

# 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](mailto:mitchell@cs.ubc.ca) (Ian M. Mitchell),  
[kaynama@eecs.berkeley.edu](mailto:kaynama@eecs.berkeley.edu) (Shahab Kaynama), [mochen72@eecs.berkeley.edu](mailto:mochen72@eecs.berkeley.edu) (Mo Chen), [moishi@ece.unm.edu](mailto: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,

29 minimum control effort in an energy constrained situation, or proximity to  
30 the human operator’s input in a collaborative control scenario.

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

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

- 50 • Reformulation of the algorithm in terms of discriminating and invari-  
51 ance kernels.
- 52 • An abstract version of the algorithm which does not depend on the  
53 Hamilton-Jacobi (HJ) partial differential equation (PDE).
- 54 • An instantiation of the abstract algorithm’s operators using ellipsoidal  
55 reachability constructs; although this version is restricted to systems  
56 with linear dynamics, it scales much better with state space dimension  
57 than the HJ PDE version.

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

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

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

74 This paper is an extended version of [3], which was presented at the 4<sup>th</sup>  
 75 IFAC Conference on the Analysis and Design of Hybrid Systems (Eindhoven,  
 76 the Netherlands, June 6–8, 2012).

## 77 2. Problem Definition

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

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

80 with initial condition  $x(0) = x_0$ , where  $x \in \Omega$  is the state,  $\Omega \subset \mathbb{R}^{d_x}$  (or some  
 81 similar vector space of dimension  $d_x$ ) is the state space,  $u \in \mathcal{U}$  is the control  
 82 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  
 83 be compact, and the dynamics  $f : \Omega \times \mathcal{U} \times \mathcal{V} \rightarrow \Omega$  are assumed to be Lipschitz  
 84 continuous in  $x$  and continuous in  $u$  and  $v$ . Additional assumptions may be  
 85 necessary for particular versions of the abstract algorithm; for example,  $f$   
 86 must be linear and  $\mathcal{U}$  and  $\mathcal{V}$  must be ellipsoids for the ellipsoidal formulation  
 87 in section 6. Input  $u$  is used to keep the system within the imposed state  
 88 constraints. Input  $v$  seeks to drive the system outside the state constraints,  
 89 and can be used to model the effects of potentially adversarial agents on  
 90 system evolution, to treat uncertainty in the dynamics in a worst case fashion,  
 91 to improve the robustness of the results, or it can be omitted for deterministic  
 92 scenarios.

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

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

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

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

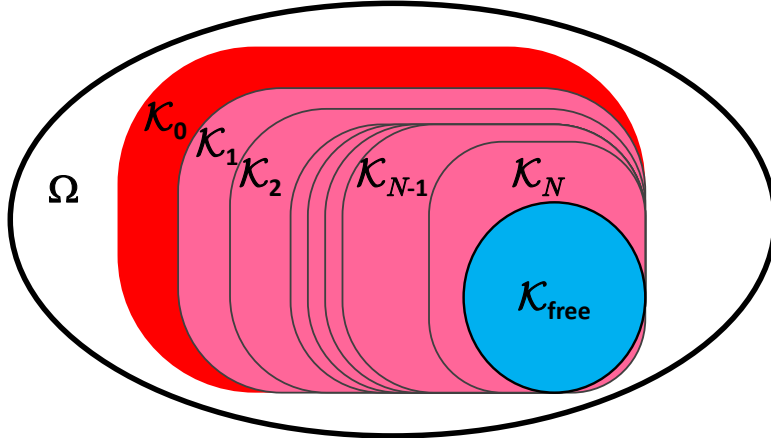


Figure 1: The subdivision of the state space. The constraint set  $\mathcal{K}_0$  and the state space  $\Omega$  are specified in the problem definition. The finite horizon safe sets  $\mathcal{K}_k$  for horizons  $k > 0$ , the free control set  $\mathcal{K}_{\text{free}}$  and the mandatory control 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.

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

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

118 an infinite horizon.

### 119 3. Related Work

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

124 In the context of verification, research on “sampled data systems” has  
125 focused on hybrid systems in which some subset of the mode switches can  
126 only occur at sampling times. In [9], a “sampled data hybrid automata”  
127 formalism was introduced and used to extend the CheckMate hybrid system  
128 verification tool to study a version of such a system with deterministic contin-  
129 uous dynamics. In [10] the authors study a “piecewise affine” version of such  
130 a system; in other words, the state space is partitioned into polyhedra which  
131 specify the modes, in each of which the continuous dynamics are affine with  
132 a control input. An algebraic condition is given which ensures the existence  
133 of a control input signal which drives the system from an initial set of states  
134 to a specific final state; however, the input is assumed to be piecewise con-  
135 tinuous (not piecewise constant) and it is only the mode switching which is  
136 sampled. In [11], the authors consider hybrid systems with nondeterministic  
137 continuous dynamics and a controller which can enable and/or force mode  
138 switches at sampling times, but assume that trajectories of those dynamics  
139 are explicitly available. They then derive necessary and sufficient conditions  
140 for a predicate to be control invariant and show that there is always a supre-  
141 mal control invariant subpredicate for any predicate. Such a subpredicate  
142 corresponds conceptually to a (hybrid) discriminating kernel of the set de-  
143 fined by the predicate, although for their systems the control input can only  
144 influence the mode switching rules, not the continuous evolution. In [12] the  
145 authors consider a hybrid system whose continuous dynamics admit only a  
146 piecewise constant control input signal; however, they must restrict them-  
147 selves to affine dynamics within each mode in order to determine an explicit  
148 representation of trajectories in terms of linear inequalities and thereby con-  
149 struct their “timed relational abstraction,” which can then be composed with  
150 a controller and analyzed with discrete time verification tools. In contrast  
151 to these earlier works on verification of sampled data systems, our abstract  
152 algorithm handles nonlinear continuous dynamics with a piecewise constant  
153 control input signal and robustness provided by allowing for a measurable but

154 bounded disturbance input signal. We do not assume availability of explicit  
155 solutions for the resulting trajectories. Our algorithm is constructive in that  
156 it yields a set-valued control law, although it is potentially conservative. At  
157 present our algorithm is restricted to systems with purely continuous state.

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

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

186 For systems with linear or affine continuous dynamics, there are a number  
187 of Lagrangian algorithms available for reachability; for example, see [19, 20]  
188 and the citations within. While these techniques have not traditionally been  
189 used for control synthesis, they are amenable to the abstract algorithm de-  
190 scribed below. We use the tool [20] to implement the ellipsoidal version of  
191 the algorithm in section 6. The techniques from [19] have been adapted to

192 discrete time viability kernels in [21], but using them for the algorithm de-  
 193 scribed below will require further modification to handle invariance kernels  
 194 and continuous time.

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

## 209 4. Abstract Algorithm

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

### 218 4.1. Preliminary Definitions

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

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

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



226 To construct an approximation to (4) we will sometimes work in an aug-  
 227 mented state space

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

228 with dynamics

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

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

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

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

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

232 From these definitions it is straightforward to show

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

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

243 Although algorithms exist to approximate both continuous and discrete  
 244 time discriminating kernels directly, in this paper we will construct an ap-  
 245 proximation of the sampled data discriminating kernel (4) using a sequence  
 246 of invariance kernels. In some cases these invariance kernels will be computed  
 247 over the augmented dynamics (5) with only input  $v$  treated as a disturbance,

248 while in other cases they will be computed over the original dynamics (1)  
 249 with both inputs  $u$  and  $v$  treated as disturbances. For that reason, we define  
 250 the invariance kernel in terms of a set of dummy variables: system dynamics  
 251  $\dot{y} = g(y, w)$  with initial condition  $y(0) = y_0$ , solution  $y(\cdot)$ , and disturbance  
 252 input  $w$ .

