Lagrangian methods for approximating the viability kernel in high-dimensional systems

John N. Maidens [a 1], Shahab Kaynama [a 1], Ian M. Mitchell [b], Meeko M. K. Oishi [c], Guy A. Dumont [a]

[a] *Electrical and Computer Engineering, University of British Columbia*
2332 Main Mall, Vancouver, BC, V6T 1Z4, CANADA

[b] *Computer Science, University of British Columbia*
2366 Main Mall, Vancouver, BC, V6T 1Z4, CANADA

[c] *Electrical and Computer Engineering, University of New Mexico*
MSC01 1100, 1 University of New Mexico, Albuquerque, NM 87131 USA

Abstract

While a number of Lagrangian algorithms to approximate reachability in dozens or even hundreds of dimensions for systems with linear dynamics have recently appeared in the literature, no similarly scalable algorithms for approximating viable sets have been developed. In this paper we describe a connection between reachability and viability that enables us to compute the viability kernel using reach sets. This connection applies to any type of system, such as those with nonlinear dynamics and/or non-convex state constraints; however, here we take advantage of it to construct three viability kernel approximation algorithms for linear systems with convex input and state constraint sets. We compare the performance of the three algorithms, and demonstrate that the two based on highly scalable Lagrangian reachability—those using ellipsoidal and support vector set representations—are able to compute the viability kernel for linear systems of larger state dimension than was previously feasible using traditional Eulerian methods. Our results are illustrated on a 6-dimensional pharmacokinetic model and a 20-dimensional model of heat conduction on a lattice.

Key words: Viability; Reachability; Controlled invariance; Set-theoretic methods; High-dimensional systems; Formal verification; Safety-critical systems.

1 Introduction

Viability theory plays an important role in safety verification for control systems (cf. Aubin, Bayen, and Saint-Pierre, 2011), a particularly important problem for high risk, expensive, or safety-critical applications. In many engineered systems, input constraints limit the system's ability to remain within a desired "safe" region of operation. Consider, for example, problems in aerodynamic flight envelope protection (Tomlin, Mitchell, Bayen, and Oishi, 2003) or underwater vehicle operation under constraints (Panagou, Margellos, Summers, Lygeros, and Kyriakopoulos, 2009). For such systems, constraints on the state space determine the "safe set". However, because the control authority for the system is also constrained, there are some configurations in the safe set for which the state may inevitably exit. Hence it is important to identify the subset of the safe set for which the existence of a control input that keeps the state of the system within the safe region can be guaranteed.

This subset, known as the viability kernel (or the controlled invariant set), takes into account the system's dynamics and bounded control authority. For a constraint set K, the viability kernel $Viab(K)$ is the subset of K for which a control input exists that keeps the state of the system within K for the duration of a known (possibly infinite) time horizon.

[⋆] This paper was not presented at any IFAC meeting. Corresponding author J. Maidens.
[1] Currently with Electrical Engineering & Computer Sciences, University of California, Berkeley, 253 Cory Hall, Berkeley, CA 94720 USA

Email addresses: `jmaidens@ece.ubc.ca` (John N. Maidens), `kaynama@ece.ubc.ca` (Shahab Kaynama), `mitchell@cs.ubc.ca` (Ian M. Mitchell), `oishi@unm.edu` (Meeko M. K. Oishi), `guyd@ece.ubc.ca` (Guy A. Dumont).

The viability kernel has traditionally been approximated using Eulerian methods such as the Viability Kernel Algorithm (Saint-Pierre, 1994) and level set approaches (Mitchell, Bayen, and Tomlin, 2005). However, Eulerian methods require gridding the state space and hence their time and memory complexity grow exponentially with the state dimension. In practice, this approach is infeasible for systems with more than 3 or 4 states. Lagrangian methods have been applied previously to the computation of viability kernels, for example in Blanchini and Miani (2008), but the implementation has relied on polyhedral set representations that also do not scale well with the number of states.

There has been recent work to compute the viability kernel for high-dimensional systems based on simulated annealing (Bonneuil, 2006), approximate dynamic programming (Coquelin, Martin, and Munos, 2007) and supervised classification (Deffuant, Chapel, and Martin, 2007). The simulated annealing method has been demonstrated for a chain of integrators in 10 dimensions, taking 22 minutes to compute each point on viability kernel's boundary. The dynamic programming method has been demonstrated on a 4-dimensional system, taking 163 seconds to compute a grid of 2×10^5 points. The supervised classification method has been demonstrated on an ecological model with 51 inputs and 6 states (Chapel, Deffuant, Martin, and Mullon, 2008) but still relies on a gridding of the state space hence its applicability to systems with a large number of *states* is limited. Our results in Section 3.3.2 show a substantial improvement over existing methods in terms of scalability in the state dimension.

Lagrangian methods have been applied successfully to the computation of reachable sets (Kurzhanski and Varaiya, 2000a; Chutinan and Krogh, 2003; Le Guernic and Girard, 2010). In contrast to Eulerian methods, Lagrangian methods use representations that follow the vector field's flow. Since Lagrangian methods do not depend on gridding the state space, it is computationally feasible to analyse high-dimensional systems.

In Section 2, we present a connection between the viability kernel and reachable sets that allows the large class of methods developed for reachability analysis to be applied to the computation of viability kernels. It can be used for any system and set representation which supports the backward maximal reach set and intersection operations (or underapproximations thereof), in theory including nonlinear dynamics and/or non-convex constraints.

In Section 3, we restrict our attention to discrete-time linear systems under convex input and state constraints, a case for which a wealth of efficient Lagrangian reachability techniques exist. We use the results from Section 2 to provide three examples of Lagrangian algorithms for computing the viability kernel, and we compare these three algorithms. The polytope method performs well in terms of accuracy but does not scale well as the state dimension grows, becoming infeasible in greater than four dimensions. In comparison, the time complexity of the ellipsoidal method increases more slowly with the state dimension, but its accuracy is limited. The support vector method strikes a balance between scalability and accuracy. It allows the user to choose a desired accuracy in terms of the number of points on the boundary of the viability kernel that they wish to evaluate. We demonstrate empirically that the runtime of the ellipsoidal and support vector methods appear to be polynomial in state dimension.

While the three algorithms presented in Section 3 apply only to discrete-time systems, the techniques developed in this paper can equally be applied to continuous-time systems, provided that we have a method of computing (or under-approximating) continuous-time reach sets. As an example, in the conference paper (Kaynama, Maidens, Mitchell, Oishi, and Dumont, 2012) we use the continuous-time techniques developed in Section 2.2.2 to under-approximate the viability kernel of a continuous-time system using ellipsoidal techniques.

In Section 4, we provide two applications of our results. We compute the viability kernel for a 6-dimensional discrete-time model of Propofol pharmacokinetics in children, and a 20-dimensional discretized heat equation.

2 Establishing connections between viability and reachability

There is a close relationship between viability theory (Aubin, Bayen, and Saint-Pierre, 2011) and constrained reachability (Kurzhanski and Varaiya, 2001). Both frameworks study the evolution of dynamic systems under input and/or state constraints. The relationship between the two theories is often discussed in the context of optimal control theory by formulating both reachability and viability problems in terms of Hamilton-Jacobi equations, for example (Lygeros, 2004).

The Hamilton-Jacobi approach has proven extremely successful in the analysis of low-dimensional systems. Level set methods can be used to approximate the viscosity solution of the Hamilton-Jacobi PDE corresponding to a given viability or reachability problem (Tomlin, Mitchell, Bayen, and Oishi, 2003). Tools are available for computing viable and reachable sets numerically (Mitchell and Templeton, 2005) but they scale poorly with state dimension.

The recent emergence of accurate and scalable methods and tools for approximating reachable sets in high-dimensional systems (Kurzhanski and Varaiya, 2000a;

Frehse, Le Guernic, Donzé, Cotton, Ray, Lebeltel, Ripado, Girard, Dang, and Maler, 2011) has inspired us to attempt to find analogous methods for the approximation of viability kernels. In this section, we expose a connection between viability theory and reachability theory. The results presented here appeared in preliminary form in a conference paper (Kaynama, Maidens, Mitchell, Oishi, and Dumont, 2012).

2.1 Preliminaries

We are concerned with analysing systems of the form

$$\begin{cases} \mathcal{L}(x(t)) = f(x(t), u(t)) \\ u(t) \in \mathcal{U} \end{cases} \tag{1}$$

where the time t ranges throughout a *time domain* $\mathbb{T}$. The time domain $\mathbb{T}$ can be either *continuous* ($\mathbb{T} = [0, \tau] \subseteq \mathbb{R}_+$) or *discrete* ($\mathbb{T} = [0, \tau] \cap \mathbb{Z}_+$). If $0 < \tau < \infty$ this problem is said to have a *finite horizon*; otherwise, if $\tau = \infty$, it is said to have an *infinite horizon.* $\mathcal{L}$ is the differential operator corresponding to the given time domain (differentiation in the case of a continuous-time system and differencing in the case of a discrete-time system). The system's *state* x ranges over the finite-dimensional vector space $\mathbb{R}^d$ and the system's *input* is constrained to a nonempty, compact, convex subset $\mathcal{U} \subseteq \mathbb{R}^m$.[1] When (1) evolves under continuous time, we assume that the function $f : \mathbb{R}^d \times \mathcal{U} \to \mathbb{R}^d$ is sufficiently smooth to guarantee the existence and uniqueness of solutions to the corresponding initial value problem.

Viability theory is concerned with ensuring that a system's state x remains within a set of *viability constraints* $K \subseteq \mathbb{R}^d$. Any trajectory of system (1) that leaves the set K at some point in time is considered to be no longer *viable.*

We call a set S viable under K if for every initial state $x_0 \in S$ there exists some measurable input $u_0 : \mathbb{T} \to \mathcal{U}$ such that the solution $x(\cdot)$ to the initial value problem

$$\begin{cases} \mathcal{L}(x(t)) = f(x(t), u_0(t)) \\ x(0) = x_0 \end{cases} \tag{2}$$

satisfies $x(t) \in K$ for all $t \in \mathbb{T}$.

