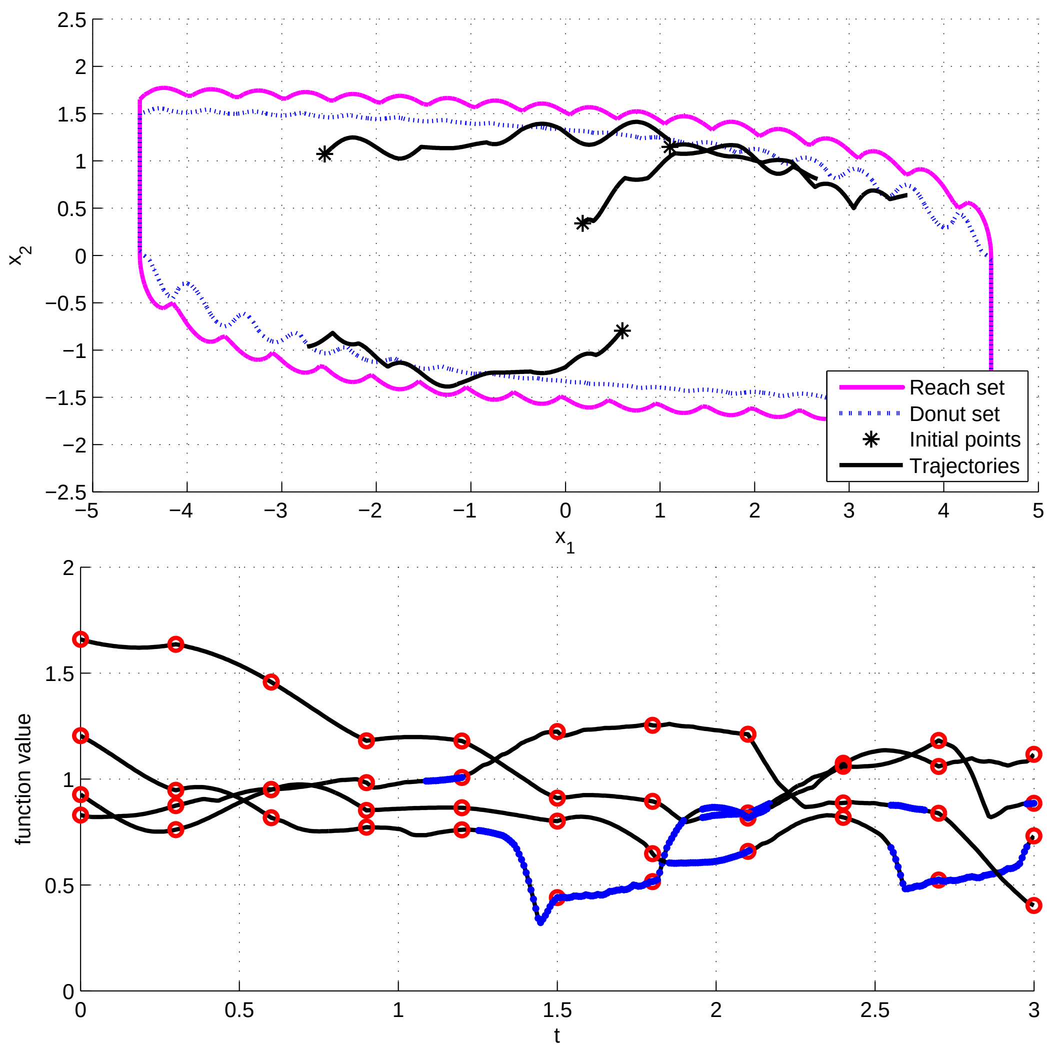

Figure 8: Sample trajectories using the aggressive safe policy $\mathcal{U}_{\mathrm{ctrl}}^{\searrow}$ for $\delta = 0.3$. Top: Trajectories $x(\cdot)$ in phase space overlaid on the state space partition. Bottom row: $\psi_{\mathcal{K}_\infty}(x(t))$ versus t. Sample times are shown as red circles, and periods during which $x(t) \in \mathcal{K}_{\mathrm{ctrl}}$ are shown with blue dots.

other choices are available. In the bottom of each plot, notice that the value of $\psi_{\mathcal{K}_\infty}$ may decrease along a trajectory between samples, but if the trajectory is in $\mathcal{K}_{\text{ctrl}}$ (as indicated by the blue dots) at the sample time, then the value of $\psi_{\mathcal{K}_\infty}$ does not decrease at the subsequent sample time.

6. Ellipsoidal Formulation

In this section we outline how to implement the abstract algorithm from section 4 using an ellipsoidal formulation of invariance kernels. The main advantage of this formulation is computational cost: polynomial (roughly cubic) in the number of dimensions in which the invariance kernel is calculated. The main disadvantages are a restriction to linear dynamics, reduced accuracy because ellipsoidal underapproximations must be used at several steps in the algorithm, and some additional intermediate steps which make the formulation of the key operators more complicated.