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

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

260 *4.2. Approximating the Sampled Data Discriminating Kernel through Iter-*  
 261 *ated Invariance Kernels*

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

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

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

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

266 **Lemma 1.** *The single step sampled data discriminating kernel is the projec-*  
 267 *tion of a  $\delta$ -horizon invariance kernel in the augmented state space*

$$\text{Disc}_1(\mathcal{S}) = \text{Proj}_x(\text{Inv}_1(\mathcal{S})) \quad (11)$$

268 *Proof.* We seek to show

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

269 To show the rightward implication, assume that  $x_0 \in \text{Disc}_1(\mathcal{S})$ . By (4) there  
 270 exists a  $u_{\text{pw}}(\cdot)$  such that for all  $v(\cdot)$  and  $t \in [0, \delta]$ ,  $x(t) \in \mathcal{S}$  where  $x(\cdot)$   
 271 solves (2) with initial condition  $x(0) = x_0$ . But for  $t \in [0, \delta]$ ,  $u_0 \triangleq u_{\text{pw}}(t)$  is a

272 constant by (3), so the augmented trajectory  $\tilde{x}(\cdot) = [x(\cdot) \ u_0]^T$  satisfies (5).  
 273 Since  $u_0 \in \mathcal{U}$  by (3) and for all  $v(\cdot)$ ,  $x(\cdot) \in \mathcal{S}$  over the same time interval,  
 274 it must be that for all  $v(\cdot)$ ,  $\tilde{x}(\cdot) \in \mathcal{S} \times \mathcal{U}$ . By (8) we have that  $[x_0 \ u_0]^T \in$   
 275  $\text{Inv}_1(\mathcal{S})$ , and hence by (6) that  $x_0 \in \text{Proj}_x(\text{Inv}_1(\mathcal{S}))$ .

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

283 Approximation of the sampled data discriminating kernel over longer hori-  
 284 zons is then performed recursively

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

#### 285 4.3. Conservatism of the Approximation

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

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

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

289 *Proof.* We start by using induction to show containment:

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

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

$$x(t) = \begin{cases} x^a(t) & 0 \leq t < \delta \\ x^b(t) & \delta \leq t \leq k\delta \end{cases} \quad \text{and} \quad u_{\text{pw}}(t) = \begin{cases} u_{\text{pw}}^a(t) & 0 \leq t < \delta \\ u_{\text{pw}}^b(t) & \delta \leq t < k\delta. \end{cases}$$

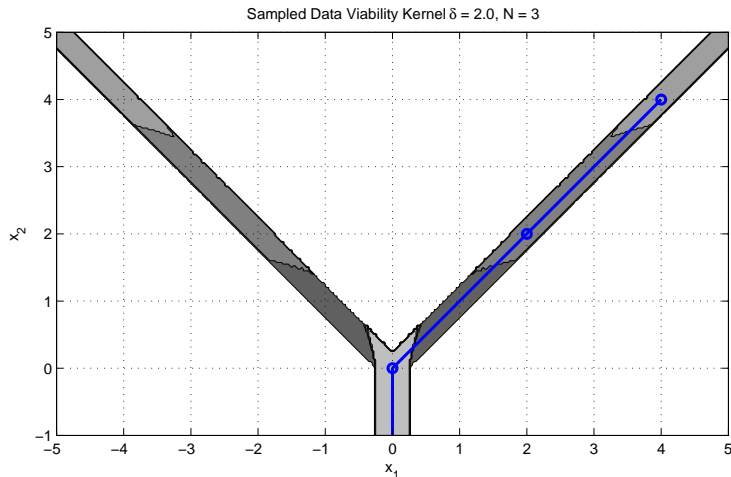


Figure 2: A demonstration that  $\text{Disc}_k(\mathcal{S})$  may exclude states which can remain inside  $\mathcal{S}$  over horizon  $k\delta$ . In this case  $\mathcal{S}$  is the Y-shaped shaded region. The states in  $\text{Disc}_k(\mathcal{S})$  for  $k = 0, 1, 2, 3$  are shown darkest to lightest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

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

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

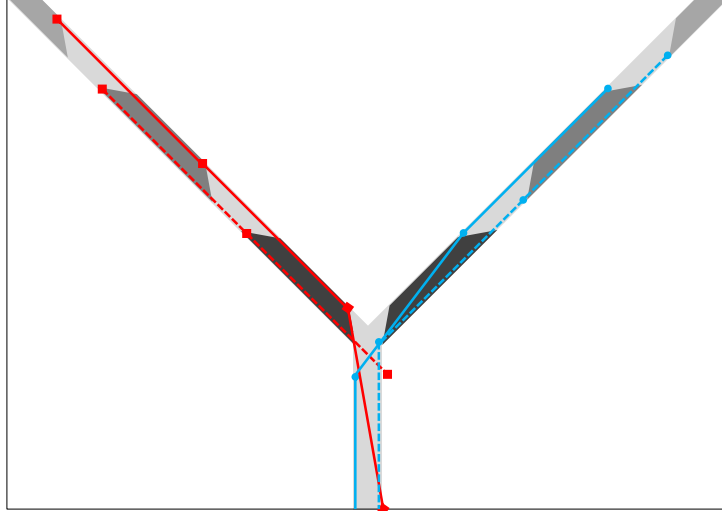


Figure 3: A sketch of the actual  $\text{Disc}_{\text{sd}}([0, k\delta], \mathcal{S})$  for  $k = 0, 1, 2, 3$  and some sample trajectories for the example in figure 2. The lightest shaded regions (including the periodic gaps between the darker regions on the arms of the Y) are actually within  $\text{Disc}_{\text{sd}}([0, \infty], \mathcal{S})$ . 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.

313 shown in figure 2 starts at  $[+4 \quad +4]^T$  and uses input signal

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

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

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

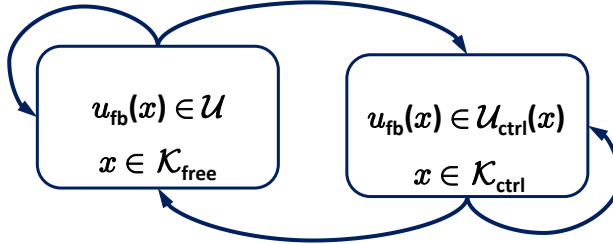


Figure 4: The general form of the switched sampled data control policy. Arrows show transitions which are possible under the policy.

322 *4.4. Subdivision of the Constraint Set*

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

$$\mathcal{K}_k = \text{Disc}_k(\mathcal{K}_0). \quad (13)$$

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

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

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

333 *4.5. Control Policy Synthesis*

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

339 In order to be permissive, the policy is often set-valued. In subsequent  
 340 sections we will examine reasons why one input might be favored over an-  
 341 other based on additional information available from specific computational

342 algorithms—for example, an approximation of how deep within a set the fu-  
 343 ture trajectory will stay—but at this stage we treat equally all control signals  
 344 for which we can guarantee safety.