The *viability kernel* of a set of viability constraints K is the largest viable set contained in K. Equivalently, the viability kernel is defined as follows:

$$Viab_{\mathbb{T}}(K) = \{x_0 \in K \mid \exists u_0 : \mathbb{T} \to \mathcal{U}\ \forall t \in \mathbb{T}\ x(t) \in K\}.$$

[1] This is not the most general context in which viability theory can be developed. Aubin (1991) allows the constraint set $\mathcal{U}$ to depend on the state x.

The related constructs of constrained reachability analysis are a popular technique for formal safety verification (for example Mitchell, 2007). They provide a method of simulating all possible trajectories of a dynamic system under all admissible inputs. Essentially, they are concerned with determining if any trajectories of the system (1) that begin in a set of initial conditions I can reach a set of terminal states T.

There are two ways to approach the problem of reachability analysis. We can begin by considering the set of initial states and follow this set forward in time under the flow of (1) to compute what is known as the *forward reachable set.* The other approach considers the set of terminal states T and follows the flow of (1) backward in time to compute the *backward reachable set.*

For our purposes, it is appropriate to use the backward approach. We define the set backward reachable from T over a time domain $\mathbb{T}$ with finite horizon τ as follows:

$$Reach_\tau(T) = \{x_0 \in \mathbb{R}^d \mid \exists u_0 : \mathbb{T} \to \mathcal{U}\ \ x(\tau) \in T\}.$$

2.2 Computing viability kernels using reachability techniques

In this section, we present a method of expressing finite horizon viability kernels in terms of reachable sets. This provides a modified version of Saint-Pierre's Viability Kernel Algorithm that can be implemented using efficient and scalable techniques developed within the context of reachability analysis. We present the algorithm first for discrete-time systems then for continuous-time systems. In the discrete-time case the viability kernel is computed exactly, while in the continuous-time case we compute an under-approximation of the true viability kernel.

Throughout this paper, whenever we cannot compute a set exactly we ensure that our approximation is an under-approximation. This guarantees that the set we compute to approximate $Viab(K)$ is indeed viable under K.

2.2.1 Discrete-time systems

We consider the case when the system (1) evolves in discrete time. The system's dynamics can be described by the constrained difference equation

$$\begin{cases} x(t+1) = f(x(t), u(t)) \\ u(t) \in \mathcal{U}. \end{cases} \tag{3}$$

For discrete-time systems, the viability kernel can be computed using Saint-Pierre's Viability Kernel Algorithm via the following recursive formula that gives the finite-horizon viability kernel $K_{n+1} := Viab_{[0,n+1]\cap\mathbb{Z}_+}(K)$ in terms of the finite-horizon viability kernel K_n at the previous step (Saint-Pierre, 1994):

$$\begin{cases} K_0 = K \\ K_{n+1} = \{x \in K_n \mid K_n \cap F(x) \neq \emptyset\} \end{cases} \tag{4}$$

where $F(x) = \{f(x,u) \mid u \in \mathcal{U}\}$. We can reformulate this recursive definition of the finite horizon viability kernels K_n in terms of the backward reach set over one discrete timestep $Reach_1(\cdot)$:

Theorem 1 *The sequence of finite horizon viability kernels K_n can be computed recursively in terms of reach sets as*

$$\begin{cases} K_0 = K \\ K_{n+1} = K_0 \cap Reach_1(K_n). \end{cases} \tag{5}$$

Proof. Let the constrained difference equation (3) be expressed as the difference inclusion $x(t+1) \in F(x(t))$. Then using the definition of K_{n+1},

$$\begin{aligned} x \in K_{n+1} &\iff x \in K_n \wedge (F(x) \cap K_n \neq \emptyset) \\ &\iff x \in K_n \wedge \exists y (y \in F(x) \wedge y \in K_n) \\ &\iff x \in K_n \wedge \exists y (\exists u \in \mathcal{U} \;\; y = f(x,u) \wedge y \in K_n) \\ &\iff x \in K_n \wedge \exists u \in \mathcal{U} \;\; f(x,u) \in K_n \\ &\iff x \in K_n \wedge x \in Reach_1(K_n) \\ &\iff x \in K_n \cap Reach_1(K_n). \end{aligned}$$

Thus $K_{n+1} = K_n \cap Reach_1(K_n)$. We prove that $K_n \cap Reach_1(K_n) = K_0 \cap Reach_1(K_n)$ by induction. The base $K_0 \cap Reach_1(K_0) = K_0 \cap Reach_1(K_0)$ is clear. It follows from $K_{n+1} \subseteq K_n$ that $K_0 \cap Reach_1(K_{n+1}) \subseteq K_0 \cap Reach_1(K_n) = K_n \cap Reach_1(K_n) = K_{n+1}$. Hence we have the first inclusion $K_0 \cap Reach_1(K_{n+1}) \subseteq K_{n+1} \cap Reach_1(K_{n+1})$. The opposite inclusion follows from the fact that $K_{n+1} \subseteq K_0$. □

Though they are both equivalent when the reach sets and intersections can be computed exactly, we intersect $Reach_1(K_n)$ with K_0 rather than K_n because this leads to better behaviour in implementations where sets must be under-approximated. Since the set K_0 is given as input to the algorithm, it is more accurate and its representation is typically simpler than the computed set K_n with $n > 0$.

The recursive formula given in Theorem 1 leads to Algorithm 1 for computing the finite horizon viability kernel over the discrete time interval $\mathbb{T} = \{t \in \mathbb{Z}_+ \mid t \leq N\}$.

Algorithm 1 Exact computation of the viability kernel (discrete-time)

```
$K_0 \leftarrow K$
$n \leftarrow 0$
while $n \leq N$ do
    if $K_n = \emptyset$ then                ▷ If true, $Viab_{\mathbb{T}}(K) = \emptyset$
        $K_N \leftarrow \emptyset$
        break
    end if
    if $K_n = K_{n-1}$ then                  ▷ If true, $Viab_{\mathbb{T}}(K) = K_n$
        $K_N \leftarrow K_n$
        break
    end if
    $L \leftarrow Reach_1(K_n)$
    $K_{n+1} \leftarrow K_0 \cap L$
    $n \leftarrow n + 1$
end while
return $(K_N)$                               ▷ $K_N = Viab_{\mathbb{T}}(K)$
```

2.2.2 Continuous-time systems

We now consider the case when the system (1) evolves in continuous time. In this case, the system's dynamics are described by the constrained differential equation

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ u(t) \in \mathcal{U}. \end{cases} \tag{6}$$

Before we can present our algorithm, we require a few definitions.

We say that a vector field $f : \mathbb{R}^d \times \mathcal{U} \to \mathbb{R}^d$ is bounded by $M > 0$ on $K \subseteq \mathbb{R}^d$ in the norm $||\cdot|| : \mathbb{R}^d \to \mathbb{R}_+$ if for all $x \in K$ and $u \in \mathcal{U}$ we have $||f(x,u)|| \leq M$. We also define the $||\cdot||$-distance of a point $x \in \mathbb{R}^d$ from a nonempty set $S \subset \mathbb{R}^d$ as

$$\text{dist}_{||\cdot||}(x, S) = \inf_{s \in S} ||x - s||. \tag{7}$$

Computing an under-approximation of the viability kernel

Let K be the set of viability constraints for the constrained differential equation (6). We assume that the vector field f is bounded by M on K in the norm $||\cdot||$. Given a discretization time interval ρ, we begin by defining an under-approximation of the viability constraint set (Figure 1a):

$$K_\rho := \{x \in K \mid \text{dist}_{||\cdot||}(x, K^c) \geq \rho M\}. \tag{8}$$

We under-approximate K by a distance ρM because we are only considering the system's state at discrete times

$t_n = n\rho$. At a time t in the interval $[t_n, t_{n+1}]$, a solution $x(\cdot)$ of (6) can travel a distance of at most

$$||x(t_n) - x(t)|| \leq \int_{t_n}^{t} ||\dot{x}(s)|| ds \leq M(t - t_n) \leq \rho M$$

from its initial location $x(t_n)$. Thus the under-approximation (8) ensures that the state does not leave K at any time in the interval $[t_n, t_{n+1}]$.

We proceed by defining a sequence of sets $\{K_n(\rho)\}$ analogously to the discrete-time case. The under-approximation K_ρ of the viability constraints is the base case. We then define subsequent sets $\{K_n(\rho)\}$ recursively:

$$\begin{cases} K_0(\rho) = K_\rho \\ K_{n+1}(\rho) = K_0(\rho) \cap Reach_\rho(K_n(\rho)). \end{cases} \tag{9}$$

At each time step, we calculate the set of states from which $K_n(\rho)$ is reachable, then intersect this set with the set of safe states. This process is illustrated in Figure 1. Each computed set $K_n(\rho)$ is an approximation of the finite horizon viability kernel $Viab_{[0,\tau]}(K)$ for $\tau = n\rho$. Note that the resulting set depends on our choice of the time step ρ. We claim that for any $\rho > 0$, $K_n(\rho)$ under-approximates $Viab_{[0,n\rho]}(K)$.