6.1. Preliminaries: Ellipsoidal Complications

Let $\mathrm{P} \in \mathbb{R}^{d_1 \times d_2}$ with $d_1 \leq d_2$ be a matrix such that $\mathrm{P}^T\mathrm{P}$ is a projection matrix (so $(\mathrm{P}^T\mathrm{P})^2 = \mathrm{P}^T\mathrm{P}$). In particular, we will use block matrices

$$\mathrm{P}_x = \begin{bmatrix} \mathrm{I}_{d_x} & 0_{d_x \times d_u} \end{bmatrix} \qquad \text{and} \qquad \mathrm{P}_u = \begin{bmatrix} 0_{d_u \times d_x} & \mathrm{I}_{d_u} \end{bmatrix}$$

where $I_d \in \mathbb{R}^{d \times d}$ is an identity matrix and $0_{d_1 \times d_2} \in \mathbb{R}^{d_1 \times d_2}$ is a zero matrix. Given an augmented state $\tilde{x} = \begin{bmatrix} x & u \end{bmatrix}^T$, we then have that $\mathrm{P}_x \tilde{x} = x$ and $\mathrm{P}_u \tilde{x} = u$. More generally, we could choose P such that the rows form an orthonormal basis for a subspace into which we want to project a vector.

6.1.1. Preliminaries: Ellipsoids

An ellipsoid in $\mathbb{R}^d$ is defined by

$$\begin{aligned} \mathcal{E}(q, \mathrm{H}) &\triangleq \{\mathrm{H}y + q \in \mathbb{R}^d \mid \|y\|_2 \leq 1\} \\ &= \{y \in \mathbb{R}^d \mid (y - q)^T \mathrm{H}^{-2} (y - q) \leq 1\} \end{aligned}$$

where $q \in \mathbb{R}^d$ is the center, $\mathrm{H} = \mathrm{H}^T \in \mathbb{R}^{d \times d}$, and $\mathrm{H}\mathrm{H}^T = \mathrm{H}^2$ is the symmetric positive definite shape matrix. For matrix A, the linear mapping of an ellipsoid is also an ellipsoid

$$\mathrm{A}\mathcal{E}(q, \mathrm{H}) = \mathcal{E}(\mathrm{A}q, \mathrm{A}\mathrm{H})$$

We will call a finite union of ellipsoids a piecewise ellipsoidal set.

Many of the sets $\mathcal{S}$ involved in the algorithm below will not be ellipsoidal, so where necessary we will construct ellipsoidal approximations $\mathcal{E}_{\mathcal{S}}$. An "ellipsoidal approximation" of a set is not a unique object, but in this algorithm it will typically be an underapproximation, it will often be a maximum volume underapproximation, and the particular choice for each such approximation in the algorithm should be clear from context.

6.1.2. Preliminaries: Maximum Volume Inscribed Ellipsoids

It is well known that given a collection of nonempty compact ellipsoids $\{\mathcal{Y}_i\}$, their intersection $\cap_i \mathcal{Y}_i$ is not in general an ellipsoid but it can be easily underapproximated by one: The maximum volume inscribed ellipsoid $\mathcal{E}_{\cap_i \mathcal{Y}_i}$ can be determined by solving a convex semi-definite program [27]. Here we slightly extend the technique to allow sets $\mathcal{Y}_i$ which can be either an ellipsoid $\mathcal{Y}_i = \mathcal{E}(q_i, \mathrm{H}_i)$ or the tensor product of lower dimensional ellipsoids

$$\mathcal{Y}_i = \mathcal{Y}_{i,x} \times \mathcal{Y}_{i,u}$$
$$\text{where } \mathcal{Y}_{i,x} \triangleq \mathcal{E}(q_{i,x}, \mathrm{H}_{i,x}) \subset \mathbb{R}^{d_x} \text{ and } \mathcal{Y}_{i,u} \triangleq \mathcal{E}(q_{i,u}, \mathrm{H}_{i,u}) \subset \mathbb{R}^{d_u}.$$

For notational simplicity we have assumed that the lower dimensional ellipsoids happen to be in the x and u subspaces of the augmented state space $\tilde{x}$, although the formulation can easily be generalized to allow different subspaces and/or the tensor product(s) of more than two lower dimensional ellipsoids.

We will also modify the objective of the optimization to find the inscribed ellipsoid whose volume is maximal in a subspace projection given by some $\bar{\mathrm{P}}$. Choosing $\bar{\mathrm{P}} = \mathrm{I}$ will generate the maximum volume inscribed ellipsoid as normal. Choosing $\bar{\mathrm{P}} = \mathrm{P}_x$ will find the inscribed ellipsoid whose volume is maximal in the x subspace.

If $\cap_i \mathcal{Y}_i \neq \emptyset$, solve the semidefinite program (SDP)

$$\begin{aligned} &\text{minimize} - \log\det \bar{\mathrm{P}}\bar{\mathrm{H}}\bar{\mathrm{P}}^T \\ &\quad \text{over } \bar{\mathrm{H}} \in \mathbb{R}^{d\times d},\ \bar{q} \in \mathbb{R}^d,\ \text{and } \lambda_i \in \mathbb{R} \end{aligned} \tag{26}$$

subject to constraints for $i = 1, 2, \ldots$ either of the form

$$\begin{gathered} \lambda_i > 0 \\ \begin{bmatrix} 1-\lambda_i & 0 & (\bar{q}-q_i)^T \\ 0 & \lambda_i \mathrm{I} & \bar{\mathrm{H}} \\ (\bar{q}-q_i) & \bar{\mathrm{H}} & \mathrm{H}_i^2 \end{bmatrix} \geq 0, \end{gathered} \tag{27}$$