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

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

347 The control policy is given by

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

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

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

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

#### 355 4.6. Safety of the Policy

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

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

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

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

366 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  
 367  $j = 1, 2, 3, \dots, n(x_0) - 1$ . Concatenating the  $u_j$  for  $j = 0, 1, 2, \dots, n(x_0) - 1$   
 368 together we arrive at a control signal which satisfies (17) and maintains  
 369  $x(t) \in \mathcal{K}_0$  for all  $t \in [0, n(x_0)\delta]$ .

370 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  
 371 particular, if  $x(\delta) \in \mathcal{K}_N$ , then by the argument above we can construct a  
 372 sampled feedback control policy according to (17) such that  $x(t) \in \mathcal{K}_0$  for all  
 373  $t \in [0, (N+1)\delta]$ .  $\square$

#### 374 4.7. What About Infinite Horizon?

375 **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*  
 376 *possible to guarantee  $x(\cdot) \in \mathcal{K}_\infty$  for all  $t > 0$ .*

377 *Proof.* Let  $x_0 \in \mathcal{K}_{k+1} = \mathcal{K}_k$ . By (12) and (9),  $x_0 \in \text{Proj}_x(\text{Inv}_1(\mathcal{K}_k))$  and  
 378 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,  
 379  $x(\delta) \in \mathcal{K}_k = \mathcal{K}_{k+1}$ , so use the same argument to construct a new constant  
 380 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  $t \in [j\delta, (j+1)\delta]$   
 381 for all  $j = 1, 2, 3, \dots$ . Concatenating the  $u_j$  together we arrive at a  
 382 control signal which satisfies (17) and maintains  $x(t) \in \mathcal{K}_k$  for all  $t > 0$  (thus  
 383 justifying the notational choice  $\mathcal{K}_k = \mathcal{K}_\infty$ ).  $\square$

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

## 392 5. Hamilton-Jacobi Formulation

393 In this section we outline how to implement the abstract algorithm above  
 394 using an HJ PDE formulation of invariance kernels. The main advantages  
 395 of this formulation are that general nonlinear dynamics (1) can be handled  
 396 and implementation of the key operators in the algorithm are straightforward.  
 397 The main disadvantage is the computational cost: exponential in the number  
 398 of dimensions in which the invariance kernel is calculated. We demonstrate  
 399 how the constants involved can be kept small for the control dimensions, but



400 there is at present no way to escape the curse of dimensionality for the state  
 401 space dimensions.

402 *5.1. Preliminaries: Implicit Surface Functions and the HJ PDE*

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

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

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

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

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

424 where  $\phi$  is the viscosity solution of the terminal value, time-dependent HJ  
 425 PDE

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

426 with Hamiltonian

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

427 and terminal condition

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

428 *5.2. Hamilton-Jacobi Formulation of Operators*

429 Using properties of the implicit surface function and the HJ PDE formu-  
 430 lation of invariance kernels described above, we can implement the operators  
 431 needed to approximate the sampled data discriminating kernel.

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

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

434 where  $\tilde{x} = [x \ u]^T$ . To find the implicit surface representation  $\psi_{\text{Inv}_1(\mathcal{S})}$  of (10)  
 435 we solve

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

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

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

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

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

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

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

442 *5.3. Control Policy Synthesis*

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

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

445 where  $\tilde{x}_0 = [x_0 \ u]^T$ . However, there is additional quantitative information  
 446 in the implicit surface functions  $\psi_{\mathcal{K}_k}$  which we can take advantage of to con-  
 447 struct alternative representations of the control policy and even alternative  
 448 control policies.

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

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

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

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

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

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

455 while two alternative policies are given by

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

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

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

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

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

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

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

463 By (22)

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

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

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

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

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

468 The intuition behind these different policies is

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

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

#### 486 5.4. Practical Implementation

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

##### 490 5.4.1. Approximating the Implicit Surface Functions

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

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

#### 508 5.4.2. Mitigating the Curse of Dimensionality

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

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

534 tive when the number of control input dimensions is low and/or the optimal  
 535 samples for the control input can be guessed a priori.

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

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

### 554 5.4.3. Constructing the Feedback Controller

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

557 To construct an implicit surface representation of the set  $\mathcal{U}_{\text{ctrl}}(x_0)$ , create  
 558 a grid of  $u$  values  $\{u_j\}_j$  and then a grid of augmented state values  $\{\tilde{x}_j\}_j$   
 559 such that  $\tilde{x}_j = [x_0 \quad u_j]^T$ . Using numerical interpolation where necessary,  
 560 evaluate  $\psi_{\text{Inv}_1(\mathcal{K}_{n(x_0)-1})}(\tilde{x}_j)$  on the grid  $\{\tilde{x}_j\}_j$ . Then (21) provides an approx-  
 561 imation on the grid  $\{u_j\}_j$  of an implicit surface function  $\psi_{\mathcal{U}_{\text{ctrl}}(x_0)}(u)$  repre-  
 562 senting  $\mathcal{U}_{\text{ctrl}}(x_0)$ . Interpolation of  $\psi_{\mathcal{U}_{\text{ctrl}}(x_0)}(u)$  (which is continuous) can be  
 563 used to approximate the full set of safe inputs if the control input dimension  
 564 is sufficiently well sampled.

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

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

#### 572 5.4.4. *Guaranteeing an Underapproximation*

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

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

#### 588 5.5. *Example*

589 Computations were done on an Intel Core2 Duo at 1.87 GHz with 4 GB  
590 RAM running 64-bit Windows 7 Professional (Service Pack 1), 64-bit MAT-  
591 LAB version 7.11 (R2010b), and TOOLBOXLS version 1.1. MATLAB code can  
592 be found at the first author’s web site <http://www.cs.ubc.ca/~mitchell>

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

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

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

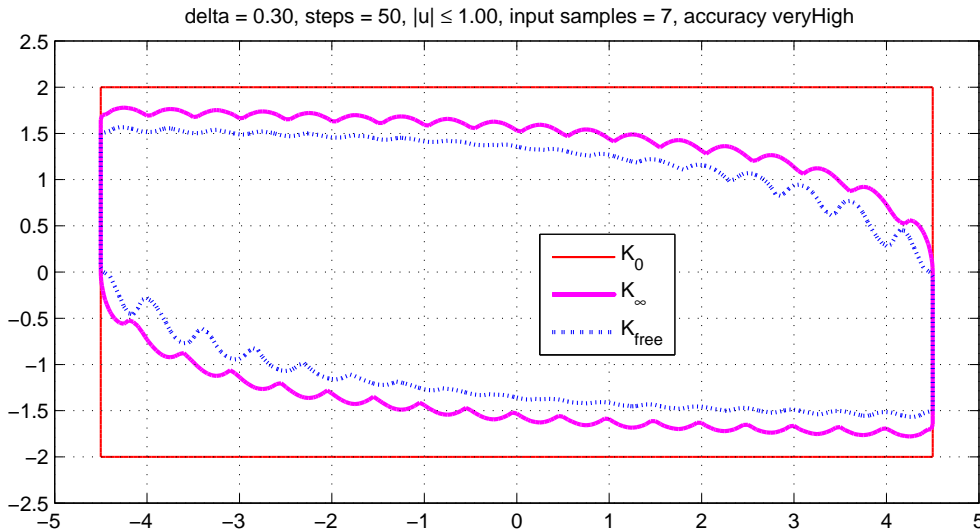


Figure 5: The partition of  $\Omega$  for the spatially varying double integrator with  $\delta = 0.3$  and horizon  $N = 50$  (eg:  $T = 15$ , long enough for convergence). The state constraint  $\mathcal{K}_0$  is the outermost thin red rectangle,  $\mathcal{K}_\infty$  is inside the thick magenta 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).

