A Summary of Recent Progress on Efficient Parametric Approximations of Viability and Discriminating Kernels

Ian M. Mitchell

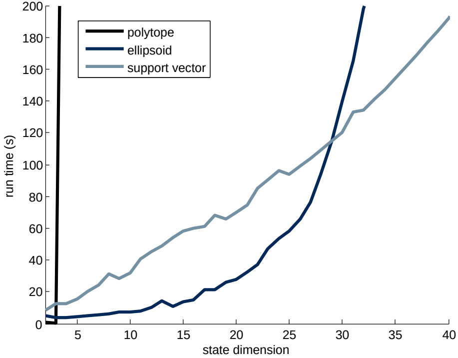
Department of Computer Science
The University of British Columbia

July 2015

mitchell@cs.ubc.ca
http://www.cs.ubc.ca/~mitchell

Let's Cut to the Chase

We can approximate the set of controllably safe states within some constraint set $\mathcal{K}$ in polynomial time for linear systems using parametric approximations.



It may be worth trading off algorithm speed and accuracy (support vector approach) for other capabilities (ellipsoidal approach).

Outline

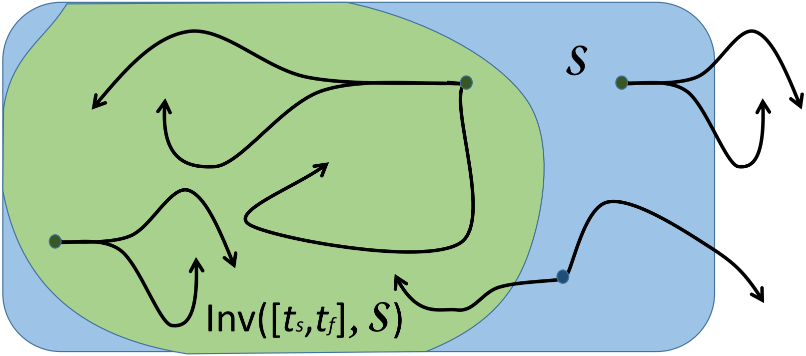
1. Constructs & Motivation
2. Models & Algorithms
3. Implementations & Results
4. Comparison & Discussion

Outline

1. Constructs & Motivation
2. Models & Algorithms
3. Implementations & Results
4. Comparison & Discussion

Invariance Kernel

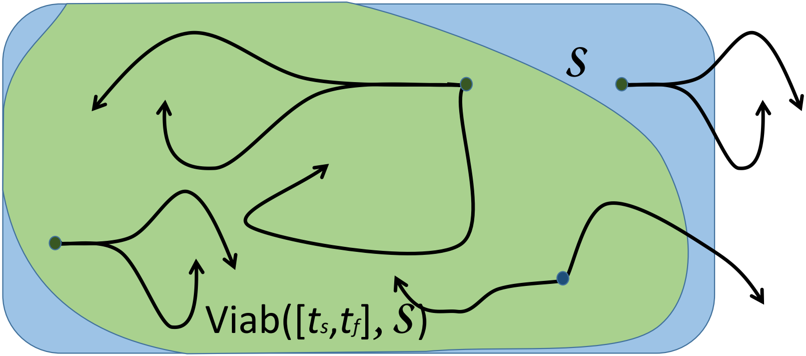
$$\text{Inv}\left([t_s, t_f], \mathcal{S}\right) \triangleq \{\tilde{x}(t_s) \in \mathcal{S} \mid \forall u(\cdot), \forall t \in [t_s, t_f], x(t) \in \mathcal{S}\},$$



- What states will remain safe despite input uncertainty.
- Inputs treated in a worst-case fashion.
- We will not further discuss this kernel.

Viability Kernel

$$\text{Inv}\left([t_s, t_f], \mathcal{S}\right) \triangleq \{\tilde{x}(t_s) \in \mathcal{S} \mid \exists u(\cdot), \forall t \in [t_s, t_f], x(t) \in \mathcal{S}\},$$



- Also called controlled invariant set.
- Inputs treated in a best-case fashion.