if $\mathcal{Y}_i = \mathcal{E}(q_i, \mathrm{H}_i)$ or of the form

$$\begin{aligned}
\lambda_{i,x} &> 0 \\
\lambda_{i,u} &> 0 \\
\begin{bmatrix} 1-\lambda_{i,x} & 0 & (\mathrm{P}_x\bar{q} - q_{i,x})^T \\ 0 & \lambda_{i,x}\mathrm{I} & \mathrm{P}_x\bar{\mathrm{H}}\mathrm{P}_x^T \\ (\mathrm{P}_x\bar{q} - q_{i,x}) & \mathrm{P}_x\bar{\mathrm{H}}\mathrm{P}_x^T & \mathrm{H}_{i,x}^2 \end{bmatrix} &\geq 0 \\
\begin{bmatrix} 1-\lambda_{i,u} & 0 & (\mathrm{P}_u\bar{q} - q_{i,u})^T \\ 0 & \lambda_{i,u}\mathrm{I} & \mathrm{P}_u\bar{\mathrm{H}}\mathrm{P}_u^T \\ (\mathrm{P}_u\bar{q} - q_{i,u}) & \mathrm{P}_u\bar{\mathrm{H}}\mathrm{P}_u^T & \mathrm{H}_{i,u}^2 \end{bmatrix} &\geq 0
\end{aligned} \tag{28}$$

if $\mathcal{Y}_i = \mathcal{Y}_{i,x} \times \mathcal{Y}_{i,u}$, where I and 0 are appropriately sized identity and zero matrices. The optimal values $\bar{\mathrm{H}}^*$ and $\bar{q}^*$ define the inscribed ellipsoid with maximum volume in the $\bar{\mathrm{P}}$ subspace:

$$\mathsf{Inscribed}_{\bar{\mathrm{P}}}\left(\cap_i \mathcal{Y}_i\right) \triangleq \mathcal{E}\left(\bar{q}^*, \bar{\mathrm{H}}^*\right).$$

We will use this operator several times in the algorithm below.

6.1.3. Preliminaries: Ellipsoidal Underapproximation of Invariance Kernels

For the implicit surface function representations used in the previous section, there was an HJ PDE whose solution governed their evolution. The situation is more complicated for the ellipsoidal representation: We will construct invariance kernels by a sequence of reachability and intersection operations.

To start with we must restrict the dynamics (1) and (2) to the forms

$$\dot{x}(t) = \mathrm{A}x(t) + \mathrm{B}u(t) + \mathrm{G}v(t) \tag{29}$$

$$\dot{x}(t) = \mathrm{A}x(t) + \mathrm{B}u_{\mathrm{pw}}(t) + \mathrm{G}v(t) \tag{30}$$

respectively, where $\mathrm{A} \in \mathbb{R}^{d_x \times d_x}$, $\mathrm{B} \in \mathbb{R}^{d_x \times d_u}$ and $\mathrm{G} \in \mathbb{R}^{d_x \times d_v}$ are constant matrices.

For a target set $\mathcal{S} \subseteq \mathbb{R}^d$ and time t, define the minimal forward reach set as

$$\mathsf{Reach}(t, \mathcal{S}) \triangleq \{y(t) \in \Omega \mid \forall w(\cdot), y_0 \in \mathcal{S}\}$$

where $y(\cdot)$ solves $\dot{y} = g(y, w)$ with initial condition $y(0) = y_0$ and $w(\cdot)$ is a measurable input function such that $w(t) \in \mathcal{W}$. If g is linear and both $\mathcal{S} = \mathcal{E}_{\mathcal{S}}$ and $\mathcal{W} = \mathcal{E}_{\mathcal{W}}$ are ellipsoidal, it is possible to construct an ellipsoidal underapproximation $\mathcal{E}_{\mathsf{Reach}(t,\mathcal{S})}(\ell) \subseteq \mathsf{Reach}(t,\mathcal{S})$ for a given vector $\ell \in \mathbb{R}^d$ [28, 29, 30]. More generally, $\cup_i \mathcal{E}_{\mathsf{Reach}(t,\mathcal{S})}(\ell_i)$ for some set of vectors $\{\ell_i\}$ can be used as a piecewise ellipsoidal underapproximation of $\mathsf{Reach}(t,\mathcal{S})$.

In [31] we presented an algorithm to underapproximate continuous time viability kernels using these ellipsoidal reachability constructs, and in [32, 33] we extended this algorithm to discriminating kernels for systems with adversarial inputs. Here we briefly outline how to simplify the latter to approximate invariance kernels. For linear dynamics g, ellipsoidal $\mathcal{S} = \mathcal{E}_{\mathcal{S}}$ and $\mathcal{W} = \mathcal{E}_{\mathcal{W}}$, and vector ℓ, the algorithm creates an ellipsoidal underapproximation $\mathcal{E}_{\mathsf{Inv}([0,\delta],\mathcal{S},w,g)}(\ell)$ using a series of substeps. Start by choosing the number of substeps $\hat{n} > 0$ and the substep length $\hat{\delta} = \delta/\hat{n}$. If necessary, erode $\mathcal{S}$ to keep trajectories safe during the substeps (several approaches to such erosion are detailed in [34, pp. 94–97]). Then compute the sequence $\hat{\mathcal{E}}_{\mathcal{S}}^{(\hat{k})}(\ell)$ for $\hat{k} = 0, 1, \ldots, \hat{n}$ where

$$\begin{aligned}
\hat{\mathcal{E}}_{\mathcal{S}}^{(0)}(\ell) &= \begin{cases} \mathcal{E}_{(\text{eroded } \mathcal{S})}, & \text{if erosion was necessary;} \\ \mathcal{E}_{\mathcal{S}}, & \text{otherwise;} \end{cases} \\
\hat{\mathcal{E}}_{\mathcal{S}}^{(\hat{k}+1)}(\ell) &= \mathsf{Inscribed}_{\bar{\mathrm{P}}}\left(\hat{\mathcal{E}}_{\mathcal{S}}^{(0)}(\ell) \cap \mathcal{E}_{\mathsf{Reach}(\hat{\delta},\hat{\mathcal{E}}_{\mathcal{S}}^{(\hat{k})})}(\ell) \right) \\
\mathcal{E}_{\mathsf{Inv}([0,\delta],\mathcal{S},w,g)}(\ell) &= \hat{\mathcal{E}}_{\mathcal{S}}^{(\hat{n})}(\ell)
\end{aligned} \tag{31}$$