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

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

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

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



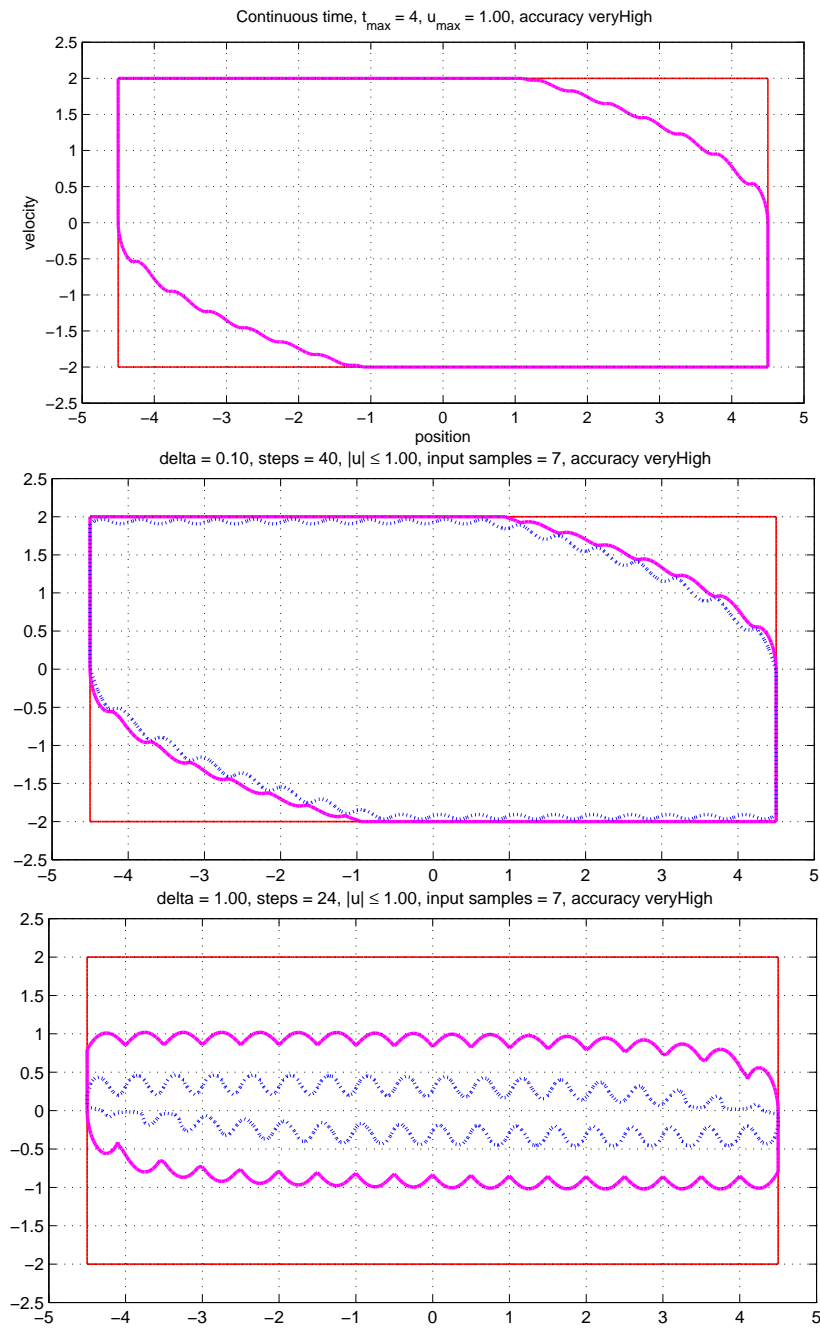


Figure 6: The effect of  $\delta$  on the spatial partition. Top: Traditional reachability with continuous state feedback and measurable control signals ( $T = 4$ ). Middle: Sampled data with  $\delta = 0.1$ ,  $N = 40$  ( $T = 4$ ). Bottom: Sampled data with  $\delta = 1.0$ ,  $N = 24$  ( $T = 24$ ).