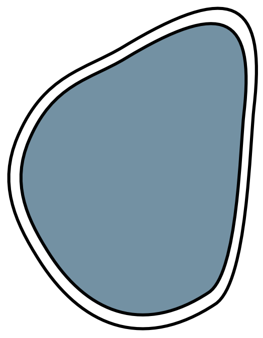
Theorem 2 *Suppose that the vector field $f : \mathbb{R}^d \times \mathcal{U} \to \mathbb{R}^d$ is bounded by $M > 0$ on a set $K \subseteq \mathbb{R}^d$ in the norm $||\cdot||$. Then for any time step ρ the sets $\{K_n\}$ defined by Equations (8) and (9) satisfy*

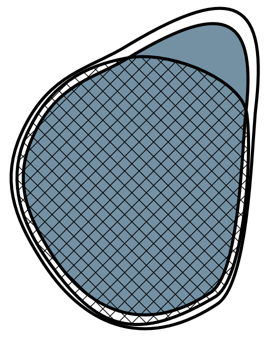
$$K_n(\rho) \subseteq Viab_{[0,n\rho]}(K). \tag{10}$$

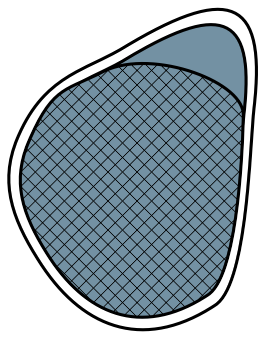
Proof. Since f is bounded by M on K, $||f(x,u)|| \leq M$ for all $x \in K$. Now, take a point $x_0 \in K_n(\rho)$. By the construction of $K_n(\rho)$, this means that for each $k = 1, \ldots, n$ there is some point $x_k \in K_k(\rho)$ and an input $u_k : [0,\rho] \to \mathcal{U}$ such that x_k can be reached from x_{k-1} at time ρ using input u_k. Thus, taking the concatenation of the inputs u_k, we get an input $u : [0, n\rho] \to \mathcal{U}$ such that the solution $x : [0, n\rho] \to \mathbb{R}^d$ to the initial value problem $\dot{x} = f(x,u)$, $x(0) = x_0$, satisfies $x(k\rho) = x_k \in K_k(\rho) \subseteq \{x \in K \mid \text{dist}_{||\cdot||}(x, K^c) \geq M\rho\}$. We claim that this guarantees that $x(t) \in K$ for all $t \in [0, n\rho]$. Indeed, any $t \in [0, n\rho)$ lies in some interval $[t_k, t_{k+1}) = [k\rho, (k+1)\rho)$. Since f is bounded by M, we have

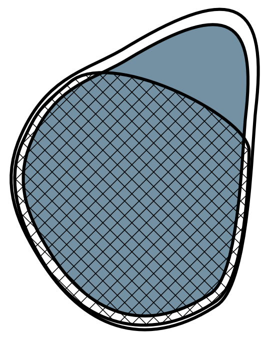
$$||x(t_k) - x(t)|| \leq \int_{t_k}^{t} ||\dot{x}(s)|| ds \leq M(t - t_k) \leq \rho M$$

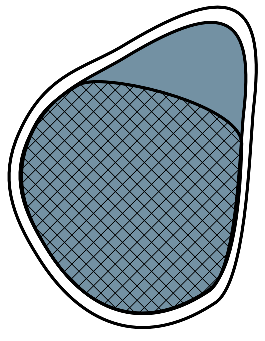
Further, $x(t_k) \in K_k(\rho)$ implies that $\text{dist}_{||\cdot||}(x(t_k), K^c) \geq$


(a) We define the initial under-approximation of the safe set $K_0(\rho) = K_\rho$.


(b) We calculate the set of backward reachable states from $K_0(\rho)$.


(c) We intersect the backward reachable set with the initial set to get $K_1(\rho)$


(d) Next, we calculate the set of backward reachable states from $K_1(\rho)$.


(e) Again, we intersect the backward reachable set with the initial set to get a new set $K_2(\rho)$

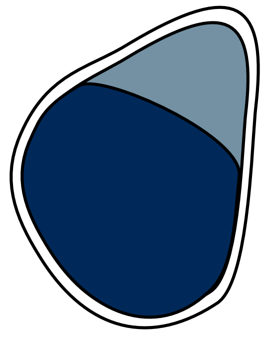

(f) By repeating this process, we eventually reach an under-approximation $K_n(\rho)$ of the viability kernel.

Fig. 1. Iteratively constructing an under-approximation of $Viab_{[0,\tau]}(K)$.

ρM. Combining these, we see that

$$\begin{aligned} \text{dist}_{||\cdot||}(x(t), K^c) &\geq \text{dist}_{||\cdot||}(x(t_k), K^c) - ||x(t) - x(t_k)|| \\ &> \rho M - \rho M = 0 \end{aligned}$$

and hence $x(t) \in K$. Thus, $x_0 \in Viab_{[0,n\rho]}(K)$. $\square$

Accuracy of the Approximation

By choosing a sufficiently small time step, the approximation can be made arbitrarily accurate in the following sense: fix a time horizon τ and partition the interval $[0,\tau]$ into N subintervals $[t_{n-1}, t_n]$ of width $\rho_N = \tau/N$. The union of the approximating sets $K_N(\rho_N)$ taken over all $N \in \mathbb{N}$ is bounded between the viability kernels of K and its interior $\mathring{K}$.

Theorem 3 *Suppose that the vector field $f : \mathbb{R}^d \times \mathcal{U} \to \mathbb{R}^d$ is bounded by M on a set $K \subseteq \mathbb{R}^d$. Then we have*

$$Viab_{[0,\tau]}(\mathring{K}) \subseteq \bigcup_{N\in\mathbb{N}} K_N(\rho_N) \subseteq Viab_{[0,\tau]}(K). \qquad (11)$$

Proof. The second inclusion $\bigcup_{N\in\mathbb{N}} K_N(\rho_N) \subseteq Viab_{[0,\tau]}(K)$ follows directly from Theorem 2. To prove the first inclusion, take $x_0 \in Viab_{[0,\tau]}(\mathring{K})$. Then there exists an input $u : [0,\tau] \to \mathcal{U}$ such that the solution $x(\cdot)$ to the initial value problem $\dot{x} = f(x,u)$, $x(0) = x_0$, satisfies $x(t) \in \mathring{K}$ for all $t \in [0,\tau]$. Since $\mathring{K}$ is open, for any $x \in \mathring{K}$ we have $\text{dist}_{||\cdot||}(x, K^c) > 0$. Further, $x : [0,\tau] \to \mathbb{R}^d$ is continuous so the function $t \mapsto \text{dist}_{||\cdot||}(x(t), K^c)$ is continuous on the compact set $[0,\tau]$. Thus, we can define $d > 0$ to be its minimum value. Now take N large enough such that $\rho_N < d/M$. We need to show that $x_0 \in K_N(\rho_N) = K_0(\rho_N) \cap Reach_{[0,\rho_N]}(K_{N-1}(\rho_N))$.

First note that N is chosen such that $\text{dist}_{||\cdot||}(x(t), K^c) > \rho_N M$ for all $t \in [0,\tau]$. Hence $x(t_{N-n}) \in K_0(\rho_N)$ for all $n = 0,\dots,N$. To show that $x(t_{N-n}) \in Reach_{[0,\rho_N]}(K_{n-1}(\rho_N))$ for all $n = 1,\dots,N$, consider the sequence of inputs $u_n : [0,\rho_N] \to \mathcal{U}$ defined as

$$u_n(t) = u(t_{n-1} + t). \qquad (12)$$

It is easy to verify that for all n, we can reach $x(t_n)$ from $x(t_{n-1})$ at time t_n using input u_n. Thus, in particular, we have $x_0 = x(0) \in Reach_{[0,\rho_N]}(K_{N-1}(\rho_N))$. So $x_0 \in K_N(\rho_N)$. Hence $Viab_{[0,\tau]}(\mathring{K}) \subseteq \bigcup_{N\in\mathbb{N}} K_N(\rho_N)$. □

Corollary 1 *When K is open,*

$$\bigcup_{N\in\mathbb{N}} K_N(\rho_N) = Viab_{[0,\tau]}(K). \qquad (13)$$

Algorithm 2 computes an approximation of the viability kernel in the continuous-time case using the recursive formula (9). Theorem 2 guarantees that the computed set always under-approximates the true viability kernel while Theorem 3 guarantees that the approximation is asymptotically tight as the time step $\rho \to 0$.

Algorithm 2 Under-approximation of the viability kernel (continuous-time)

```
Choose $\rho > 0$   ▷ Determines approximation accuracy
$N \leftarrow \tau/\rho$   ▷ Number of time steps
$K_0 \leftarrow K_\rho$   ▷ Initial under-approximation of $K_0$
$n \leftarrow 0$
while $n \leq N$ do
    if $K_n = \emptyset$ then   ▷ If true, $Viab_{[0,\tau]}(K) = \emptyset$
        $K_N \leftarrow \emptyset$
        break
    end if
    if $K_n = K_{n-1}$ then   ▷ If true, $Viab_{[0,\tau]}(K) = K_n$
        $K_N \leftarrow K_n$
        break
    end if
    $L \leftarrow Reach_{[0,\rho_N]}(K_n)$
    $K_{n+1} \leftarrow K_0 \cap L$
    $n \leftarrow n + 1$
end while
return $(K_N)$   ▷ $K_N \subseteq Viab_{[0,\tau]}(K)$
```

3 Lagrangian algorithms for computing viability kernels in linear systems

In Section 2 we demonstrated that the viability kernel of an input-constrained dynamic system can be computed in terms of reachable sets. In this section we use this result to develop efficient algorithms for computing or approximating the viability kernel in high-dimensional discrete-time linear systems.

As the viability kernel is often used for safety verification, it is desirable that any approximations made are conservative so that the computed safe set under-approximates the true viability kernel. This way, we can guarantee that any point within the computed kernel is truly a safe initial state.

There are numerous Lagrangian algorithms for approximating reachable sets in discrete-time linear systems. These algorithms rely on particular geometric representations such as polytopes (Chutinan and Krogh, 2003), ellipsoids (Kurzhanski and Varaiya, 2000b), zonotopes (Girard, Le Guernic, and Maler, 2006), or support functions (Le Guernic and Girard, 2010). Each representation has advantages and disadvantages in terms of representation size, approximation fidelity and ease of performing geometric operations.

We begin by comparing various set representations. We then develop a number of practical implementations of Algorithm 1 and compare their performance on a pair of benchmark examples.

3.1 Set representations for linear reachability

We now restrict our attention to linear dynamics [2]

$$\begin{cases} \mathcal{L}(x(t)) = Ax(t) - v(t) \\ v(t) \in \mathcal{V}. \end{cases} \tag{14}$$

under compact, convex constraints K and $\mathcal{V}$. In the discrete-time case, the backward reachable set over a single time step is computed as

$$Reach(K) = A^{-1}(K \oplus \mathcal{V}). \tag{15}$$

Here $A^{-1}(\cdot)$ denotes the preimage of a set under the map $A : \mathbb{R}^d \to \mathbb{R}^d$. Throughout the remainder of this section, we will assume that A is non-singular, and thus the preimage of A can be calculated simply by applying the linear transformation A^{-1} to the set $K \oplus \mathcal{V}$. This is a fair assumption because we are mainly concerned with discrete-time systems that arise from the discretization of continuous time systems. Such systems have a dynamics matrix of the form $A = \exp(\rho A_c)$ which is always invertible.

As the operations performed on sets include Minkowski summation, linear transformation and intersection, the ideal set representation would be a class of objects closed under these three operations. We also hope for a representation under which all three operations can be performed accurately, efficiently and using a constant amount of memory.