Discriminating Kernel

$$\mathsf{Inv}\left([t_s, t_f], \mathcal{S}\right) \triangleq \{\tilde{x}(t_s) \in \mathcal{S} \mid \exists u(\cdot), \forall v(\cdot), \forall t \in [t_s, t_f], x(t) \in \mathcal{S}\},$$

That is hard to draw...

- Also called robust controlled invariant set.
- Two inputs "control" $u(\cdot)$ and "disturbance" $v(\cdot)$ treated adversarially.

The Challenge: Efficient Parametric Representations

Existing algorithms used non-parametric representations; complexity is exponential in state space dimension.

- Viability algorithms: for example [Saint-Pierre 1994; Cardaliaguet et al 1999].
- Level set methods: for example [Mitchell et al 2005].

In contrast, algorithms using parametric representations for reachable sets are widely available.

$$\mathrm{Reach}_{+}(t, \mathcal{S}) \triangleq \{x_0 \mid \exists u(\cdot), x(t) \in \mathcal{S}\},$$
$$\mathrm{Reach}_{-}(t, \mathcal{S}) \triangleq \{x_0 \mid \forall u(\cdot), x(t) \in \mathcal{S}\},$$

- Ellipsoids: for example [Kurzhanski & Valyi 1996; Kurzhanski & Varaiya 2000; Kurzhanskiy & Varaiya 2006].
- Support functions / vectors: for example [Le Guernic 2009; Le Guernic & Girard 2010; Frehse et al 2011].

Outline

1. Constructs & Motivation

2. Models & Algorithms

3. Implementations & Results

4. Comparison & Discussion

Discrete and Continuous Time

Discrete time:

$$x(t+1) = f(x(t), u(t), v(t)) \qquad \text{general dynamics}$$
$$x(t+1) = \mathrm{A}x(t) + \mathrm{B}u(t) + \mathrm{C}v(t) \qquad \text{linear dynamics}$$

- Assume state feedback: Choose $u(t)$ knowing $x(t)$.
- Conservative treatment of uncertainty: Choose $v(t)$ knowing $x(t)$ and $u(t)$.

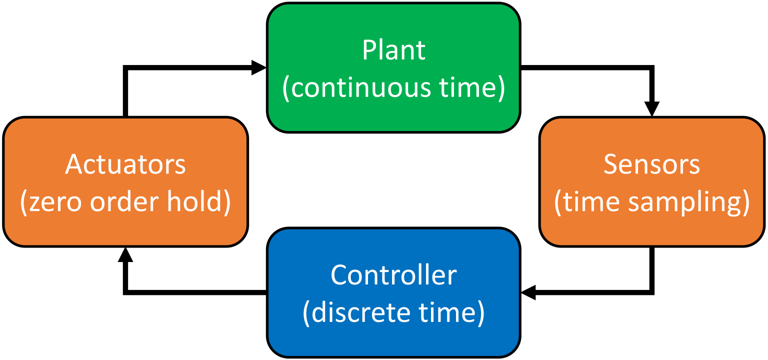
Continuous time:

$$\dot{x}(t) = f(x(t), u(t), v(t)) \qquad \text{general dynamics}$$
$$\dot{x}(t) = \mathrm{A}x(t) + \mathrm{B}u(t) + \mathrm{C}v(t) \qquad \text{linear dynamics}$$

- "Non-anticipative strategies" rigorously resolve input ordering issue; equivalent to state feedback in all but artificially constructed examples.
- Optimal input signals often have little regularity and hence may not be physically realizable.

Sampled Data Model of Time

Sampled data is a model of a common approach to designing cyber-physical systems:



- Unlike continuous time models, change to feedback control is only possible at sample times.
- Unlike discrete time models, state of plant between sample times is relevant.