6.2. Ellipsoidal Formulation of Operators

Using the maximum volume inscribed ellipsoid and ellipsoidal invariance kernel algorithms described above, we can implement the operators needed to approximate the sampled data discriminating kernel.

Given ellipsoidal $\mathcal{S} = \mathcal{E}_{\mathcal{S}}$ and $\mathcal{U} = \mathcal{E}_{\mathcal{U}}$, we use the SDP (26)–(28) to construct

$$\mathcal{E}_{\mathcal{S}\times\mathcal{U}} = \mathsf{Inscribed}_{\mathrm{I}}\left(\mathcal{S} \times \mathcal{U}\right).$$

To find an ellipsoidal underapproximation of $\mathsf{Inv}_1(\mathcal{S})$ from (10), choose $\ell \in \mathbb{R}^{d_x+d_u}$ and run the iteration (31) for $\mathsf{Inv}([0,\delta], \mathcal{E}_{\mathcal{S}\times\mathcal{U}}, v, \tilde{f})$ where $\tilde{f}$ is the obvious restriction of (5) to the linear case (29). Given the result

$$\mathcal{E}_{\mathsf{Inv}([0,\delta],\mathcal{E}_{\mathcal{S}\times\mathcal{U}},v,\tilde{f})}(\ell) = \mathcal{E}_{\mathsf{Inv}_1(\mathcal{S})}(\ell)$$

of that iteration, projecting out the u dimension to accomplish (11) is a simple projection operation

$$\mathcal{E}_{\mathsf{Disc}_1(\mathcal{S})}(\ell) = \mathsf{Proj}_x(\mathcal{E}_{\mathsf{Inv}_1(\mathcal{S})}(\ell)) = \mathrm{P}_x \mathcal{E}_{\mathsf{Inv}_1(\mathcal{S})}(\ell)$$

By (12) and (13), this sequence of ellipsoid inscribed tensor product, ellipsoidal invariance kernel and projection can be repeated to construct ellipsoidal underapproximations $\mathcal{E}_{\mathcal{K}_k}(\ell)$ for $k = 1, 2, \ldots, N$ for a single direction ℓ, and then repeated for additional directions if desired.

Once $\mathcal{E}_{\mathcal{K}_N}(\ell)$ is determined, one more ellipsoidal invariance kernel calculation implements (14): run iteration (31) for $\mathsf{Inv}([0, \delta], \mathcal{E}_{\mathcal{K}_N}, (u, v), f)$ to create underapproximation

$$\mathcal{E}_{\mathcal{K}_{\text{free}}}(\ell) = \mathcal{E}_{\mathsf{Inv}([0,\delta],\mathcal{E}_{\mathcal{K}_N},(u,v),f)}(\ell).$$

6.3. Control Policy Synthesis

For $x_0 \in \mathcal{K}_{\text{ctrl}}$, let

$$\mathcal{E}_{\mathsf{Inv}_1(\mathcal{K}_{n(x_0)-1})}(\ell) = \mathcal{E}\left(\begin{bmatrix}\bar{q}_x \\ \bar{q}_u\end{bmatrix}, \begin{bmatrix}\bar{\mathrm{H}}_{xx} & \bar{\mathrm{H}}_{xu} \\ \bar{\mathrm{H}}_{ux} & \bar{\mathrm{H}}_{uu}\end{bmatrix}\right).$$

Then an ellipsoidal representation of $\mathcal{U}_{\text{ctrl}}(x_0)$ is given by

$$\begin{aligned}
\mathcal{E}_{\mathcal{U}_{\text{ctrl}}(x_0)}(\ell) &= \left\{ \mathrm{P}_u \left(\begin{bmatrix}\bar{\mathrm{H}}_{xx} & \bar{\mathrm{H}}_{xu} \\ \bar{\mathrm{H}}_{ux} & \bar{\mathrm{H}}_{uu}\end{bmatrix} \begin{bmatrix} x_0 \\ u \end{bmatrix} + \begin{bmatrix}\bar{q}_x \\ \bar{q}_u\end{bmatrix} \right) \;\middle|\; \left\| \begin{bmatrix} x_0 \\ u \end{bmatrix} \right\|_2^2 \le 1 \right\} \\
&= \left\{ \bar{\mathrm{H}}_{uu} u + (\bar{q}_u + \bar{\mathrm{H}}_{ux} x_0) \mid \|u\|_2^2 \le 1 - \|x_0\|_2^2 \right\} \\
&= \mathcal{E}\left(\bar{q}_u + \bar{\mathrm{H}}_{ux} x_0, (1 - \|x_0\|_2^2)^{-\frac{1}{2}} \bar{\mathrm{H}}_{uu} \right)
\end{aligned} \tag{32}$$

6.4. Practical Implementation

We use the Ellipsoidal Toolbox (ET) [20] to implement $\mathcal{E}_{\mathsf{Reach}(t,\mathcal{K})}(\ell)$ and YALMIP [35] to implement the SDPs. Both packages use standard double precision floating point arithmetic operations; consequently, it is possible that roundoff error may cause failure of the underapproximation guarantees that the algorithms described above provide in exact arithmetic. In practice we have not had problems as long as the ellipsoids do not get exceedingly eccentric.