3.1.1 Convex polytopes

A convex polytope $P \subset \mathbb{R}^d$ (hereafter simply polytope) is a bounded geometric object with flat sides. A polytope can be defined as the convex hull of a finite set of points $v_1, \cdots, v_k$. This is known as the vertex representation and in this case, P is called a $\mathcal{V}$-polytope. A polytope can also be defined as an intersection of half-spaces

$$P = \bigcap_{i=1}^{f} \{x \in \mathbb{R}^d \mid h_i \cdot x \leq b_i\} = \{x \in \mathbb{R}^d \mid Hx \leq b\}$$

This is known as the facet representation and in this case, P is known as an $\mathcal{H}$-polytope.

The problem of switching between the $\mathcal{H}$-representation and the $\mathcal{V}$-representation is known as the vertex/facet enumeration problem (Avis and Fukuda, 1992). It is a well-studied problem and many algorithms have been proposed to solve it. These algorithms tend to be slow, however, taking $O(dkf)$ time where k is the number of vertices and f the number of facets in the polytope.

Polytopes are a good choice of set representation for our purposes because the class of polytopes is closed under all three operations that we wish to perform. Let $P_1 = \{x \in \mathbb{R}^d \mid A_1 x \leq b_1\}$ and $P_2 = \{x \in \mathbb{R}^d \mid A_2 x \leq b_2\}$ be two polytopes described by their $\mathcal{H}$-representation. We can compute the image of P_1 under the linear transformation T as

$$TP_1 = \{x \in \mathbb{R}^d \mid A_1 T^{-1} x \leq b_1\}.$$

The intersection of P_1 and P_2 can also be easily computed simply by combining the constraints as follows

$$\begin{aligned} P_1 \cap P_2 &= \{x \in \mathbb{R}^d \mid A_1 x \leq b_1 \;\; and \;\; A_2 x \leq b_2\} \\ &= \left\{ x \in \mathbb{R}^d \; : \; \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} x \leq \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right\}. \end{aligned}$$

The Minkowski sum however is difficult to compute using the $\mathcal{H}$-representation and is typically done by first converting to the $\mathcal{V}$-representation.

Given two polytopes $P_1 = \mathrm{Conv}\{v_1, \cdots, v_k\}$ and $P_2 = \mathrm{Conv}\{u_1, \cdots, u_l\}$ their Minkowski sum is computed as

$$P_1 \oplus P_2 = \mathrm{Conv}\{v_i + u_j \mid i = 1, \cdots, k \;\; j = 1, \cdots, l\}.$$

The linear transformation of P_1 under T is also easily computed as

$$TP_1 = \mathrm{Conv}\{Tv_1, \cdots, Tv_k\}.$$

The disadvantage of the $\mathcal{V}$-representation is that it is difficult to perform intersections.

The Multi-Parametric Toolbox (Kvasnica, Grieder, Baotić, and Morari, 2004) provides a comprehensive set of tools for computations using polytopes.

3.1.2 Ellipsoids

An ellipsoid $\mathcal{E} \subset \mathbb{R}^d$ is a smooth geometric object contained by a bounded quadratic surface. Any ellipsoid $\mathcal{E} \subset \mathbb{R}^d$ can be expressed as the image of the Euclidean unit ball under an affine transformation $T : \mathbb{R}^d \to \mathbb{R}^d$. $\mathcal{E}$ is called nondegenerate if the transformation T is nondegenerate (*i.e.* invertible). Ellipsoids can also be defined uniquely by a symmetric, positive definite shape matrix $\mathcal{Q}$ and a centre $q \in \mathbb{R}^d$ as

$$\mathcal{E}(q, \mathcal{Q}) = \{x \in \mathbb{R}^d \mid (x - q) \cdot \mathcal{Q}^{-1}(x - q) \leq 1\}.$$

The class of ellipsoids is closed under nondegenerate linear transformations but it is not closed under either

[2] A general linear system $\mathcal{L}(x(t)) = Ax(t) + Bu(t) \;\; u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ can be written in this form by setting $\mathcal{V} = -B\mathcal{U} = \{-Bu | u \in \mathcal{U}\} \subseteq \mathbb{R}^d$.

Minkowski summation or intersection. However, the Ellipsoidal Toolbox (Kurzhanskiy and Varaiya, 2006) provides a set of routines to efficiently compute tight under- and over-approximations of Minkowski sums and intersections of ellipsoids.

3.1.3 Support functions

An arbitrary compact, convex set $\mathcal{A} \subset \mathbb{R}^d$ can be represented in terms of its support function. The support function $\sigma_{\mathcal{A}} : \mathbb{R}^d \to \mathbb{R}$ of $\mathcal{A}$ is a convex function defined as

$$\sigma_{\mathcal{A}}(\ell) = \max_{x \in \mathcal{A}} x \cdot \ell. \tag{16}$$

The support function $\sigma_{\mathcal{A}}$ is a complete representation of $\mathcal{A}$ in the sense that $\mathcal{A}$ can be reconstructed from $\sigma_{\mathcal{A}}$ as the intersection of all its supporting half spaces

$$\mathcal{A} = \bigcap_{\ell \in \mathbb{R}^d} \{x \in R^d \mid x \cdot \ell \leq \sigma_{\mathcal{A}}(\ell)\}. \tag{17}$$

Support functions are convenient for our purposes because all three operations that we wish to perform can be performed directly on the support functions. This fact is given in Theorem 4.

Theorem 4 *Let $\mathcal{A} \subset \mathbb{R}^d$ and $\mathcal{B} \subset \mathbb{R}^d$ be compact, convex sets and let $A : \mathbb{R}^d \to \mathbb{R}^d$ be a linear transformation represented by a matrix A. We have the following properties:*

- $\sigma_{A\mathcal{B}}(\ell) = \sigma_{\mathcal{B}}(A^T \ell)$
- $\sigma_{\mathcal{A} \oplus \mathcal{B}}(\ell) = \sigma_{\mathcal{A}}(\ell) + \sigma_{\mathcal{B}}(\ell)$
- $\sigma_{\mathcal{A} \cap \mathcal{B}}(\ell) = \inf_{w \in \mathbb{R}^d} \{\sigma_{\mathcal{A}}(\ell - w) + \sigma_{\mathcal{B}}(w)\}$.

Proof. The first two properties follow directly from the definition. The proof of the third is given in (Rockafellar and Wets, 1998). □

Thus given support functions for $\mathcal{A}$ and $\mathcal{B}$, it is simple to compute the support functions of $\mathcal{A} \oplus \mathcal{B}$, $A\mathcal{B}$ and $\mathcal{A} \cap \mathcal{B}$.

A compact, convex set $\mathcal{A} \subset \mathbb{R}^d$ can be over-approximated by an $\mathcal{H}$-polytope with arbitrary accuracy by sampling its support function. Consider a finite set of vectors $\mathcal{L} \subset \mathbb{R}^d$. Following (Le Guernic, 2009), we can define an over-approximation of $\mathcal{A}$ by restricting the intersection in (17) to the set $\mathcal{L}$, giving us

$$\mathcal{A}_{\uparrow} = \bigcap_{\ell \in \mathcal{L}} \{x \in R^d \mid x \cdot \ell \leq \sigma_{\mathcal{A}}(\ell)\}.$$

This over-approximation is tight (*i.e.* the approximation touches the boundary of $\mathcal{A}$) in the directions of $\mathcal{L}$. An example of this approximation is shown in Figure 2.

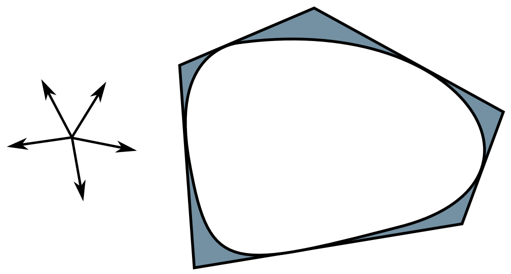


Fig. 2. A compact, convex set $\mathcal{A}$ (white) and the corresponding tight over-approximation (grey) in the set of directions $\mathcal{L}$ shown at left.

3.1.4 Support vectors

When computing viability kernels, it is usually desirable to under-approximate rather than over-approximate a given set. Again following (Le Guernic, 2009), for a compact, convex set $\mathcal{A} \subset \mathbb{R}^d$ we can construct an under-approximation of $\mathcal{A}$ using support vectors.

Given a direction vector $\ell \in \mathbb{R}^d$ the set of support vectors of $\mathcal{A}$ is defined as

$$v_{\mathcal{A}}(\ell) = \operatorname*{argmax}_{x \in \mathcal{A}} x \cdot \ell. \tag{18}$$

The support vectors and support function of a convex set are related via the subgradient operation ∂ (see Appendix) as $v_{\mathcal{A}}(\ell) = \partial \sigma_{\mathcal{A}}(\ell)$. In particular, when $\sigma_{\mathcal{A}}$ is differentiable at ℓ, the set of support vectors in the direction ℓ is the singleton set $v_{\mathcal{A}}(\ell) = \{\nabla \sigma_{\mathcal{A}}(\ell)\}$.

As was the case for support functions, all three operations that we wish to perform can be performed directly on the support vectors. This result is given in Theorem 5.

Theorem 5 *Let $\mathcal{A} \subset \mathbb{R}^d$ and $\mathcal{B} \subset \mathbb{R}^d$ be compact, convex sets and let $A : \mathbb{R}^d \to \mathbb{R}^d$ be a linear transformation represented by a matrix A. We have the following properties:*

- $v_{A\mathcal{A}}(\ell) = A v_{\mathcal{A}}(A^T \ell)$
- $v_{\mathcal{A} \oplus \mathcal{B}}(\ell) = v_{\mathcal{A}}(\ell) \oplus v_{\mathcal{B}}(\ell)$
- $v_{\mathcal{A} \cap \mathcal{B}}(\ell) = v_{\mathcal{A}}(\ell - \bar{w}) \cap v_{\mathcal{B}}(\bar{w})$

where $\bar{w} \in \arg\inf_{w \in \mathbb{R}^d} \{\sigma_{\mathcal{A}}(\ell - w) + \sigma_{\mathcal{B}}(w)\}$.