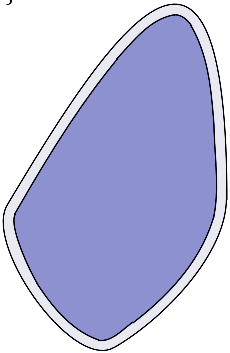
Continuous-Time Viability Algorithm

- Start with an under-approximation $\mathcal{K}_\downarrow$ of $\mathcal{K}$ (ρ: small computational timestep; M: uniform bound on f)

$$\mathcal{K}_\downarrow := \{x \in \mathcal{K} \mid \mathrm{dist}(x, \mathcal{K}^c) \geq \rho M\}$$

- Iteratively compute K_{n+1}:

$$\begin{aligned} \mathcal{K}_0 &= \mathcal{K}_\downarrow, \\ \mathcal{K}_{n+1}(P) &= \mathcal{K}_0 \cap \mathsf{Reach}_+(\rho, \mathcal{K}_n) \end{aligned}$$



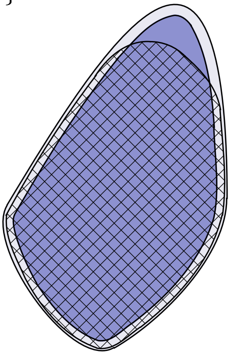
Continuous-Time Viability Algorithm

- Start with an under-approximation $\mathcal{K}_\downarrow$ of $\mathcal{K}$ (ρ: small computational timestep; M: uniform bound on f)

$$\mathcal{K}_\downarrow := \{x \in \mathcal{K} \mid \mathrm{dist}(x, \mathcal{K}^c) \geq \rho M\}$$

- Iteratively compute K_{n+1}:

$$\begin{aligned} \mathcal{K}_0 &= \mathcal{K}_\downarrow, \\ \mathcal{K}_{n+1}(P) &= \mathcal{K}_0 \cap \mathsf{Reach}_+(\rho, \mathcal{K}_n) \end{aligned}$$



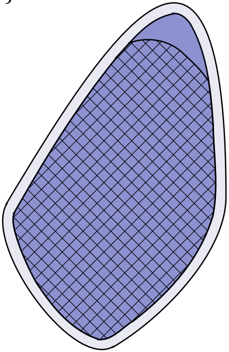
Continuous-Time Viability Algorithm

- Start with an under-approximation $\mathcal{K}_\downarrow$ of $\mathcal{K}$ (ρ: small computational timestep; M: uniform bound on f)

$$\mathcal{K}_\downarrow := \{x \in \mathcal{K} \mid \mathrm{dist}(x, \mathcal{K}^c) \geq \rho M\}$$

- Iteratively compute K_{n+1}:

$$\begin{aligned} \mathcal{K}_0 &= \mathcal{K}_\downarrow, \\ \mathcal{K}_{n+1}(P) &= \mathcal{K}_0 \cap \mathsf{Reach}_+ \left(\rho, \mathcal{K}_n\right) \end{aligned}$$



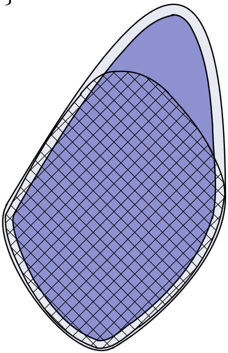
Continuous-Time Viability Algorithm

- Start with an under-approximation $\mathcal{K}_\downarrow$ of $\mathcal{K}$
  (ρ: small computational timestep; M: uniform bound on f)

$$\mathcal{K}_\downarrow := \{x \in \mathcal{K} \mid \mathrm{dist}(x, \mathcal{K}^c) \geq \rho M\}$$

- Iteratively compute K_{n+1}:

$$\begin{aligned} \mathcal{K}_0 &= \mathcal{K}_\downarrow, \\ \mathcal{K}_{n+1}(P) &= \mathcal{K}_0 \cap \mathsf{Reach}_+(\rho, \mathcal{K}_n) \end{aligned}$$



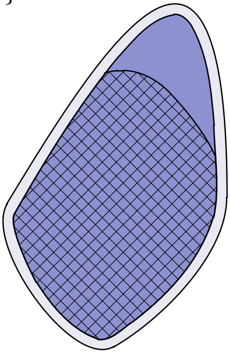
Continuous-Time Viability Algorithm

- Start with an under-approximation $\mathcal{K}_\downarrow$ of $\mathcal{K}$ (ρ: small computational timestep; M: uniform bound on f)

$$\mathcal{K}_\downarrow := \{x \in \mathcal{K} \mid \mathrm{dist}(x, \mathcal{K}^c) \geq \rho M\}$$