When using (31) to approximate $\mathcal{K}_k$, it is necessary to erode $\mathcal{K}_0$ before computing $\mathcal{E}_{\mathcal{K}_1}$, but it is not necessary to erode $\mathcal{K}_k$ (or its ellipsoidal underapproximation) before computing $\mathcal{E}_{\mathcal{K}_{k+1}}$ for $k \ge 1$. By eroding $\mathcal{K}_0$ before running the iteration (31), we ensure that trajectories cannot exit and reenter $\mathcal{K}_0$ during the substeps of length $\hat{\delta}$ used by the reach set computation. Without erosion, trajectories in subsequent outer steps $k \ge 1$ can exit $\mathcal{K}_k$

during a substep. However, they cannot exit $\mathcal{K}_0$ since $\mathcal{K}_k \subseteq \mathcal{K}_1$ and $\mathcal{K}_1$ does not contain any states giving rise to trajectories which exit $\mathcal{K}_0$ even during the substeps (because we used erosion before computing $\mathcal{K}_1$). Therefore, even if trajectories do exit and reenter $\mathcal{K}_k$ during the reachability substeps, they remain inside $\mathcal{K}_0$ and hence safe during the outer step, and by construction they finish the outer step within $\mathcal{K}_k$.

Furthermore, when computing $\mathsf{Inv}_1(\mathcal{K}_0) = \mathsf{Inv}([0,\delta], \mathcal{E}_{\mathcal{K}_0 \times \mathcal{U}}, v, \tilde{f})$ to find $\mathcal{K}_1$, we erode $\mathcal{K}_0$ before determining $\mathcal{E}_{\mathcal{K}_0 \times \mathcal{U}}$—rather than eroding $\mathcal{E}_{\mathcal{K}_0 \times \mathcal{U}}$ directly—because the dynamics for the u dimension in $\tilde{f}$ are zero, and so no erosion in those dimensions is required to ensure safety of trajectories during the substeps.

Obvious choices for the projection operator $\bar{\mathrm{P}}$ in (31) are the identity I or the projection into the x dimensions P_x. Not surprisingly, the latter tends to generate a $\hat{\mathcal{E}}_{\mathcal{S}}^{(\hat{k}+1)}(\ell)$ at each substep whose projection into the x dimensions is somewhat larger but whose extent in the u dimensions is significantly smaller. However, our goal is to maximize the size of the eventual invariance kernel at the end of all of the substeps, and in our experiments no clear winner according to this metric has emerged. The example given below used $\bar{\mathrm{P}} = \mathrm{I}$, and we will continue to investigate these alternatives in future work.

In order to avoid additional notational complexity, the formulation above focused on the case of only a single direction vector ℓ. More generally, the algorithm can be repeated for a set of direction vectors $\{\ell_i\}$ and the results used to construct piecewise ellipsoidal underapproximations

$$\cup_i \mathcal{E}_{\mathcal{K}_k}(\ell_i) \subseteq \mathcal{K}_k \quad \text{and} \quad \cup_i \, \mathcal{E}_{\mathcal{K}_{\text{free}}}(\ell_i) \subseteq \mathcal{K}_{\text{free}}.$$

Details regarding control synthesis from piecewise ellipsoidal approximations can be found in [32, 33]. The main complication is that to extract a control policy for $x \in \mathcal{K}_{\text{ctrl}}$ from these piecewise ellipsoidal representations, the definition of $\mathcal{E}_{\mathcal{U}_{\text{ctrl}}(x)}(\ell)$ in (32) must use an ℓ_i corresponding to an ellipsoid containing x. The example given below uses only a single direction vector in order to avoid these additional complications; however, the choice of direction vector did not seem to significantly affect the final kernel approximation in this particular case.

As explained in section 6.2, there are several steps in the algorithm where a maximum volume inscribed ellipsoid is constructed. Such approximations necessarily reduce accuracy (albeit in a conservative manner) and almost certainly remove any chance that the resulting approximation of the kernel is