Proof. Again, the first two properties follow directly from the definition. The proof of the third requires some background in convex analysis and is given in the Appendix. □

As was the case for support functions, the set $\mathcal{A}$ can be

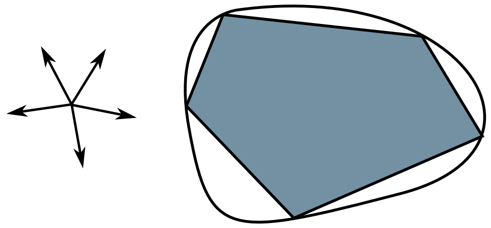


Fig. 3. A compact, convex set $\mathcal{A}$ (white) and the corresponding tight under-approximation (grey) in the set of directions $\mathcal{L}$ shown at left.

reconstructed from its support vectors:

$$\mathcal{A} = \text{Conv}\left(\bigcup_{\ell \in \mathbb{R}^d} v_{\mathcal{A}}(\ell)\right).$$

Now, given a subset $\mathcal{L}$ of directions, we can define an under-approximation of $\mathcal{A}$ that is tight in the directions $\mathcal{L}$ by selecting a support vector $u_\ell \in v_{\mathcal{A}}(\ell)$ in each direction $\ell \in \mathcal{L}$. An under-approximation of $\mathcal{A}$ is then given by

$$\mathcal{A}_\downarrow = \text{Conv}(\{u_\ell \mid \ell \in \mathcal{L}\}).$$

This approximation is illustrated in Figure 3.

3.2 *Algorithms*

We now present three algorithms that compute an approximation of the viability kernel of the discrete-time system

$$\begin{cases} x(t+1) = Ax(t) - v(t) \\ v(t) \in \mathcal{V}. \end{cases} \tag{19}$$

3.2.1 *Exact polytopic method*

The class of polytopes is closed under linear transformation, Minkowski summation and intersection. Therefore when the input constraint set $\mathcal{V}$ and the viability constraints K are both polytopes, we can compute the viability kernel of the system (19) exactly.

This method is similar to other methods of computing viability kernels and controlled invariant sets as described in (Blanchini and Miani, 2008) and implemented in the Multi-Parametric Toolbox (Kvasnica, Grieder, Baotić, and Morari, 2004).

Algorithm 3 computes the viability kernel using Algorithm 1 with *Reach*(K_n) computed using (15).

Algorithm 3 Exact polytopic method

```
$K_0 \leftarrow K$
$n \leftarrow 0$
while $n \leq N$ do
    if $K_n = \emptyset$ then
        $K_N \leftarrow \emptyset$
        break
    end if
    if $K_n = K_{n-1}$ then
        $K_N \leftarrow K_n$
        break
    end if
    $L \leftarrow A^{-1}(K_n \oplus \mathcal{V})$
    $K_{n+1} \leftarrow K_0 \cap L$
    $n \leftarrow n + 1$
end while
return $(K_N)$                    $\triangleright K_N = Viab_{\mathbb{T}}(K)$
```

Since no approximations are made, the accuracy of this algorithm is perfect. However, the amount of information required to represent the polytope K_n increases exponentially with successive Minkowski sums as the number of vertices of $K_n \oplus \mathcal{U}$ is (in the worst case) $|V(K_n)| \cdot |V(\mathcal{U})|$. A possible remedy to this problem is to under-approximate the polytope generated at each step by a polytope of fixed complexity (Kanade, Alur, Ivancic, Ramesh, Sankaranarayanan, and Shashidhar, 2009).

3.2.2 *Ellipsoidal method*

Using ellipsoids provides an approach to keeping the complexity of our set representation constant. This leads to a scalable method of computing an under-approximation of the viability kernel over large time horizons in high-dimensional spaces.

The details of this algorithm are given in (Kaynama, Maidens, Mitchell, Oishi, and Dumont, 2012) and are not repeated here. The issue with this algorithm is that the class of ellipsoids is closed under neither Minkowski sum nor intersection. Hence both must be under-approximated, leading to a reduction in accuracy.

3.2.3 *Support vector method*

We now present a method of under-approximating the viability kernel using support vectors. This method is based on ideas developed in (Le Guernic, 2009) which uses support functions to compute an over-approximation of reachable sets. The computation of the viability kernel presents an additional challenge compared with the computation of reach sets in that intersections must also be performed.

We first present a method of computing the support function of the viability kernel in a given direction by finding the solution to a convex optimization problem. The solution to this optimization can then be used to compute a support vector in the same direction by means of a recursive formula.

Approximating the viability kernel using support functions and support vectors provides an advantage over ellipsoids in terms of accuracy. Given an arbitrary direction $\ell \in \mathbb{R}^d$, the support function method allows us to find a hyperplane tangent to the viability kernel in the direction ℓ. Performing this computation in multiple directions ℓ_k allows us to over-approximate the discrete-time viability kernel as the intersection of half-spaces bounded by the tangent hyperplanes. Similarly, we can compute a tight under-approximation of the viability kernel as the convex hull of a set of support vectors in the directions ℓ_k. This procedure can be made arbitrarily accurate simply by choosing a sufficient number of directions ℓ_k.

The support vector method has an advantage over the polytope method in terms of scalability. After presenting Algorithm 4, we demonstrate its scalability experimentally using a chain of integrators of varying length.

By Theorem 4 we can express the value of the support function of $K_{n+1} = A^{-1}(K_n \oplus \mathcal{V}) \cap K_0$ in the direction ℓ as

$$\begin{aligned}\sigma_{K_{n+1}}(\ell) = \inf_{w\in\mathbb{R}^d} \{&\sigma_{K_0}(\ell - w) \\ &+ \sigma_{\mathcal{V}}(A^{-T}w) + \sigma_{K_n}(A^{-T}w)\} \qquad (20)\end{aligned}$$

where $A^{-T} = (A^T)^{-1}$ is the inverse transpose of A. The function $w \mapsto \sigma_{K_0}(\ell - w) + \sigma_{\mathcal{V}}(A^{-T}w) + \sigma_{K_n}(A^{-T}w)$ is convex so if the functions σ_{K_0}, $\sigma_{\mathcal{V}}$ and σ_{K_n} could be evaluated in constant time, this problem could be solved efficiently. However, since σ_{K_n} in turn depends on $\sigma_{K_{n-1}}$, a naive implementation of this formula would result in a number of calls to σ_{K_0} that is exponential in n.

This problem can be avoided by writing a closed-form expression for σ_{K_n}.

Theorem 6 *The value of the support function σ_{K_n} in the direction ℓ can be expressed as the solution to a convex optimization over an nd-dimensional space. It is given by*

$$\sigma_{K_n}(\ell) = \inf_{w\in\mathbb{R}^{nd}} \xi(\ell, w)$$

where

$$\begin{aligned}\xi(\ell, w) &= \xi(\ell, w_1, \cdots, w_n) \\ &= \sigma_{K_0}(\ell - w_n) + \sum_{k=1}^{n-1} \sigma_{K_0}(A^{-T}w_{k+1} - w_k) \qquad (21) \\ &\quad +\sigma_{K_0}(A^{-T}w_1) + \sum_{k=1}^{n} \sigma_{\mathcal{V}}(A^{-T}w_k).\end{aligned}$$

Proof. Follows from (20) by induction on n. □

Using Theorem 5 we can express the set of support vectors of the set K_{n+1} in the direction ℓ as

$$\begin{aligned}v_{K_{n+1}}(\ell) &= v_{A^{-1}(K_n\oplus\mathcal{V})\cap K_0}(\ell) \\ &= v_{K_0}(\ell - \bar{w}) \cap A^{-1}\big(v_{\mathcal{V}}(A^{-T}\bar{w}) \oplus v_{K_n}(A^{-T}\bar{w})\big)\end{aligned}$$

where

$$\bar{w} \in \arg\inf_{w\in\mathbb{R}^d} \left\{\sigma_{K_0}(\ell - w) + \sigma_{\mathcal{V}}(A^{-T}w) + \sigma_{K_n}(A^{-T}w)\right\}.$$

We have the following algorithm for computing under-approximations $\mathcal{P}_\downarrow$ and over-approximations $\mathcal{P}_\uparrow$ of the viability kernel that are tight in the set of directions $\mathcal{L}$. Although it contains states that are not viable, the over-approximation can be used to provide an upper bound on the error in the under-approximation, and the latter is guaranteed to contain only states that are viable under the discrete dynamics.

Algorithm 4 Support vector method

```
for $\ell \in \mathcal{L}$ do
    minimize
        $\xi(\ell, w) = \sigma_{K_0}(\ell - w_n) + \sum_{k=1}^{n-1} \sigma_{K_0}(A^{-T}w_{k+1} - w_k) + \sigma_{K_0}(A^{-T}w_1) + \sum_{k=1}^{n} \sigma_{\mathcal{V}}(A^{-T}w_k).$
    subject to $w \in \mathbb{R}^{nd}$
    $\sigma(\ell) \leftarrow \xi(\ell, \bar{w})$        ▷ Minimum value stored as $\sigma(\ell)$
    $V_0 = v_{K_0}(A^{-T}\bar{w}_n)$
    for $k = 1 \ldots n$ do
        $V_k = v_{K_0}(A^{-T}\bar{w}_{n-k} - \bar{w}_{n-k+1}) \cap A^{-1}\Big(v_{\mathcal{V}}(A^{-T}\bar{w}_{n-k+1}) \oplus V_{k-1}\Big)$
    end for
    $v(\ell) \leftarrow V_n$
end for
$\mathcal{P}_\downarrow \leftarrow \mathrm{Conv}\left(\bigcup_{\ell\in\mathcal{L}} v(\ell)\right)$
$\mathcal{P}_\uparrow \leftarrow \bigcap_{\ell\in\mathcal{L}} \{x \mid x \cdot \ell \le \sigma(\ell)\}$
return $(\mathcal{P}_\downarrow, \mathcal{P}_\uparrow)$        ▷ $\mathcal{P}_\downarrow \subseteq Viab_{\mathbb{T}}(K_0) \subseteq \mathcal{P}_\uparrow$
```

3.3 Comparison of algorithms

We now compare the three Lagrangian algorithms that we have presented. The polytope, ellipsoid and support vector algorithms are implemented in MATLAB using the Multi-Parametric Toolbox v. 2.6.3 (Kvasnica, Grieder, Baotić, and Morari, 2004), Ellipsoidal Toolbox v. 1.1.3 (Kurzhanskiy and Varaiya, 2006) and CVX (Grant and Boyd, 2008) respectively. Computations were performed using MATLAB release 2009b on a machine with an Intel Pentium 4 processor running at 3.00 GHz and 2GB RAM. The MATLAB code to generate the figures in this section and Section 4 can be downloaded from the web at `http://www.ece.ubc.ca/~jmaidens/viability_supplement.zip`

3.3.1 Accuracy

We compare the accuracy of the three algorithms by comparing their performance on a standard example. Consider the discrete-time double integrator

$$\begin{cases} \begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & \rho \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\rho^2 \\ \rho \end{bmatrix} u(t) \\ u(t) \in \mathcal{U} = [-u_0, u_0] \end{cases} \tag{22}$$

with time step $\rho > 0$. Figures 4 – 6 show the results of our three algorithms run on the model (22) with $\rho = 0.1$, a horizon of 40 steps and input constraint $|u(t)| \leq 0.3$.