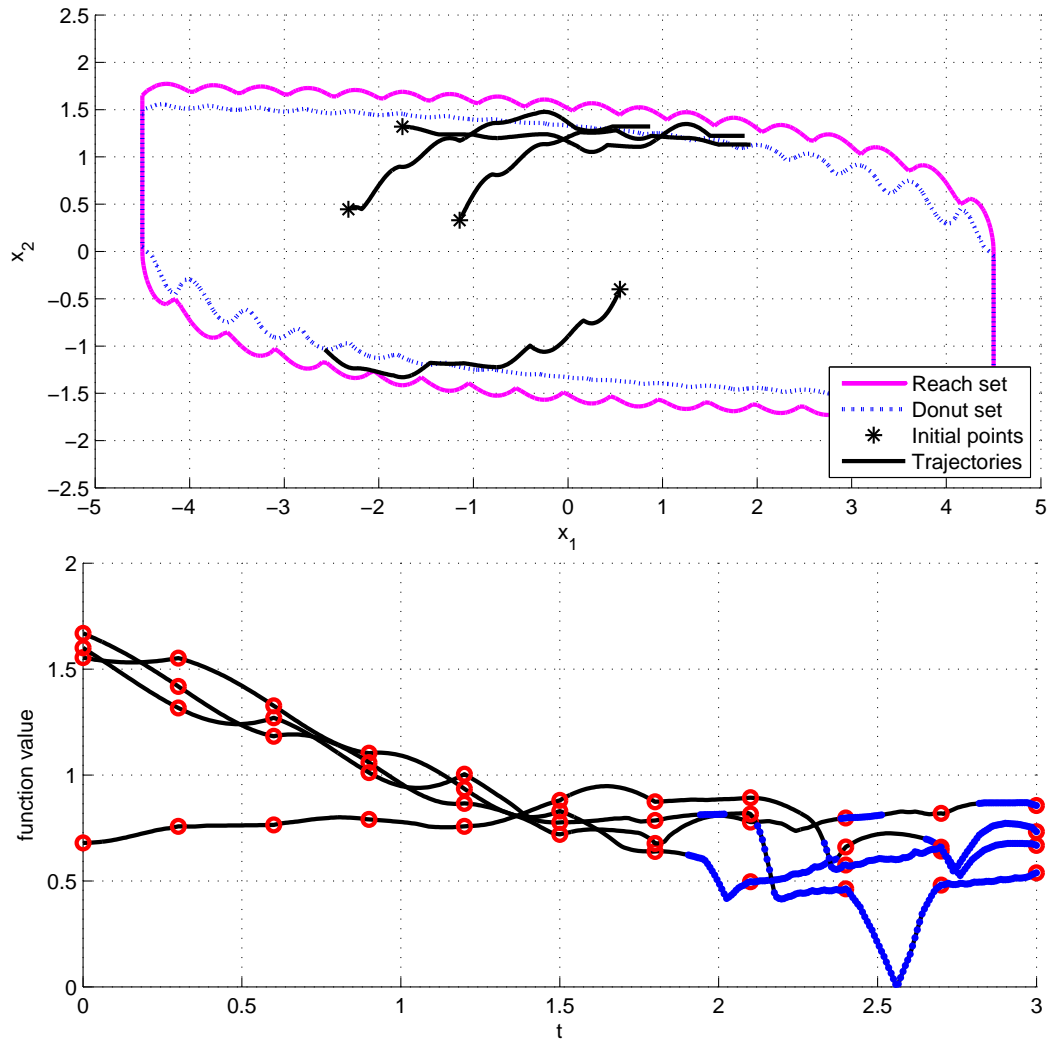


Figure 7: Sample trajectories using the intermediate safe policy  $\mathcal{U}_{\text{ctrl}}^{\vec{}}$  for  $\delta = 0.3$ . Top: Trajectories  $x(\cdot)$  in phase space overlaid on the state space partition. Bottom row:  $\psi_{\mathcal{X}_{\infty}}(x(t))$  versus  $t$ . Sample times are shown as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

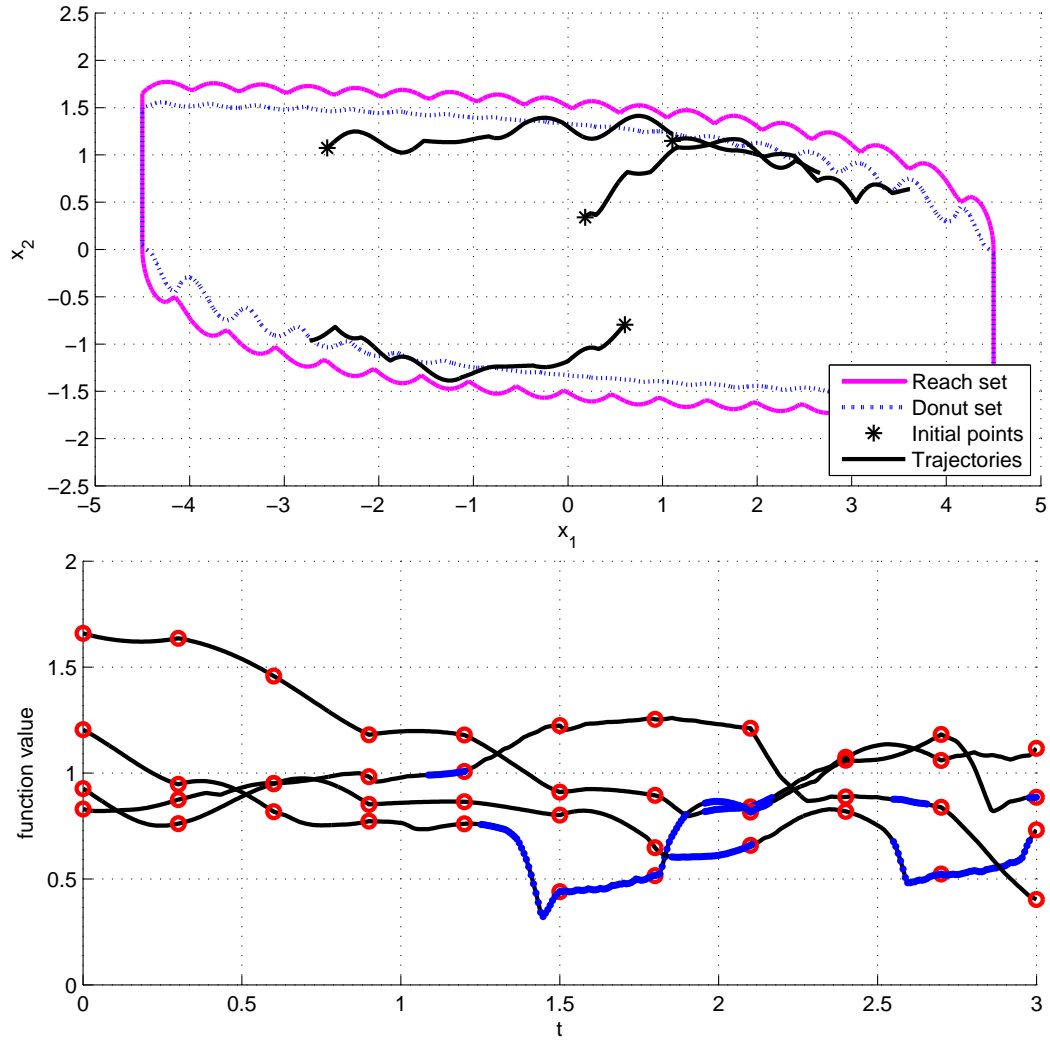


Figure 8: Sample trajectories using the aggressive safe policy  $\mathcal{U}_{ctrl}^{\searrow}$  for  $\delta = 0.3$ . Top: Trajectories  $x(\cdot)$  in phase space overlaid on the state space partition. Bottom row:  $\psi_{\mathcal{X}_{\infty}}(x(t))$  versus  $t$ . Sample times are shown as red circles,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

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

## 624 6. Ellipsoidal Formulation

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

### 633 6.1. Preliminaries: Ellipsoidal Complications

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

$$P_x = \begin{bmatrix} I_{d_x} & 0_{d_x \times d_u} \end{bmatrix} \quad \text{and} \quad P_u = \begin{bmatrix} 0_{d_u \times d_x} & I_{d_u} \end{bmatrix}$$

636 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.  
637 Given an augmented state  $\tilde{x} = \begin{bmatrix} x & u \end{bmatrix}^T$ , we then have that  $P_x \tilde{x} = x$  and  
638  $P_u \tilde{x} = u$ . More generally, we could choose  $P$  such that the rows form an  
639 orthonormal basis for a subspace into which we want to project a vector.

#### 640 6.1.1. Preliminaries: Ellipsoids

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

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

642 where  $q \in \mathbb{R}^d$  is the center,  $H = H^T \in \mathbb{R}^{d \times d}$ , and  $HH^T = H^2$  is the symmet-  
643 ric positive definite shape matrix. For matrix  $A$ , the linear mapping of an  
644 ellipsoid is also an ellipsoid

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

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

646 Many of the sets  $\mathcal{S}$  involved in the algorithm below will not be ellip-  
 647 soidal, so where necessary we will construct ellipsoidal approximations  $\mathcal{E}_{\mathcal{S}}$ .  
 648 An “ellipsoidal approximation” of a set is not a unique object, but in this  
 649 algorithm it will typically be an underapproximation, it will often be a max-  
 650 imum volume underapproximation, and the particular choice for each such  
 651 approximation in the algorithm should be clear from context.

