Modeling and Analysis of Hybrid Systems

7. Linear hybrid automata II

Prof. Dr. Erika Ábrahám

Informatik 2 - LuFG Theory of Hybrid Systems
RWTH Aachen University

Szeged, Hungary, 27 September - 06 October 2017

Contents

Contents

Some tools for hybrid automata reachability analysis

Tool	Characteristics
Ariadne	non-linear ODEs; Taylor models, boxes; interval constraint propagation, deduction
C2D2	non-linear ODEs; guaranteed simulation
Cora	non-linear ODEs; geometric representations; several algorithms, linear abstraction
dReach	non-linear ODEs; logical representation; interval constraint propagation, δ-reachability, bounded model checking
Flow*	non-linear ODEs; Taylor models; flowpipe construction
HSolver	non-linear ODEs; logical representation; interval constraint propagation
HyCreate	non-linear ODEs; boxes; flowpipe construction
HyPro	linear ODEs; several representations; flowpipe construction
HyReach	linear ODEs; support functions; flowpipe construction
HySon	non-linear ODEs; guaranteed simulation
iSAT-ODE	non-linear ODEs; logical representation; interval constraint propagation, bounded model checking
KeYmaera	differential dynamic logic; logical representation; theorem proving, computer algebra
NLTOOLBOX	non-linear ODEs; Bernstein expansion, hybridisation
SoapBox	linear ODEs; symbolic orthogonal projections; flowpipe construction
SpaceEx	linear ODEs; geometric and symbolic representations; flowpipe construction

We will learn how flowpipe-construction-based methods work. Flow* and HyPro were/are developed in our group. Besides them, most closely related is the SpaceEx tool.

Forward reachability computation

Input: Set Init of initial states.
Output: Set R of reachable states.

Algorithm:

$R^{\text{new}} := \text{Init};$
$R := \emptyset;$
while $(R^{\text{new}} \neq \emptyset)\{$
$\quad R := R \cup R^{\text{new}};$
$\quad R^{\text{new}} := \text{Reach}(R^{\text{new}}) \backslash R;$
$\};$
return R

Most well-known state set representations

Geometric objects:

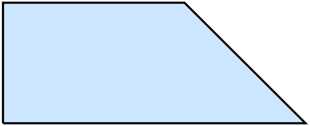
- hyperrectangles [Moore et al., 2009]
- oriented rectangular hulls [Stursberg et al., 2003]
- convex polyhedra [Ziegler, 1995] [Chen at el, 2011]
- orthogonal polyhedra [Bournez et al., 1999]
- template polyhedra [Sankaranarayanan et al., 2008]
- ellipsoids [Kurzhanski et al., 2000]
- zonotopes [Girard, 2005])

Other symbolic representations:

- support functions [Le Guernic et al., 2009]
- Taylor models [Berz and Makino, 1998, 2009] [Chen et al., 2012]

Reminder: Polytopes

- Halfspace: set of points satisfying $c^T \cdot x \leq z$
- Polyhedron: an intersection of finitely many halfspaces
- Polytope: a bounded polyhedron



representation	union	intersection	Minkowski sum
$\mathcal{V}$-representation by vertices	easy	hard	easy
$\mathcal{H}$-representation by facets	hard	easy	hard

Contents

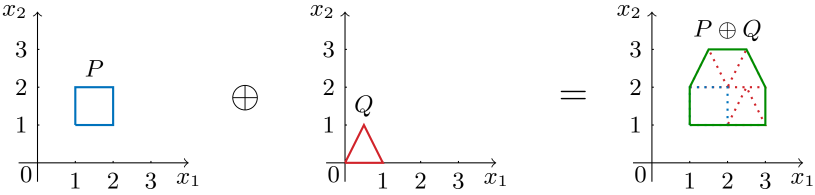
Linear hybrid automata I

Linear hybrid automata of type I (LHA I) are hybrid automata with the following restrictions:

- All derivatives are defined by intervals.
- All invariants and jump guards are defined by polytopes.
- All jump resets are defined by linear transformations of the form $x := Ax + b$.

Hybrid automata of this type have linear behaviour, i.e., when time passes by in a location, the values of the variables evolve according to a linear function. (To be more precise, each reachable state can be reached by such a linear evolution.)

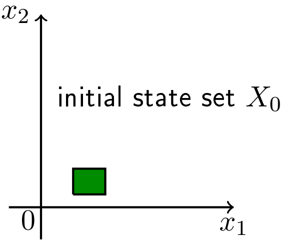
Reminder: Minkowski sum



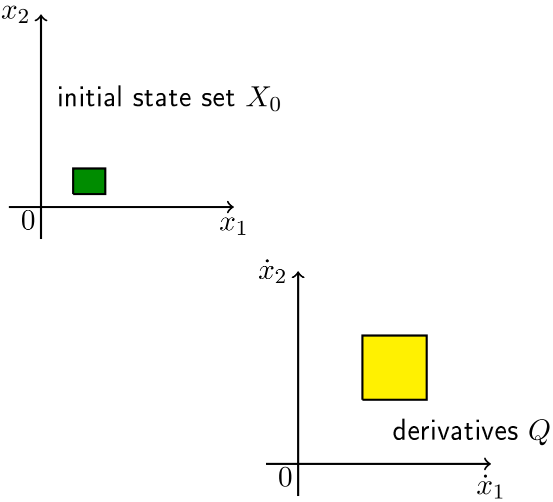
$$P \oplus Q = \{p + q \mid p \in P \text{ and } q \in Q\}$$

Linear hybrid automata I: Time evolution

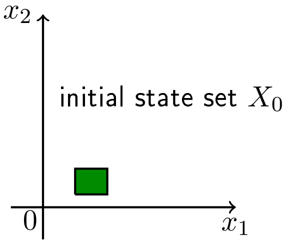
Linear hybrid automata I: Time evolution