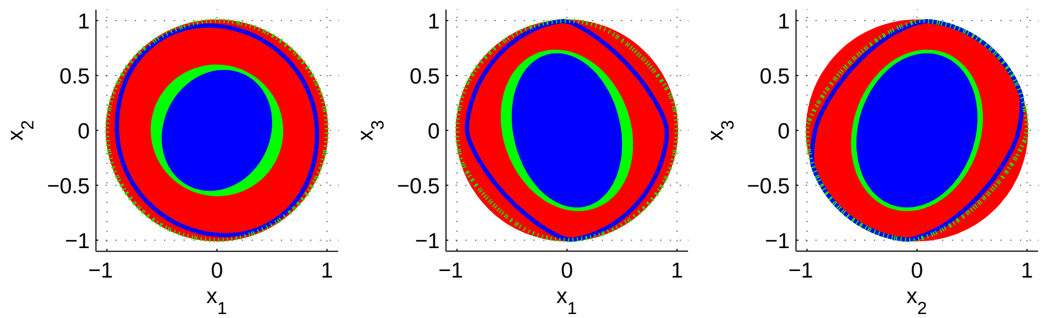

Figure 9: Projections of the partition of Ω into various pairs of state variables for the oscillating double integrator with $\delta = 0.1$ and $N = 30$. The outermost solid circle is $\mathcal{K}_0$. The innermost solid ellipse is $\mathcal{E}_{\mathcal{K}_{\text{free}}}$, which is an underapproximation of the true $\mathcal{K}_{\text{free}}$ shown by a solid contour. The light green solid ellipse in the middle is $\mathcal{E}_{\mathcal{K}_N}$, which is an underapproximation of the true $\mathcal{K}_N$ shown by the dotted light green contour. The ellipsoidal underapproximations $\mathcal{E}_{\mathcal{K}_{\text{free}}}$ and $\mathcal{E}_{\mathcal{K}_N}$ were computed using a single direction vector ℓ. The true sets $\mathcal{K}_{\text{free}}$ and $\mathcal{K}_N$ (the contours) were approximated by the HJ PDE formulation described in section 5.

tight. In particular, we have found that the underapproximating ellipsoid for a given direction vector can become degenerate and hence empty even if the true sampled data discriminating kernel is nonempty. We are investigating approaches to determine emptiness of the sampled data discriminating kernel conclusively, but at present we just try additional direction vectors in the hopes of constructively demonstrating nonemptiness.

6.5. Example

We illustrate the algorithm using another variation of the double integrator: dynamics (29) with

$$\mathrm{A} = \begin{bmatrix} 0 & -10 & 0 \\ +10 & 0 & 0 \\ +2 & +2 & 0 \end{bmatrix} \qquad \mathrm{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

and $\mathcal{U} = [-1, +1]$. Intuitively, the first two components of x provide an oscillating "velocity" so that the optimal input u varies rapidly with time along trajectories. The constraint set $\mathcal{K}_0$ is the unit ball. For $\delta = 0.1$, $N = 30$ and a single direction vector $\ell = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T$, figure 9 shows approximations of $\mathcal{K}_N$ and $\mathcal{K}_{\text{free}}$ as computed by both the HJ PDE based approach from section 5 and the ellipsoidal approach from this section. The

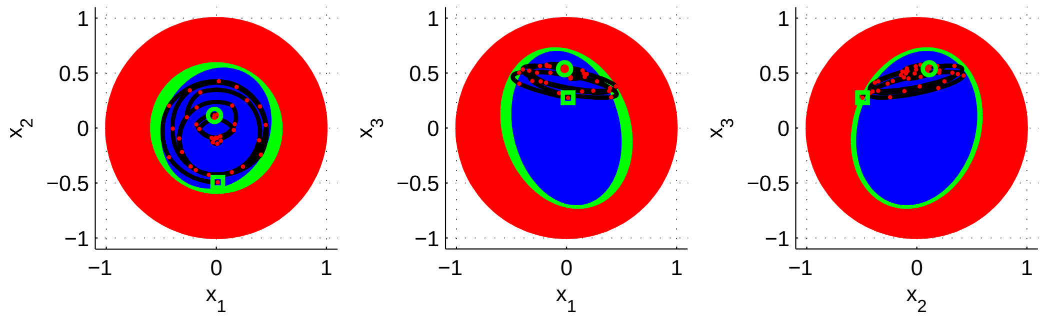

Figure 10: Projections of a trajectory for the oscillating double integrator with $\delta = 0.1$ for $0 \leq t \leq 4$, overlaid on the state space partition from figure 9. The initial condition is shown with a light green circle, the final state by a light green square, and intermediate sample times by red dots.

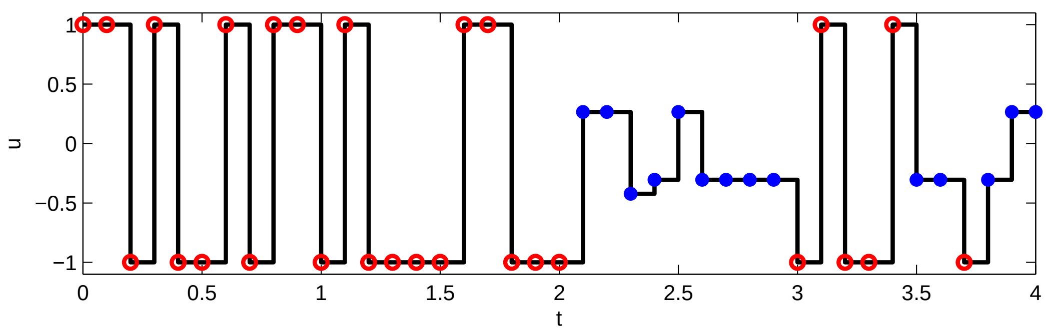

Figure 11: Control signal for the trajectory shown in figure 10. Circles mark sample times. Controls chosen in $\mathcal{U}$ are shown with open red circles, while those chosen in $\mathcal{E}_{\mathcal{U}_{\mathrm{ctrl}}(x)}$ are shown with closed blue circles.

HJ PDE based approximations are more accurate, but their cost would scale exponentially with additional state space dimensions, while the ellipsoidal approximation's cost is roughly cubic in state space dimension.