### 652 6.1.2. Preliminaries: Maximum Volume Inscribed Ellipsoids

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

$$\mathcal{Y}_i = \mathcal{Y}_{i,x} \times \mathcal{Y}_{i,u}$$

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

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

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

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

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

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

$$\begin{aligned} & \lambda_i > 0 \\ & \begin{bmatrix} 1 - \lambda_i & 0 & (\bar{q} - q_i)^T \\ 0 & \lambda_i I & \bar{H} \\ (\bar{q} - q_i) & \bar{H} & H_i^2 \end{bmatrix} \geq 0, \end{aligned} \quad (27)$$

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

$$\begin{aligned}
& \lambda_{i,x} > 0 \\
& \lambda_{i,u} > 0 \\
& \begin{bmatrix} 1 - \lambda_{i,x} & 0 & (P_x \bar{q} - q_{i,x})^T \\ 0 & \lambda_{i,x} I & P_x \bar{H} P_x^T \\ (P_x \bar{q} - q_{i,x}) & P_x \bar{H} P_x^T & H_{i,x}^2 \end{bmatrix} \geq 0 \\
& \begin{bmatrix} 1 - \lambda_{i,u} & 0 & (P_u \bar{q} - q_{i,u})^T \\ 0 & \lambda_{i,u} I & P_u \bar{H} P_u^T \\ (P_u \bar{q} - q_{i,u}) & P_u \bar{H} P_u^T & H_{i,u}^2 \end{bmatrix} \geq 0
\end{aligned} \tag{28}$$

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

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

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

### 676 6.1.3. Preliminaries: Ellipsoidal Underapproximation of Invariance Kernels

677 For the implicit surface function representations used in the previous sec-  
678 tion, there was an HJ PDE whose solution governed their evolution. The  
679 situation is more complicated for the ellipsoidal representation: We will con-  
680 struct invariance kernels by a sequence of reachability and intersection oper-  
681 ations.

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

$$\dot{x}(t) = Ax(t) + Bu(t) + Gv(t) \tag{29}$$

$$\dot{x}(t) = Ax(t) + Bu_{pw}(t) + Gv(t) \tag{30}$$

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

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

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

686 where  $y(\cdot)$  solves  $\dot{y} = g(y, w)$  with initial condition  $y(0) = y_0$  and  $w(\cdot)$  is  
687 a measurable input function such that  $w(t) \in \mathcal{W}$ . If  $g$  is linear and both  
688  $\mathcal{S} = \mathcal{E}_{\mathcal{S}}$  and  $\mathcal{W} = \mathcal{E}_{\mathcal{W}}$  are ellipsoidal, it is possible to construct an ellipsoidal

689 underapproximation  $\mathcal{E}_{\text{Reach}(t,\mathcal{S})}(\ell) \subseteq \text{Reach}(t, \mathcal{S})$  for a given vector  $\ell \in \mathbb{R}^d$  [28,  
690 29, 30]. More generally,  $\cup_i \mathcal{E}_{\text{Reach}(t,\mathcal{S})}(\ell_i)$  for some set of vectors  $\{\ell_i\}$  can be  
691 used as a piecewise ellipsoidal underapproximation of  $\text{Reach}(t, \mathcal{S})$ .

692 In [31] we presented an algorithm to underapproximate continuous time  
693 viability kernels using these ellipsoidal reachability constructs, and in [32, 33]  
694 we extended this algorithm to discriminating kernels for systems with adver-  
695 sarial inputs. Here we briefly outline how to simplify the latter to approx-  
696 imate invariance kernels. For linear dynamics  $g$ , ellipsoidal  $\mathcal{S} = \mathcal{E}_{\mathcal{S}}$  and  
697  $\mathcal{W} = \mathcal{E}_{\mathcal{W}}$ , and vector  $\ell$ , the algorithm creates an ellipsoidal underapproxima-  
698 tion  $\mathcal{E}_{\text{Inv}([0,\delta],\mathcal{S},w,g)}(\ell)$  using a series of substeps. Start by choosing the number  
699 of substeps  $\hat{n} > 0$  and the substep length  $\hat{\delta} = \delta/\hat{n}$ . If necessary, erode  $\mathcal{S}$   
700 to keep trajectories safe during the substeps (several approaches to such ero-  
701 sion are detailed in [34, pp. 94–97]). Then compute the sequence  $\hat{\mathcal{E}}_{\mathcal{S}}^{(\hat{k})}(\ell)$  for  
702  $\hat{k} = 0, 1, \dots, \hat{n}$  where

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

### 703 6.2. Ellipsoidal Formulation of Operators

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

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

$$\mathcal{E}_{\mathcal{S} \times \mathcal{U}} = \text{Inscribed}_{\mathbb{I}}(\mathcal{S} \times \mathcal{U}).$$

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

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

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

$$\mathcal{E}_{\text{Disc}_1(\mathcal{S})}(\ell) = \text{Proj}_x(\mathcal{E}_{\text{Inv}_1(\mathcal{S})}(\ell)) = P_x \mathcal{E}_{\text{Inv}_1(\mathcal{S})}(\ell)$$

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

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

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

### 721 6.3. Control Policy Synthesis

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

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

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

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

### 724 6.4. Practical Implementation

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

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



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

744 Furthermore, when computing  $\text{Inv}_1(\mathcal{K}_0) = \text{Inv}([0, \delta], \mathcal{E}_{\mathcal{K}_0 \times \mathcal{U}}, v, \tilde{f})$  to find  
 745  $\mathcal{K}_1$ , 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—  
 746 because the dynamics for the  $u$  dimension in  $\tilde{f}$  are zero, and so no erosion  
 747 in those dimensions is required to ensure safety of trajectories during the  
 748 substeps.

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

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

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

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

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

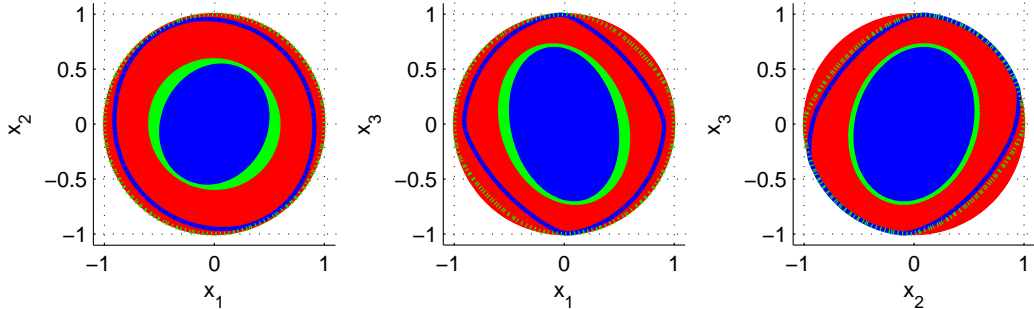


Figure 9: Projections of the partition of  $\Omega$  into various pairs of state variables for the oscillating double integrator with  $\delta = 0.1$  and  $N = 30$ . The outermost solid circle is  $\mathcal{K}_0$ . The innermost solid ellipse is  $\mathcal{E}_{\mathcal{K}_{\text{free}}}$ , which is 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  $\ell$ . The true sets  $\mathcal{K}_{\text{free}}$  and  $\mathcal{K}_N$  (the contours) were approximated by the HJ PDE formulation described in section 5.