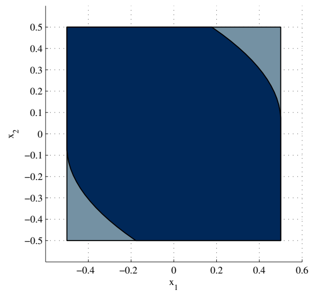
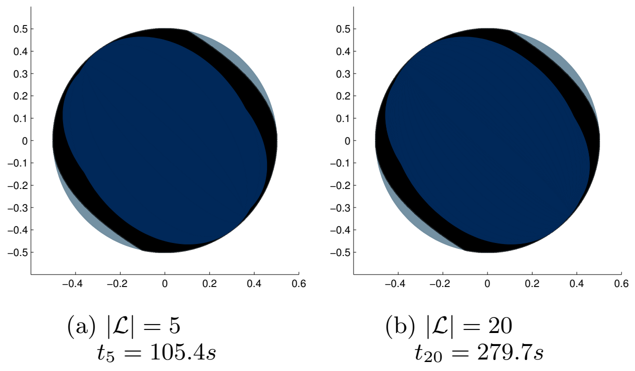


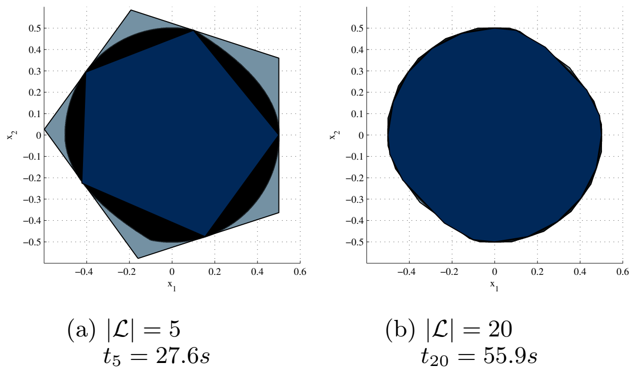
Fig. 4. Viability constraints $K = \{x : ||x||_\infty \leq 0.5\}$ (light grey) and the corresponding viability kernel $Viab_{\mathbb{Z}_+}(K)$ (dark blue) for the double integrator (22) computed using Algorithm 3. The finite horizon viability kernel converges to the infinite horizon viability kernel within 16 steps. The viability kernel contains 34 vertices and took $t_p = 1.65$ seconds to compute.

The polytope algorithm performs best in terms of accuracy, computing the discrete time viability kernel exactly. The ellipsoidal method provides a conservative under-approximation of the viability kernel due to the approximations that must be performed at each intersection step. Adding additional approximation directions improves the approximation to a limited extent. The support vector method provides only a rough approximation when evaluated in a small number of directions



(a) $|\mathcal{L}| = 5$ $t_5 = 105.4s$ (b) $|\mathcal{L}| = 20$ $t_{20} = 279.7s$

Fig. 5. Viability constraints $K = \{x : ||x||_2 \leq 0.5\}$ (light grey) and the corresponding viability kernel $Viab_{\mathbb{R}_+}(K)$ (dark blue) for the double integrator (22) computed using Algorithm 2 from (Kaynama, Maidens, Mitchell, Oishi, and Dumont, 2012). This computation was performed for two different uniformly-spaced sets of directions $\mathcal{L}$ and the corresponding computation times are noted. An Eulerian approximation to the viability kernel based on a 1001 × 1001 grid of points (which is highly accurate, but infeasible in higher dimensions) is shown in black.



(a) $|\mathcal{L}| = 5$ $t_5 = 27.6s$ (b) $|\mathcal{L}| = 20$ $t_{20} = 55.9s$

Fig. 6. Over- (light grey) and under- (dark blue) approximations of the viability kernel of the constraint set $K = \{x : ||x||_2 \leq 0.5\}$ computed using Algorithm 4 for two different uniformly- spaced sets of directions $\mathcal{L}$. An Eulerian approximation to the viability kernel based on a 1001 × 1001 grid of points (which is highly accurate, but infeasible in higher dimensions) is shown in black.

but can be made arbitrarily accurate by choosing a sufficiently large number of evaluation directions $\mathcal{L}$.

3.3.2 Scalability

We compare how well the three algorithms scale as a function of the state dimension by comparing their performance on a discrete-time model of a chain of d integrators

$$x(t+1) = \begin{bmatrix} 9.49(10^{-1}) & 2.75(10^{-2}) & 1.02(10^{-2}) & 5.09(10^{-6}) & -1.44(10^{-5}) & 1.43(10^{-5}) \\ 1.33(10^{-2}) & 9.86(10^{-1}) & 7.04(10^{-5}) & 1.02(10^{-7}) & -1.39(10^{-7}) & 9.42(10^{-8}) \\ 8.03(10^{-4}) & 1.14(10^{-5}) & 9.99(10^{-1}) & 6.21(10^{-9}) & -8.41(10^{-9}) & 5.68(10^{-9}) \\ 0 & 0 & 0 & -1.22(10^{-1}) & -7.93(10^{-2}) & 5.66(10^{-2}) \\ 0 & 0 & 0 & -1.20(10^{-1}) & -3.04(10^{-1}) & -3.85(10^{-1}) \\ 0 & 0 & 0 & 4.11(10^{-1}) & 5.56(10^{-1}) & 4.66(10^{-1}) \end{bmatrix} x(t) + \begin{bmatrix} -1.11(10^{-7}) \\ 4.23(10^{-9}) \\ 2.54(10^{-10}) \\ -6.04(10^{-2}) \\ 4.11(10^{-1}) \\ 5.68(10^{-1}) \end{bmatrix} u(t) \quad (23)$$

$$\begin{cases} x(t+1) = \begin{bmatrix} 1 & \rho & \frac{1}{2}\rho^2 & \cdots & \frac{1}{(d-1)!}\rho^{d-1} \\ 0 & 1 & \rho & & \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & & \rho \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} x(t) + \begin{bmatrix} \frac{1}{d!}\rho^d \\ \\ \vdots \\ \frac{1}{2}\rho \\ \rho \end{bmatrix} u(t) \\ u(t) \in \mathcal{U} = [-u_0, u_0]. \end{cases}$$

Its viability kernel is computed over a horizon of 10 steps, with $\rho = 0.4$, input constraint $|u(t)| \leq 0.3$ and state constraint set $K = \{x : ||x||_\infty \leq 0.5\}$ for the polytope algorithm and $K = \{x : ||x||_2 \leq 0.5\}$ for the ellipsoid and support vector algorithms.

In Figure 7 we plot the time it takes to compute (a) the viability kernel using the polytope method (b) an ellipsoidal under-approximation to the viability kernel in a single direction (c) $2d$ support vectors on the boundary of the viability kernel tight in a set of directions consisting of the standard basis vectors in $\mathbb{R}^d$ and their negatives.

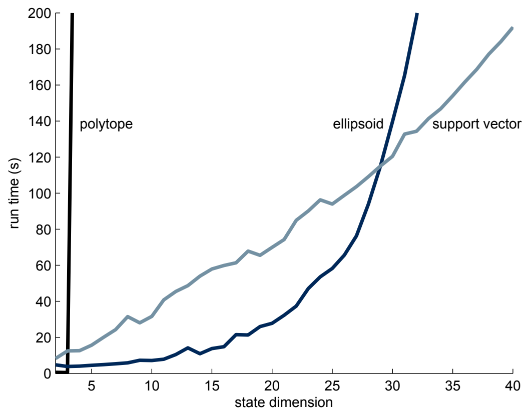


Fig. 7. Comparison of the run time for a chain of integrators of length d.

4 Applications

We conclude by using the support vector algorithm developed in Section 3.2.3 to tackle two problems which were previously infeasible using Eulerian methods. In Section 4.1 we compute the viability kernel for a model of the pharmacokinetics of the anaesthetic drug Propofol and in Section 4.2, we compute the viability kernel for a 20-dimensional temperature control problem.

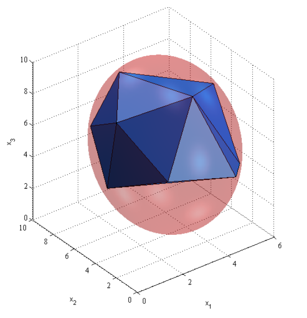
4.1 Closed-loop control of anaesthesia

The Electrical & Computer Engineering in Medicine group at the University of British Columbia recently completed a set of clinical tests for a paediatric closed-loop anaesthesia system at the British Columbia Children's Hospital (Soltesz, van Heusden, Dumont, Hägglund, Petersen, West, and Ansermino, 2012). To ensure safe operation, we would like to place hard bounds on the Propofol administration rate (the input) and compartmental Propofol concentrations (the states).

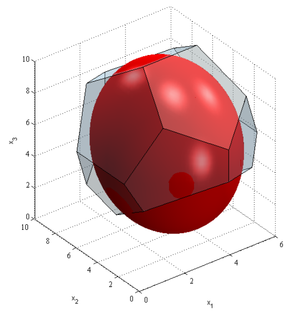
In this context, we compute the viability kernel for the purposes of "fallback mode" initiation (ISO/IEC, 2007). Due to modeling inaccuracies or unmodeled disturbances in the surgical theatre, it is possible that the system's state might leave the "safe" viable region. Thus if at any point in time the control system determines that the system's state is outside the viability kernel, an alarm should be sounded and a fallback mode initiated.