- Iteratively compute K_{n+1}:

$$\begin{aligned}\mathcal{K}_0 &= \mathcal{K}_\downarrow, \\ \mathcal{K}_{n+1}(P) &= \mathcal{K}_0 \cap \mathsf{Reach}_+(\rho, \mathcal{K}_n)\end{aligned}$$



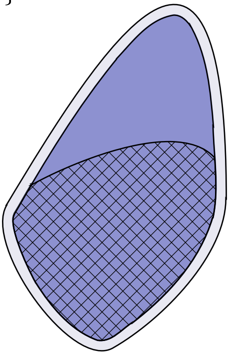
Continuous-Time Viability Algorithm

- Start with an under-approximation $\mathcal{K}_\downarrow$ of $\mathcal{K}$ (ρ: small computational timestep; M: uniform bound on f)

$$\mathcal{K}_\downarrow := \{x \in \mathcal{K} \mid \mathrm{dist}(x, \mathcal{K}^c) \geq \rho M\}$$

- Iteratively compute K_{n+1}:

$$\mathcal{K}_0 = \mathcal{K}_\downarrow,$$
$$\mathcal{K}_{n+1}(P) = \mathcal{K}_0 \cap \mathsf{Reach}_+(\rho, \mathcal{K}_n)$$



Other Constructs and Models

- Discriminating kernel algorithm is straightforward, albeit notationally complicated.
- Discrete time algorithm omits initial erosion: $\mathcal{K}_0 = \mathcal{K}$.
- Sampled data algorithm uses continuous time algorithm in an augmented state space

$$\tilde{x} \triangleq \begin{bmatrix} x \\ u \end{bmatrix} \qquad \tilde{f}(\tilde{x}) \triangleq \begin{bmatrix} f(x,u) \\ 0 \end{bmatrix}.$$

  - Control input held constant over each sample period.
  - Disturbance input allowed to vary (measurably).
  - Tensor products and projections move between original and augmented state space.

Outline

1. Constructs & Motivation

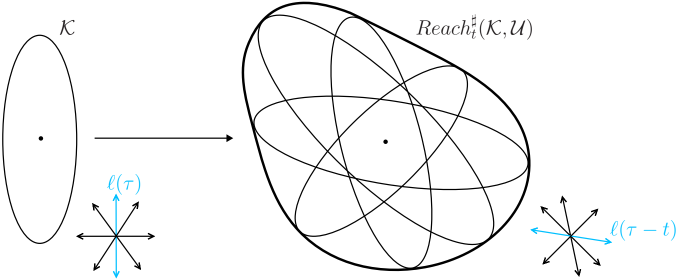
2. Models & Algorithms

3. Implementations & Results

4. Comparison & Discussion

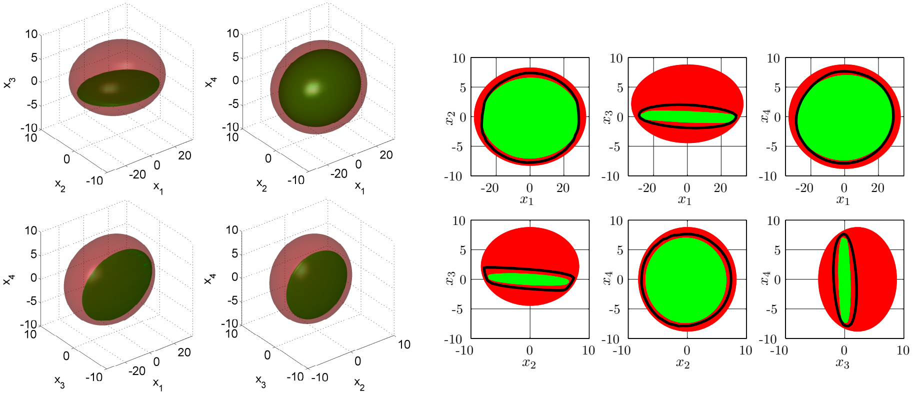
Ellipsoids

Ellipsoidal techniques (under-)approximating the maximal reach set:



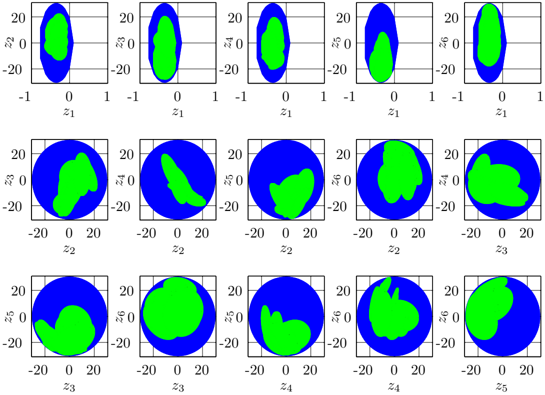
- Key operations (set evolution, intersection) are accomplished through ODEs and convex optimization.
- Class of ellipsoids are not closed under these operations, so underapproximations must be used.
- Set evolution possible in discrete or continuous time.
- Control and/or disturbance inputs can be treated.

Applications: Flight Envelope Protection (CT, 4D)



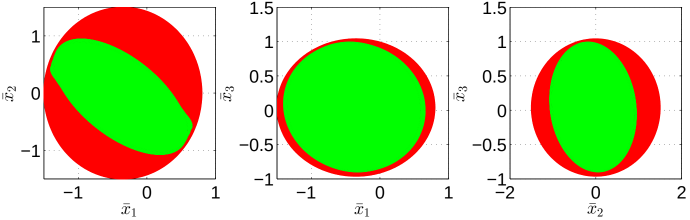
Level-Set (non-parametric, black): 5.5 hr
Piecewise Ellipsoidal (parametric, green): 10 min

Applications: Automated Anesthesia (DT Laguerre model, 7D)

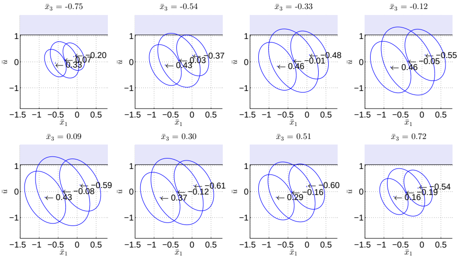


Level Set (non-parametric): infeasible
Piecewise Ellipsoidal (parametric): 15 min

Applications: Quadrotor Altitude Maintenance (nonlinear SD, 3D)

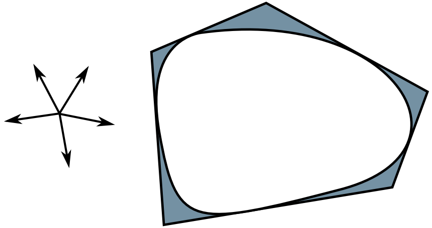


- Linearize within constraint set, use discriminating kernel to ensure robustness to linearization error.
- One second horizon with 10 Hz sample cycle.
- 20 directions, execution time 5 min.
- Also generate safe range of inputs (slices shown at right).

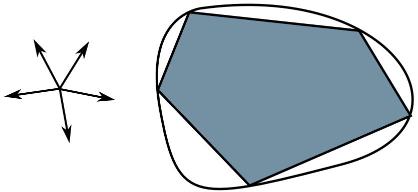


Support Vectors

Support functions provide polytopic overapproximation in specified directions



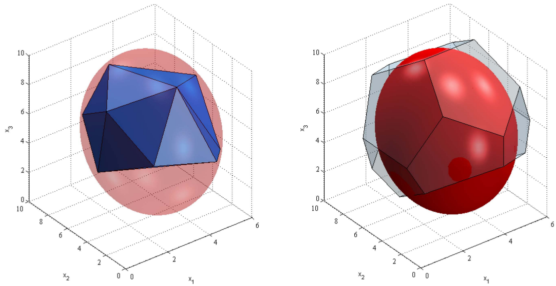
Corresponding support vectors provide polytopic underapproximation in specified directions



- Key operations (set evolution, intersection) are accomplished through convex optimization.
- Support functions / vectors are closed under these operations, so no need to further underapproximate.
- Only discrete time.
- Only control input (no discriminating kernel version).