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

### 779 6.5. Example

780 We illustrate the algorithm using another variation of the double integra-  
 781 tor: dynamics (29) with

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

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

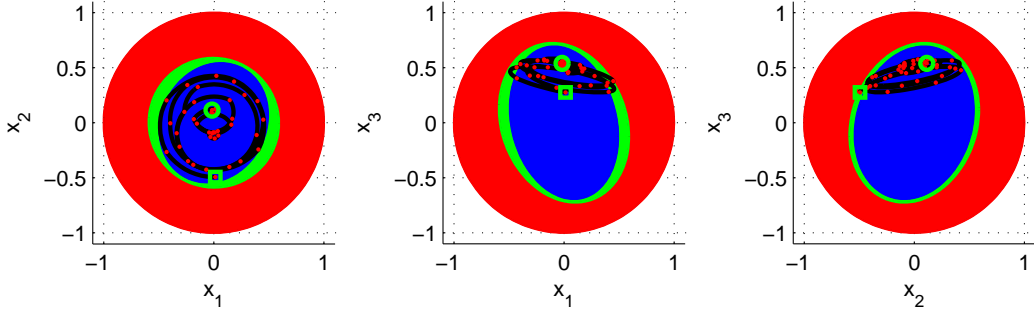


Figure 10: Projections of a trajectory for the oscillating double integrator with  $\delta = 0.1$  for  $0 \leq t \leq 4$ , overlaid on the state space partition from figure 9. The initial condition is shown with a light green circle, the final state by a light green square, and intermediate sample times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

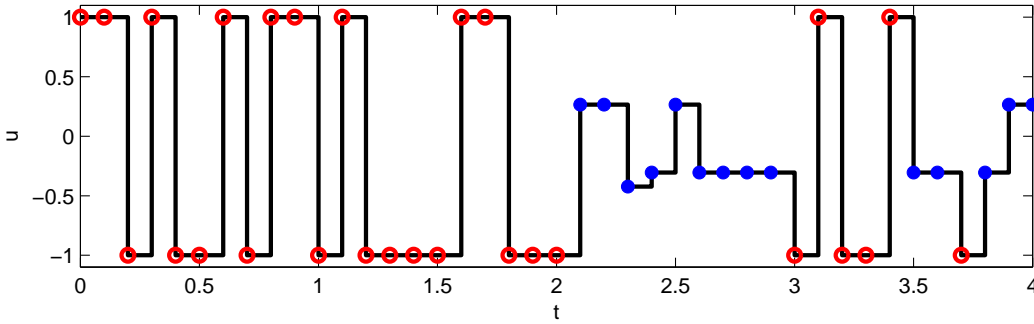


Figure 11: Control signal for the trajectory shown in figure 10. Circles mark sample times. Controls chosen in  $\mathcal{U}$  are shown with open red circles, while those chosen in  $\mathcal{E}_{\mathcal{U}_{\text{ctrl}}}(x)$  are shown with closed blue circles.

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

791 Projections of a sample trajectory  $x(\cdot)$  are shown in figure 10, and the  
 792 control signal  $u_{\text{pw}}(\cdot)$  used to generate this trajectory is shown in figure 11. In  
 793 this example both  $\mathcal{U}$  and  $\mathcal{E}_{\mathcal{U}_{\text{ctrl}}}(x)$  are always an interval (the latter possibly  
 794 degenerate). The control signal in figure 11 was generated by randomly  
 795 choosing one of the endpoints of the interval  $\mathcal{U}$  (if  $x(t_k) \in \mathcal{E}_{\mathcal{K}_{\text{free}}}$ ) or  $\mathcal{E}_{\mathcal{U}_{\text{ctrl}}}(x(t_k))$   
 796 (if  $x(t_k) \in \mathcal{E}_{\mathcal{K}_{\text{ctrl}}}$ ) at each sample time  $t_k$ . Although the state space partition  
 797 was constructed with finite horizon  $N = 30$  (corresponding to  $t = 3$ ), the  
 798 trajectory clearly stays well within  $\mathcal{K}_0$  out to  $t = 4$  (the final time shown

799 in figures 10 and 11); in fact, in our simulations it stayed within  $\mathcal{K}_0$  for all  
800 times that we tried.

## 801 **7. Conclusions and Future Work**

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

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

## 819 **8. Role of the Funding Source**

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

827 The funding agencies had no role in study design; in the collection, analy-  
828 sis, and interpretation of data; in the writing of the report; or in the decision  
829 to submit the paper for publication.

830 **References**

- 831 [1] E. M. Clarke, The birth of model checking, in: O. Grumberg, H. Veith  
832 (Eds.), 25 Years of Model Checking, no. 5000 in Lecture Notes in  
833 Computer Science, Springer Verlag, 2008, pp. 1–26. doi:10.1007/  
834 978-3-540-69850-0\_1.
- 835 [2] J.-P. Aubin, A. M. Bayen, P. Saint-Pierre, Viability Theory: New Direc-  
836 tions, Systems & Control: Foundations & Applications, Springer, 2011.  
837 doi:10.1007/978-3-642-16684-6.
- 838 [3] I. M. Mitchell, M. Chen, M. Oishi, Ensuring safety of nonlinear sampled  
839 data systems through reachability, in: Proceedings of the IFAC Con-  
840 ference on Analysis and Design of Hybrid Systems, 2012, pp. 108–114.  
841 doi:10.3182/20120606-3-NL-3011.00018.
- 842 [4] J. Ding, C. J. Tomlin, Robust reach-avoid controller synthesis for  
843 switched nonlinear systems, in: Proceedings of the IEEE Conference  
844 on Decision and Control, Atlanta, GA, 2010, pp. 6481–6486. doi:  
845 10.1109/CDC.2010.5717115.
- 846 [5] J. Lygeros, C. Tomlin, S. Sastry, Controllers for reachability specifi-  
847 cations for hybrid systems, Automatica 35 (3) (1999) 349–370. doi:  
848 10.1016/S0005-1098(98)00193-9.
- 849 [6] P. Cardaliaguet, M. Quincampoix, P. Saint-Pierre, Set-valued numer-  
850 ical analysis for optimal control and differential games, in: M. Bardi,  
851 T. E. S. Raghavan, T. Parthasarathy (Eds.), Stochastic and Differential  
852 Games: Theory and Numerical Methods, Vol. 4 of Annals of Interna-  
853 tional Society of Dynamic Games, Birkhäuser, 1999, pp. 177–247.
- 854 [7] S. Monaco, D. Normand-Cyrot, Advanced tools for nonlinear sampled-  
855 data systems’ analysis and control, European Journal of Control 13 (2-3)  
856 (2007) 221–241. doi:10.3166/ejc.13.221-241.
- 857 [8] D. Nešić, A. R. Teel, A framework for stabilization of nonlinear sampled-  
858 data systems based on their approximate discrete-time models, IEEE  
859 Transactions on Automatic Control 49 (7) (2004) 1103–1122. doi:10.  
860 1109/TAC.2004.831175.