Consider the three-compartment pharmacokinetic system

$$\begin{cases} \begin{bmatrix} \dot{c}_1(t) \\ \dot{c}_2(t) \\ \dot{c}_3(t) \end{bmatrix} = \begin{bmatrix} -(k_{10}+k_{12}+k_{13}) & k_{12} & k_{13} \\ k_{21} & -k_{21} & 0 \\ k_{31} & 0 & -k_{31} \end{bmatrix} \begin{bmatrix} c_1(t) \\ c_2(t) \\ c_3(t) \end{bmatrix} \\ \qquad + \begin{bmatrix} 1/V_1 \\ 0 \\ 0 \end{bmatrix} u(t-t_d) \\ u(t) \in \mathcal{U} = [0, u_0] \end{cases}$$



(a) Viability constraints (red ellipsoid) and under-approximation of the viability kernel (blue polytope)



(b) Viability constraints (red ellipsoid) and over-approximation of the viability kernel (grey polytope)

Fig. 8. Viability kernel for the 6-dimensional discrete-time pharmacokinetic model given in (23) computed using Algorithm 4. Constraints and viability kernels are shown projected onto the first 3 coordinates of the 6-dimensional state space. These approximations were computed in $693s$.

with input delay $t_d = 0.5$ minutes and the model parameters for an 11 year-old child of 35 kg taken from the Paedfusor data set (Absalom and Kenny, 2005). The delay is approximated using a third order Padé approximation and then the system is discretized with a time step $\rho = 0.25$ minutes, yielding the 6-dimensional discrete-time model given in (23). We simulate a 90 minute surgery (time horizon $\tau = 90$ minutes) with input constraint $u(t) \in [0, 7\,000]$ μg/min (200μg/kg/min) to compute the viability kernel of the state constraints given by a tight ellipsoid inscribed in the box $[1, 6] \times [0, 10] \times [0, 10] \times [-100, 100] \times [-100, 100] \times [-100, 100]$. The computed approximation is sampled in the set of directions

$$\mathcal{L} = \left\{ \mathcal{Q}^{-1} \begin{bmatrix} \ell \\ 0 \\ 0 \\ 0 \end{bmatrix} : \ell \in V_I \right\} \cup \left\{ \mathcal{Q}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \pm e_i \end{bmatrix} : i = 1, 2, 3 \right\}$$

where V_I is the set of the 12 vertices of an icosahedron in $\mathbb{R}^3$, e_i are the standard basis vectors in $\mathbb{R}^3$ and $\mathcal{Q}$ is the shape matrix for the ellipsoid of state constraints. The computed over- and under-approximations of the viability kernel for (23) are shown in Figure 8.

We see that initial states with a low concentration in the slowly-equilibrating compartment c_3 are not viable under the delayed dynamics. This result emphasizes the importance of providing a bolus dose, or high-rate infusion, during the induction of anaesthesia to allow a sufficient concentration of Propofol to accumulate in the slowly-equilibrating tissues before limiting the infusion rate to $\leq 200\mu$g/kg/min.

4.2 Forced heat equation

The forced heat equation

$$\frac{\partial \xi}{\partial t}(x, t) = \alpha \frac{\partial^2 \xi}{\partial x^2}(x, t) - \beta(\xi - \xi_0) + u(x, t)$$

describes how the temperature distribution over a finite, one-dimensional rod evolves over time when a heat input u is provided to the system. We assume that heat is lost from the rod at a rate (with proportionality constant β) dependent on the difference between the rod's temperature and the ambient temperature ξ_0 (assumed to be zero), that the system's thermal diffusivity is α and that the rod is heated from one end. To study solutions to this equation numerically, we consider the temperature ξ_i at d discrete points x_i on a uniform one-dimensional lattice of spacing Δx. After finite difference approximation of the spatial derivative, followed by a discretization with time step Δt, we get the dynamics

$$\xi(t + 1) = \exp\left(-\Delta t \left(\frac{\alpha}{2(\Delta x)^2} L + D\right)\right) \xi(t) + B_d u(t)$$

where ξ is now the vector of temperatures at the lattice points, L is the lattice's Laplacian matrix, D is the diagonal matrix with entries β and

$$B_d = \int_0^{\Delta t} \exp\left(-t \left(\frac{\alpha}{2(\Delta x)^2} L + D\right)\right) dt \cdot e_1.$$

We compute the viability kernel for this system over a time horizon $\tau = 5$ for the state constraint set consisting of a sphere of radius 10 centred at $[10, \ldots, 10]^T$ using a lattice of 20 points, yielding a 20-dimensional system. We set $\alpha = 8$, $\beta = 0.015$, $\Delta x = 1$, $\Delta t = 0.25$ and constrain the input u to the set $[0, 5]$. We compute an approximation of the viability kernel in a set of 120 directions consisting of 12 uniformly-spaced unit vectors in the $\xi_{2k-1} \times \xi_{2k}$ plane for $k = 1, \ldots, 10$. Figure 9 shows the resulting approximation of the viability kernel projected onto a selection of coordinate planes.

Note that under- and over-approximations appear tight in the projections onto the $\xi_1 \times \xi_2$ and $\xi_{19} \times \xi_{20}$ planes since we sampled more support vectors in these subspaces. The projections onto the $\xi_1 \times \xi_{10}$ and $\xi_1 \times \xi_{20}$ subspaces could be improved simply by sampling in more directions.

5 Conclusions

We presented a connection between viability and reachability that enables us to compute viability kernels in terms of backward reachable sets. While this theoretical connection applies to systems with nonlinear dynamics and general state constraints, we take advantage of it

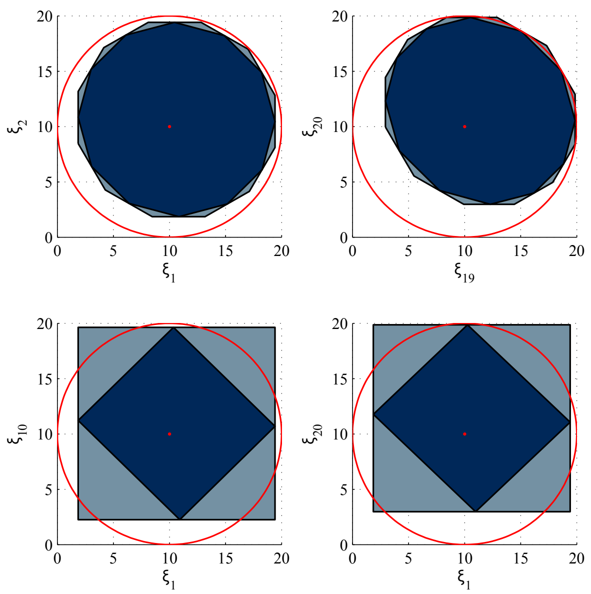


Fig. 9. Viability kernel for the discretized heat conduction problem projected onto various coordinate planes. The projection of the state constraint set is outlined in red and under- and over-approximations of the viability kernel are shown in dark blue and light grey respectively. This approximation in 120 directions was performed in 425 seconds. As expected, we see that the set of viable states appears smaller when projected onto coordinates farther from the heated end at ξ_1.

here to develop scalable Lagrangian algorithms for systems with linear dynamics and convex, compact input and state constraints.

The algorithm based on support vectors performed exceedingly well in the sense that it is scalable as the number of time steps or the state dimension increases and it can be made arbitrarily accurate. The algorithm based on ellipsoids is somewhat less scalable and accurate, but we are currently extending it to approximate discriminating kernels / robust viability kernels in a differential game setting, and to synthesize permissive safety preserving control laws.

The continuous time version of the reachability to viability connection is detailed in (Kaynama, Maidens, Mitchell, Oishi, and Dumont, 2012) and applied there to the ellipsoidal representation. We are currently investigating whether the support vector representation could also be adapted to continuous time.

Scalable reachability for systems with nonlinear dynamics is still an open problem, but should any such algorithm be developed then the techniques presented here will permit its use for viability kernel approximation provided only that the set representation supports a reasonably efficient and accurate intersection operation.

Acknowledgements

Research supported by NSERC Discovery Grants #327387 (Oishi) and #298211 (Mitchell), NSERC Collaborative Health Research Project #CHRPJ-350866-08 (Dumont), an NSERC Canada Graduate Scholarship, and the Institute for Computing, Information and Cognitive Systems (ICICS) at UBC.

References

Absalom, A., Kenny, G., 2005. Paedfusor pharmacokinetic data set. British Journal of Anaesthesia 95, 110.

Aubin, J.-P., 1991. Viability Theory. Systems and Control: Foundations and Applications. Birkhäuser.

Aubin, J.-P., Bayen, A. M., Saint-Pierre, P., 2011. Viability Theory: New Directions. Springer.

Avis, D., Fukuda, K., 1992. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. Discrete & Computational Geometry 8, 295–313.

Blanchini, F., Miani, S., 2008. Set-Theoretic Methods in Control. Springer.

Bonneuil, N., 2006. Computing the viability kernel in large state dimension. Journal of Mathematical Analysis and Applications 323, 1444 – 1454.

Chapel, L., Deffuant, G., Martin, S., Mullon, C., 2008. Defining yield policies in a viability approach. Ecological Modelling 212, 10–15.

Chutinan, A., Krogh, B., 2003. Computational techniques for hybrid system verification. IEEE Transactions on Automatic Control 48, 64 – 75.

Coquelin, P., Martin, S., Munos, R., 2007. A dynamic programming approach to viability problems. In: IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning. pp. 178 –184.

Deffuant, G., Chapel, L., Martin, S., 2007. Approximating viability kernels with support vector machines. IEEE Transactions on Automatic Control 52 (5), 933–937.

Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O., 2011. SpaceEx: Scalable verification of hybrid systems. In: Proceedings of the Conference on Computer Aided Verification. pp. 379–395.

Girard, A., Le Guernic, C., Maler, O., 2006. Efficient computation of reachable sets of linear time-invariant systems with inputs. In: Hybrid Systems: Computation and Control. pp. 257–271.

Grant, M., Boyd, S., 2008. Graph implementations for nonsmooth convex programs. In: Recent Advances in Learning and Control. Springer-Verlag, pp. 95–110.

ISO/IEC, 2007. IEC 60601-1-10:2007 Requirements for the development of physiologic closed-loop controllers. Tech. rep., International Organization for Standardization, Geneva, Switzerland.