Application: Automated Anesthesia (DT compartment model, 6D)

- Three compartment LTI model of Propofol metabolism.
- Third order Padé approximation of input delay yields six dimensional state space.
- 18 directions, execution time 11.5 min.
- Support vector underapproximation (left) and free support function overapproximation (right).

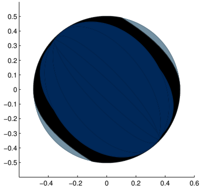


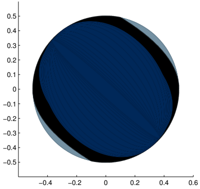
Outline

1. Constructs & Motivation
2. Models & Algorithms
3. Implementations & Results
4. Comparison & Discussion

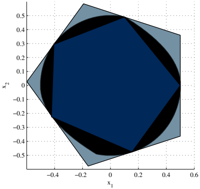
Comparing Accuracy: A Double Integrator

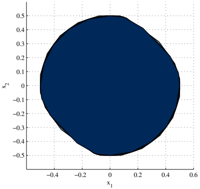
All images: True viability kernel (black).





5 directions, execution time 105s. 20 directions, execution time 280s.
Ellipsoidal approximation (dark blue) and constraint (light grey).



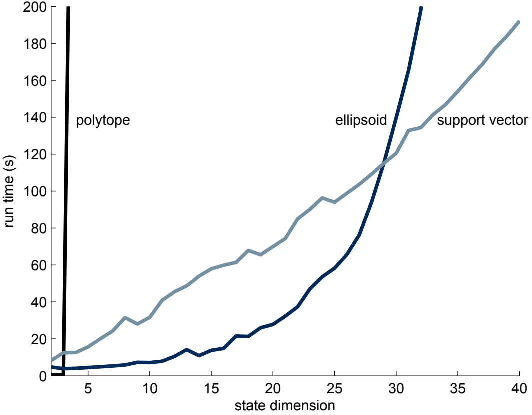


5 directions, execution time 28s. 20 directions, execution time 56s.
Support vector approximation (dark blue) and support function (light grey).

Scaling with Dimension: A Chain of Integrators

Compare execution time over ten steps for a discrete time model.

- Exact polytopic method (non-parametric).
- Ellipsoidal algorithm in a single direction.
- Support vector algorithm in $2d_x$ standard basis vectors (positive and negative directions).



Comparing the Options

	Level Set	Ellipsoidal	Support Vector
Dynamics	nonlinear	linear	linear
Time	CT / SD	CT / DT / SD	DT
Complexity	$\mathcal{O}(n^d)$	$\mathcal{O}(kd^3)$	$\mathcal{O}(kd^2)$
Control input	optimal / sampled	optimal	optimal
Control synthesis	✓	✓	–
Discriminating kernel	optimal	optimal	–
Accuracy	excellent	fair	good
Inner guarantee	–	✓	✓
Outer approx	–	?	free

- Time models are continuous (CT), discrete (DT) or sampled data (SD).
- Complexity parameters are dimension ($d = d_x$ for CT or DT, $d = d_x + d_u$ for SD), grid resolution per dimension (n) and number of ellipses / support vectors (k).

And Yet You Insist on Using Ellipsoids...

Support vector approach is faster and more accurate, so why we are working more actively on the ellipsoidal approach?

- All models are wrong, but discriminating kernels can generate approximations robust to model error.
- Discrete time approximation is too simplistic for continuous time systems with fast dynamics.
- Viability analysis without control synthesis only accomplishes half the job.

It is possible that the support vector approach could be extended to handle disturbance inputs, continuous time and/or control synthesis.

Future Work

Control filtering to ensure safety of human-in-the-loop quadrotor control.

- Longitudinal 6D quadrotor model.
- Sampled data with 10 Hz sample cycle.
- Control inputs are total thrust and differential thrust.
- Linearization about hover condition with robustness to linearization error.
- Two second safety horizon.
- Display current safety horizon and safe control set.
- Clip human input to safe control set (somehow. . .).