Projections of a sample trajectory $x(\cdot)$ are shown in figure 10, and the control signal $u_{\mathrm{pw}}(\cdot)$ used to generate this trajectory is shown in figure 11. In this example both $\mathcal{U}$ and $\mathcal{E}_{\mathcal{U}_{\mathrm{ctrl}}(x)}$ are always an interval (the latter possibly degenerate). The control signal in figure 11 was generated by randomly choosing one of the endpoints of the interval $\mathcal{U}$ (if $x(t_k) \in \mathcal{E}_{\mathcal{K}_{\mathrm{free}}}$) or $\mathcal{E}_{\mathcal{U}_{\mathrm{ctrl}}(x(t_k))}$ (if $x(t_k) \in \mathcal{E}_{\mathcal{K}_{\mathrm{ctrl}}}$) at each sample time t_k. Although the state space partition was constructed with finite horizon $N = 30$ (corresponding to $t = 3$), the trajectory clearly stays well within $\mathcal{K}_0$ out to $t = 4$ (the final time shown in figures 10 and 11); in fact, in our simulations it stayed within $\mathcal{K}_0$ for all times that we tried.

7. Conclusions and Future Work

We have generalized the sampled data reachability algorithm described in [3, 4] to discriminating kernels with an abstract algorithm that does not depend on Hamilton-Jacobi equations but rather works in an augmented state space with a sequence of tensor products, invariance kernels and projections. We proved that this abstract algorithm can conservatively approximate the sampled data discriminating kernel. Using this kernel, we can synthesize a permissive but safe hybrid control policy—it allows as large a set of controls as possible at every state in the constraint set while still maintaining that constraint whenever possible. Two concrete versions of the algorithm were then demonstrated: one using Hamilton-Jacobi equations which can handle nonlinear dynamics but scales poorly with state space dimension, and another using ellipsoidal reachability which scales polynomially with state space dimension but requires linear dynamics and is less accurate.

In the future we plan to apply these control synthesis algorithms to more complex, higher dimensional, and hybrid systems. Our long-term goal is to use the set-valued control policies to tackle collaborative and multi-objective control problems while still providing safety guarantees.

8. Role of the Funding Source

This research was supported by the National Science and Engineering Council of Canada (NSERC) Discovery Grants #298211 (Mitchell) & #327387 (Oishi), an NSERC Undergraduate Student Research Award (Chen), the Canadian Foundation for Innovation (CFI) Leaders Opportunity Fund / British Columbia Knowledge Development Fund Grant #13113, and CanWheel, the Canadian Institutes of Health Research (CIHR) Emerging Team in Wheeled Mobility for Older Adults #AMG-100925.

The funding agencies had no role in study design; in the collection, analysis, and interpretation of data; in the writing of the report; or in the decision to submit the paper for publication.

References

[1] E. M. Clarke, The birth of model checking, in: O. Grumberg, H. Veith (Eds.), 25 Years of Model Checking, no. 5000 in Lecture Notes in Computer Science, Springer Verlag, 2008, pp. 1–26. doi:10.1007/978-3-540-69850-0_1.

[2] J.-P. Aubin, A. M. Bayen, P. Saint-Pierre, Viability Theory: New Directions, Systems & Control: Foundations & Applications, Springer, 2011. doi:10.1007/978-3-642-16684-6.

[3] I. M. Mitchell, M. Chen, M. Oishi, Ensuring safety of nonlinear sampled data systems through reachability, in: Proceedings of the IFAC Conference on Analysis and Design of Hybrid Systems, 2012, pp. 108–114. doi:10.3182/20120606-3-NL-3011.00018.

[4] J. Ding, C. J. Tomlin, Robust reach-avoid controller synthesis for switched nonlinear systems, in: Proceedings of the IEEE Conference on Decision and Control, Atlanta, GA, 2010, pp. 6481–6486. doi:10.1109/CDC.2010.5717115.

[5] J. Lygeros, C. Tomlin, S. Sastry, Controllers for reachability specifications for hybrid systems, Automatica 35 (3) (1999) 349–370. doi:10.1016/S0005-1098(98)00193-9.

[6] P. Cardaliaguet, M. Quincampoix, P. Saint-Pierre, Set-valued numerical analysis for optimal control and differential games, in: M. Bardi, T. E. S. Raghavan, T. Parthasarathy (Eds.), Stochastic and Differential Games: Theory and Numerical Methods, Vol. 4 of Annals of International Society of Dynamic Games, Birkhäuser, 1999, pp. 177–247.

[7] S. Monaco, D. Normand-Cyrot, Advanced tools for nonlinear sampled-data systems' analysis and control, European Journal of Control 13 (2-3) (2007) 221–241. doi:10.3166/ejc.13.221-241.

[8] D. Nešić, A. R. Teel, A framework for stabilization of nonlinear sampled-data systems based on their approximate discrete-time models, IEEE Transactions on Automatic Control 49 (7) (2004) 1103–1122. doi:10.1109/TAC.2004.831175.

[9] B. I. Silva, B. H. Krogh, Modeling and verification of hybrid systems with clocked and unclocked events, in: Proceedings of the IEEE Conference on Decision and Control, Orlando, FL, 2001, pp. 762–767. doi:10.1109/.2001.980198.

[10] S. Azuma, J. Imura, Synthesis of optimal controllers for piecewise affine systems with sampled-data switching, Automatica 42 (5) (2006) 697–710. doi:10.1016/j.automatica.2005.12.023.

[11] Y. Tsuchie, T. Ushio, Control-invariance of sampled-data hybrid systems with periodically clocked events and jitter, in: Proceedings of the IFAC Conference on Analysis and Design of Hybrid Systems, 2006, pp. 417–422. doi:10.3182/20060607-3-IT-3902.00075.