Kanade, A., Alur, R., Ivancic, F., Ramesh, S., Sankaranarayanan, S., Shashidhar, K. C., 2009. Generating and analyzing symbolic traces of simulink/stateflow models. In: Proc. Computer Aided Verification. pp. 430–445.

Kaynama, S., Maidens, J., Mitchell, I. M., Oishi, M., Dumont, G. A., 2012. Computing the viability kernel using maximal reachable sets. In: Hybrid Systems: Computation and Control. pp. 55–64.

Kurzhanski, A. B., Varaiya, P., 2000a. Ellipsoidal techniques for reachability analysis. In: Hybrid Systems: Computation and Control. pp. 202–214.

Kurzhanski, A. B., Varaiya, P., 2000b. Ellipsoidal techniques for reachability analysis: internal approximation. Systems & Control Letters 41, 201–211.

Kurzhanski, A. B., Varaiya, P., 2001. Dynamic optimization for reachability problems. Journal of Optimization Theory and Applications 108, 227–251.

Kurzhanskiy, A. A., Varaiya, P., 2006. Ellipsoidal toolbox (ET). In: Proceedings of IEEE Conference on Decision and Control. pp. 1498–1503.

Kvasnica, M., Grieder, P., Baotić, M., Morari, M., 2004. Multi-Parametric Toolbox (MPT). In: Hybrid Systems: Computation and Control. pp. 448–462.

Le Guernic, C., 2009. Reachability analysis of hybrid systems with linear continuous dynamics. Ph.D. thesis, Université Grenoble 1 - Joseph Fourier.

Le Guernic, C., Girard, A., 2010. Reachability analysis of linear systems using support functions. Nonlinear Analysis: Hybrid Systems 4 (2), 250 – 262.

Lygeros, J., 2004. On reachability and minimum cost optimal control. Automatica 40 (6), 917–927.

Mitchell, I. M., 2007. Comparing forward and backward reachability as tools for safety analysis. In: Hybrid Systems: Computation and Control. pp. 428–443.

Mitchell, I. M., Bayen, A. M., Tomlin, C. J., 2005. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. IEEE Transactions on Automatic Control 50 (7), 947–957.

Mitchell, I. M., Templeton, J. A., 2005. A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems. In: Hybrid Systems: Computation and Control. pp. 480–494.

Panagou, D., Margellos, K., Summers, S., Lygeros, J., Kyriakopoulos, K. J., 2009. A viability approach for the stabilization of an underactuated underwater vehicle in the presence of current disturbances. In: Proceedings of IEEE Conference on Decision and Control. pp. 8612–8617.

Rifkin, R. M., Lippert, R. A., 2007. Value regularization and fenchel duality. Journal of Machine Learning Research 8, 441–479.

Rockafellar, R., Wets, R., 1998. Variational analysis. Grundlehren der mathematischen Wissenschaften. Springer.

Saint-Pierre, P., 1994. Approximation of the viability kernel. Applied Mathematics and Optimization 29 (2), 187–209.

Soltesz, K., van Heusden, K., Dumont, G., Hägglund, T., Petersen, C., West, N., Ansermino, J., 2012. Closed-loop anesthesia in children using a PID controller: A pilot study. In: Proc. IFAC Conf. on Advances in PID Control.

Tomlin, C. J., Mitchell, I. M., Bayen, A. M., Oishi, M., 2003. Computational techniques for the verification and control of hybrid systems. Proceedings of the IEEE 91 (7), 986–1001.

A Appendix: Proof of Theorem 5

As the proof of Theorem 5 requires the introduction of some elementary convex analysis, we have placed it here in the appendix.

A.1 Convex conjugation and duality

For a proper convex function $f : \mathbb{R}^d \to \bar{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$, its conjugate function $f^* : \mathbb{R}^d \to \bar{\mathbb{R}}$ is a proper convex function defined as

$$f^*(\ell) = \sup_{x \in \mathbb{R}^d} \{\langle x, \ell \rangle - f(x)\}.$$

If f is further assumed to be lower semi continuous (lsc) then we have

$$f^{**} = f$$

and hence there is a duality, or conjugacy correspondence, between lsc proper convex functions.

Define the characteristic function $\delta_S : \mathbb{R}^d \to \bar{\mathbb{R}}$ of a nonempty convex set $S \subset \mathbb{R}^d$

$$\delta_S(x) = \begin{cases} 0 & \text{if } x \in S \\ \infty & \text{if } x \notin S \end{cases}$$

The convex conjugate of the characteristic function δ_S is the support function σ_S

$$\sigma_S(\ell) = \sup_{x \in S} \langle x, \ell \rangle = \sup_{x \in \mathbb{R}^d} \{\langle x, \ell \rangle - \delta_S(x)\} = \delta_S^*(\ell).$$

The conjugation operation also induces a duality between operations on functions. Define the infimal convolution (or epi-sum) $f_1 \# f_2$ of two convex functions f_1 and f_2 as

$$f_1 \# f_2 \ (\ell) = \inf_{w \in \mathbb{R}^d} f_1(\ell - w) + f_2(w).$$

The following proposition establishes that infimal convolution is the dual of the summation operation.

Lemma 1 (Theorem 11.23a, Rockafellar and Wets (1998)) *If f_1 and f_2 are two lsc proper convex functions whose domains have nonempty intersection then*

- $(f_1 + f_2)^* = f_1^* \# f_2^*$
- $(f_1 \# f_2)^* = f_1^* + f_2^*$.

A.2 Subgradients and support vectors

Let $f : \mathbb{R}^d \to \bar{\mathbb{R}} = (-\infty, \infty]$ be convex. The set $\partial f(\ell)$ of subgradients of f at ℓ is the set of vectors $x \in \mathbb{R}^d$ such that for all $\tilde{\ell} \in \mathbb{R}^d$

$$f(\ell) - f(\tilde{\ell}) \geq x \cdot (\ell - \tilde{\ell})$$

The following result is standard:

Lemma 2 (Fenchel's Inequality, Proposition 11.3, Rockafellar and Wets (1998)) *Let $f : \mathbb{R}^d \to \bar{\mathbb{R}}$ be a proper convex function. Then for all $\ell, x \in \mathbb{R}^d$ we have*

$$f(\ell) - \ell \cdot x + f^*(x) \geq 0.$$

Further, equality holds if and only if $x \in \partial f(\ell)$.

From Fenchel's Inequality, we get the following characterization of the set of support vectors of a compact, convex set $\mathcal{U}$ in the direction ℓ:

Theorem 7 *Let $\sigma_{\mathcal{U}}(\ell)$ and $v_{\mathcal{U}}(\ell)$ be defined as in (16) and (18). Then*

$$v_{\mathcal{U}}(\ell) = \partial \sigma_{\mathcal{U}}(\ell)$$

Proof.

$$\begin{aligned} x \in v_{\mathcal{U}}(\ell) &\iff x \in \mathcal{U} \text{ and } x \cdot \ell = \sigma_{\mathcal{U}}(\ell) \\ &\iff \sigma_{\mathcal{U}}(\ell) - x \cdot \ell + \delta_{\mathcal{U}}(x) = 0 \\ &\iff x \in \partial \sigma_{\mathcal{U}}(\ell). \qquad \square \end{aligned}$$

A.3 Theorem 5

Before proving Theorem 5 we need one final result:

Lemma 3 (Proposition 2.22a and Theorem 10.13, Rockafellar and Wets (1998)) *Let $g : \mathbb{R}^d \times \mathbb{R}^n \to \bar{\mathbb{R}}$ be a proper convex function. Then $\tilde{g}(\ell) = \inf_{w \in \mathbb{R}^n} g(\ell, w)$ is a proper convex function and*

$$(x, 0) \in \partial g(\ell, \bar{w}) \iff x \in \partial \tilde{g}(\ell) \text{ and } \tilde{g}(\ell) = g(\ell, \bar{w}).$$

Proof of Theorem 5.

The first two bullet points in Theorem 5 follow directly from the definition. We prove only the third. The proof is adapted from (Rifkin and Lippert, 2007).

Define $g(\ell, w) = \sigma_{\mathcal{U}}(\ell - w) + \sigma_{\mathcal{V}}(w)$ so that $\tilde{g}(\ell) = \inf_{w \in \mathbb{R}^n} g(\ell, w) = (\sigma_{\mathcal{U}} \# \sigma_{\mathcal{V}})(\ell)$. The assumption $\bar{w} \in \arg\inf_{w \in \mathbb{R}^d} \{\sigma_{\mathcal{U}}(\ell - w) + \sigma_{\mathcal{V}}(w)\}$ means that $\tilde{g}(\ell) = g(\ell, \bar{w})$. Hence it follows from Lemmas 2 and 3 that

$$\begin{aligned} & x \in v_{\mathcal{U} \cap \mathcal{V}}(\ell) = \partial \sigma_{\mathcal{U} \cap \mathcal{V}}(\ell) = \partial (\sigma_{\mathcal{U}} \# \sigma_{\mathcal{V}})(\ell) = \partial \tilde{g}(\ell) \\ \iff & (x, 0) \in \partial g(\ell, \bar{w}) \\ \iff & \\ & 0 = g(\ell, \bar{w}) - x \cdot \ell - 0 \cdot \bar{w} + g^*(x, 0) \\ & \quad = \sigma_{\mathcal{U}}(\ell - \bar{w}) + \sigma_{\mathcal{V}}(\bar{w}) - x \cdot \ell + \sigma_{\mathcal{U}}^*(x) + \sigma_{\mathcal{V}}^*(x + 0) \\ & \quad = [\sigma_{\mathcal{U}}(\ell - \bar{w}) - x \cdot (\ell - \bar{w}) + \sigma_{\mathcal{U}}^*(x)] \\ & \qquad + [\sigma_{\mathcal{V}}(\bar{w}) - x \cdot \bar{w} + \sigma_{\mathcal{V}}^*(x)] \\ \iff & x \in \partial \sigma_{\mathcal{U}}(\ell - \bar{w}) \text{ and } x \in \partial \sigma_{\mathcal{V}}(\bar{w}) \\ \iff & x \in v_{\mathcal{U}}(\ell - \bar{w}) \cap v_{\mathcal{V}}(\bar{w}). \qquad \square \end{aligned}$$