- 861 [9] B. I. Silva, B. H. Krogh, Modeling and verification of hybrid sys-  
862 tems with clocked and unclocked events, in: Proceedings of the IEEE  
863 Conference on Decision and Control, Orlando, FL, 2001, pp. 762–767.  
864 doi:10.1109/.2001.980198.
- 865 [10] S. Azuma, J. Imura, Synthesis of optimal controllers for piecewise affine  
866 systems with sampled-data switching, *Automatica* 42 (5) (2006) 697–  
867 710. doi:10.1016/j.automatica.2005.12.023.
- 868 [11] Y. Tsuchie, T. Ushio, Control-invariance of sampled-data hybrid systems  
869 with periodically clocked events and jitter, in: Proceedings of the IFAC  
870 Conference on Analysis and Design of Hybrid Systems, 2006, pp. 417–  
871 422. doi:10.3182/20060607-3-IT-3902.00075.
- 872 [12] A. Zutshi, S. Sankaranarayanan, A. Tiwari, Timed relational abstrac-  
873 tions for sampled data control systems, in: P. Madhusudan, S. Se-  
874 shia (Eds.), *Computer Aided Verification (CAV)*, Vol. 7358 of Lec-  
875 ture Notes in Computer Science, Springer Verlag, 2012, pp. 343–361.  
876 doi:10.1007/978-3-642-31424-7\_27.
- 877 [13] I. M. Mitchell, Comparing forward and backward reachability as tools  
878 for safety analysis, in: A. Bemporad, A. Bicchi, G. Buttazzo (Eds.),  
879 *Hybrid Systems: Computation and Control*, no. 4416 in *Lecture Notes*  
880 *in Computer Science*, Springer Verlag, 2007, pp. 428–443. doi:10.1007/  
881 978-3-540-71493-4\_34.
- 882 [14] M. S. Branicky, G. Zhang, Solving hybrid control problems: Level sets  
883 and behavioral programming, in: Proceedings of the American Control  
884 Conference, Chicago, IL, 2000, pp. 1175–1180.
- 885 [15] J. A. Sethian, A. Vladimirovsky, Ordered upwind methods for hybrid con-  
886 trol, in: C. J. Tomlin, M. R. Greenstreet (Eds.), *Hybrid Systems: Com-  
887 putation and Control*, no. 2289 in *Lecture Notes in Computer Science*,  
888 Springer Verlag, 2002, pp. 393–406.
- 889 [16] J. Lygeros, On reachability and minimum cost optimal control, *Auto-  
890 matica* 40 (6) (2004) 917–927. doi:10.1016/j.automatica.2004.01.  
891 012.
- 892 [17] I. M. Mitchell, A. M. Bayen, C. J. Tomlin, A time-dependent Hamilton-  
893 Jacobi formulation of reachable sets for continuous dynamic games,

- 894 IEEE Transactions on Automatic Control 50 (7) (2005) 947–957. doi:  
895 10.1109/TAC.2005.851439.
- 896 [18] I. M. Mitchell, J. A. Templeton, A toolbox of Hamilton-Jacobi solvers  
897 for analysis of nondeterministic continuous and hybrid systems, in:  
898 M. Morari, L. Thiele (Eds.), Hybrid Systems: Computation and Control,  
899 no. 3414 in Lecture Notes in Computer Science, Springer Verlag,  
900 2005, pp. 480–494. doi:10.1007/978-3-540-31954-2\_31.
- 901 [19] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel,  
902 R. Ripado, A. Girard, T. Dang, O. Maler, SpaceEx: Scalable verification  
903 of hybrid systems, in: G. Gopalakrishnan, S. Qadeer (Eds.), Proceedings  
904 of the International Conference on Computer Aided Verification, no.  
905 6806 in Lecture Notes in Computer Science, Springer, 2011, pp. 379–  
906 395. doi:10.1007/978-3-642-22110-1\_30.
- 907 [20] A. A. Kurzhanskiy, P. Varaiya, Ellipsoidal toolbox, Tech. Rep.  
908 UCB/EECS-2006-46, Department of Electrical Engineering and Com-  
909 puter Science, University of California, Berkeley (May 2006).  
910 URL [http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/  
911 EECS-2006-46.html](http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-46.html)
- 912 [21] J. N. Maidens, S. Kaynama, I. M. Mitchell, M. M. K. Oishi, G. A.  
913 Dumont, Lagrangian methods for approximating the viability kernel in  
914 high-dimensional systems, *Automatica* (2013) 15 pages(in press).
- 915 [22] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, New  
916 York, 2006.
- 917 [23] M. Branicky, M. Curtiss, J. Levine, S. Morgan, Sampling-based plan-  
918 ning, control and verification of hybrid systems, *IEE Proceedings Control  
919 Theory and Applications* 153 (5) (2006) 575 – 590.
- 920 [24] E. Plaku, L. Kavraki, M. Vardi, Hybrid systems: from verification to  
921 falsification by combining motion planning and discrete search, *Formal  
922 Methods in System Design* 34 (2009) 157–182. doi:10.1007/  
923 s10703-008-0058-5.
- 924 [25] I. M. Mitchell, M. Chen, M. Oishi, Ensuring safety of nonlinear sam-  
925 pled data systems through reachability (extended version), Tech. Rep.

- 926 TR-2012-02, Department of Computer Science, University of British  
927 Columbia, Vancouver, BC, Canada (April 2012).
- 928 [26] S. Osher, R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*,  
929 Springer, 2002. doi:10.1007/b98879.
- 930 [27] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University  
931 Press, Cambridge, UK, 2004.
- 932 [28] A. B. Kurzhanski, P. Varaiya, Ellipsoidal techniques for reachability  
933 analysis, in: B. Krogh, N. Lynch (Eds.), *Hybrid Systems: Computation  
934 and Control*, no. 1790 in *Lecture Notes in Computer Science*, Springer  
935 Verlag, 2000, pp. 202–214.
- 936 [29] A. B. Kurzhanski, P. Varaiya, Ellipsoidal techniques for reachability  
937 analysis: Internal approximation, *Systems and Control Letters* 41 (2000)  
938 201–211.
- 939 [30] A. B. Kurzhanski, P. Varaiya, On reachability under uncertainty, *SIAM  
940 Journal of Control and Optimization* 41 (1) (2002) 181–216.
- 941 [31] S. Kaynama, J. Maidens, M. Oishi, I. M. Mitchell, G. A. Dumont, Com-  
942 puting the viability kernel using maximal reachable sets, in: *Hybrid  
943 Systems: Computation and Control*, Beijing, China, 2012, pp. 55–64.  
944 doi:10.1145/2185632.2185644.
- 945 [32] S. Kaynama, I. M. Mitchell, M. M. K. Oishi, G. A. Dumont, Safety-  
946 preserving control of high-dimensional continuous-time uncertain linear  
947 systems, Poster presented at *Hybrid Systems Computation and Control*,  
948 a part of *Cyber-Physical Systems Week* (April 2013).
- 949 [33] S. Kaynama, I. M. Mitchell, M. M. K. Oishi, G. A. Dumont, Scalable  
950 safety-preserving robust control synthesis for continuous-time linear sys-  
951 tems, submitted February 2013 to *IEEE Transactions on Automatic  
952 Control*.
- 953 [34] S. Kaynama, Scalable techniques for the computation of viable and  
954 reachable sets: Safety guarantees for high-dimensional linear time-  
955 invariant systems, Ph.D. thesis, Department of Electrical and Computer  
956 Engineering, University of British Columbia (July 2012).



- 957 [35] J. Löfberg, YALMIP : a toolbox for modeling and optimization in MAT-  
958 LAB, in: Computer Aided Control Systems Design, 2004, pp. 284–289.  
959 doi : 10.1109/CACSD.2004.1393890.