[12] A. Zutshi, S. Sankaranarayanan, A. Tiwari, Timed relational abstractions for sampled data control systems, in: P. Madhusudan, S. Seshia (Eds.), Computer Aided Verification (CAV), Vol. 7358 of Lecture Notes in Computer Science, Springer Verlag, 2012, pp. 343–361. doi:10.1007/978-3-642-31424-7_27.

[13] I. M. Mitchell, Comparing forward and backward reachability as tools for safety analysis, in: A. Bemporad, A. Bicchi, G. Buttazzo (Eds.), Hybrid Systems: Computation and Control, no. 4416 in Lecture Notes in Computer Science, Springer Verlag, 2007, pp. 428–443. doi:10.1007/978-3-540-71493-4_34.

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

[15] J. A. Sethian, A. Vladimirsky, Ordered upwind methods for hybrid control, in: C. J. Tomlin, M. R. Greenstreet (Eds.), Hybrid Systems: Computation and Control, no. 2289 in Lecture Notes in Computer Science, Springer Verlag, 2002, pp. 393–406.

[16] J. Lygeros, On reachability and minimum cost optimal control, Automatica 40 (6) (2004) 917–927. doi:10.1016/j.automatica.2004.01.012.

[17] I. M. Mitchell, A. M. Bayen, C. J. Tomlin, A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games,

IEEE Transactions on Automatic Control 50 (7) (2005) 947–957. doi:10.1109/TAC.2005.851439.

[18] I. M. Mitchell, J. A. Templeton, A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems, in: M. Morari, L. Thiele (Eds.), Hybrid Systems: Computation and Control, no. 3414 in Lecture Notes in Computer Science, Springer Verlag, 2005, pp. 480–494. doi:10.1007/978-3-540-31954-2_31.

[19] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, O. Maler, SpaceEx: Scalable verification of hybrid systems, in: G. Gopalakrishnan, S. Qadeer (Eds.), Proceedings of the International Conference on Computer Aided Verification, no. 6806 in Lecture Notes in Computer Science, Springer, 2011, pp. 379–395. doi:10.1007/978-3-642-22110-1_30.

[20] A. A. Kurzhanskiy, P. Varaiya, Ellipsoidal toolbox, Tech. Rep. UCB/EECS-2006-46, Department of Electrical Engineering and Computer Science, University of California, Berkeley (May 2006).
URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-46.html

[21] J. N. Maidens, S. Kaynama, I. M. Mitchell, M. M. K. Oishi, G. A. Dumont, Lagrangian methods for approximating the viability kernel in high-dimensional systems, Automatica (2013) 15 pages(in press).

[22] S. M. LaValle, Planning Algorithms, Cambridge University Press, New York, 2006.

[23] M. Branicky, M. Curtiss, J. Levine, S. Morgan, Sampling-based planning, control and verification of hybrid systems, IEE Proceedings Control Theory and Applications 153 (5) (2006) 575 – 590.

[24] E. Plaku, L. Kavraki, M. Vardi, Hybrid systems: from verification to falsification by combining motion planning and discrete search, Formal Methods in System Design 34 (2009) 157–182. doi:10.1007/s10703-008-0058-5.

[25] I. M. Mitchell, M. Chen, M. Oishi, Ensuring safety of nonlinear sampled data systems through reachability (extended version), Tech. Rep.

TR-2012-02, Department of Computer Science, University of British Columbia, Vancouver, BC, Canada (April 2012).

[26] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, 2002. doi:10.1007/b98879.

[27] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, UK, 2004.

[28] A. B. Kurzhanski, P. Varaiya, Ellipsoidal techniques for reachability analysis, in: B. Krogh, N. Lynch (Eds.), Hybrid Systems: Computation and Control, no. 1790 in Lecture Notes in Computer Science, Springer Verlag, 2000, pp. 202–214.

[29] A. B. Kurzhanski, P. Varaiya, Ellipsoidal techniques for reachability analysis: Internal approximation, Systems and Control Letters 41 (2000) 201–211.

[30] A. B. Kurzhanski, P. Varaiya, On reachability under uncertainty, SIAM Journal of Control and Optimization 41 (1) (2002) 181–216.

[31] S. Kaynama, J. Maidens, M. Oishi, I. M. Mitchell, G. A. Dumont, Computing the viability kernel using maximal reachable sets, in: Hybrid Systems: Computation and Control, Beijing, China, 2012, pp. 55–64. doi:10.1145/2185632.2185644.

[32] S. Kaynama, I. M. Mitchell, M. M. K. Oishi, G. A. Dumont, Safety-preserving control of high-dimensional continuous-time uncertain linear systems, Poster presented at Hybrid Systems Computation and Control, a part of Cyber-Physical Systems Week (April 2013).

[33] S. Kaynama, I. M. Mitchell, M. M. K. Oishi, G. A. Dumont, Scalable safety-preserving robust control synthesis for continuous-time linear systems, submitted February 2013 to IEEE Transactions on Automatic Control.

[34] S. Kaynama, Scalable techniques for the computation of viable and reachable sets: Safety guarantees for high-dimensional linear time-invariant systems, Ph.D. thesis, Department of Electrical and Computer Engineering, University of British Columbia (July 2012).

[35] J. Löfberg, YALMIP : a toolbox for modeling and optimization in MATLAB, in: Computer Aided Control Systems Design, 2004, pp. 284–289. doi:10.1109/CACSD.2004.1393890.