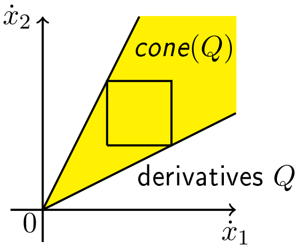


Linear hybrid automata I: Time evolution

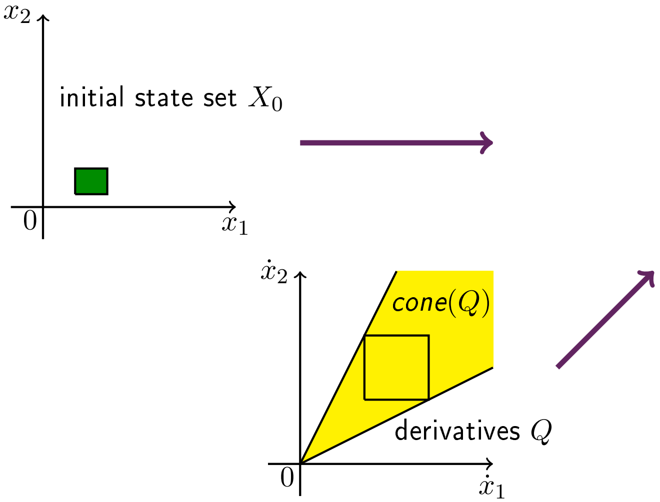


Linear hybrid automata I: Time evolution

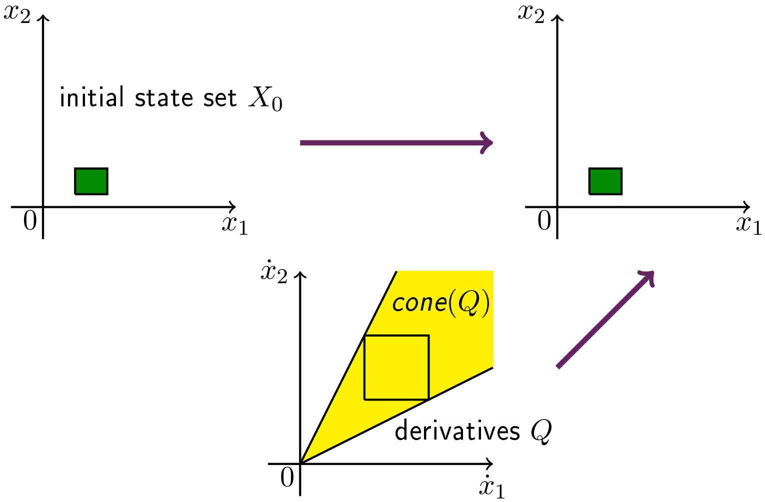




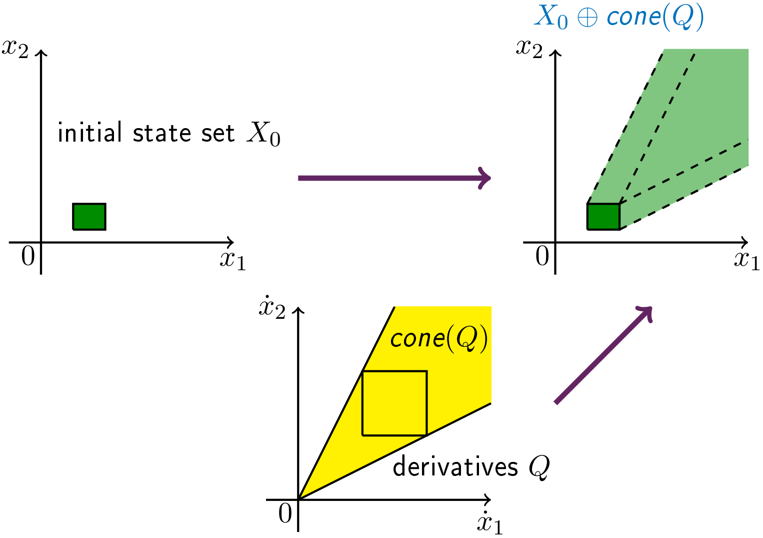
Linear hybrid automata I: Time evolution



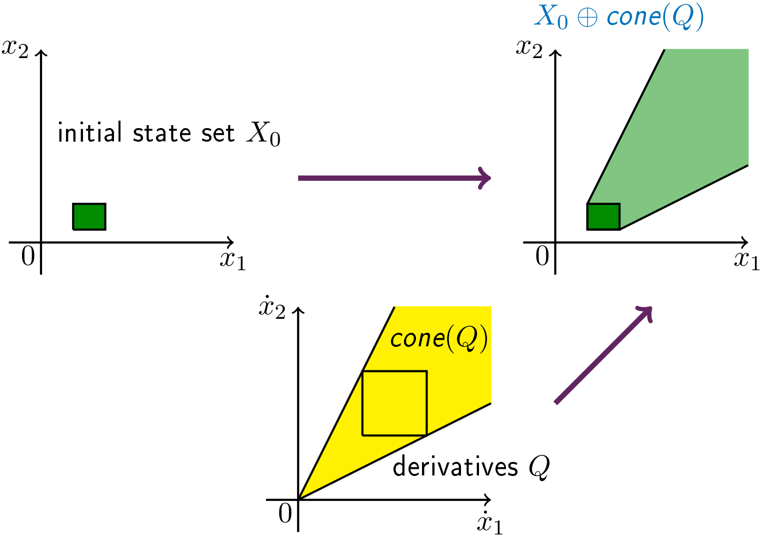
Linear hybrid automata I: Time evolution



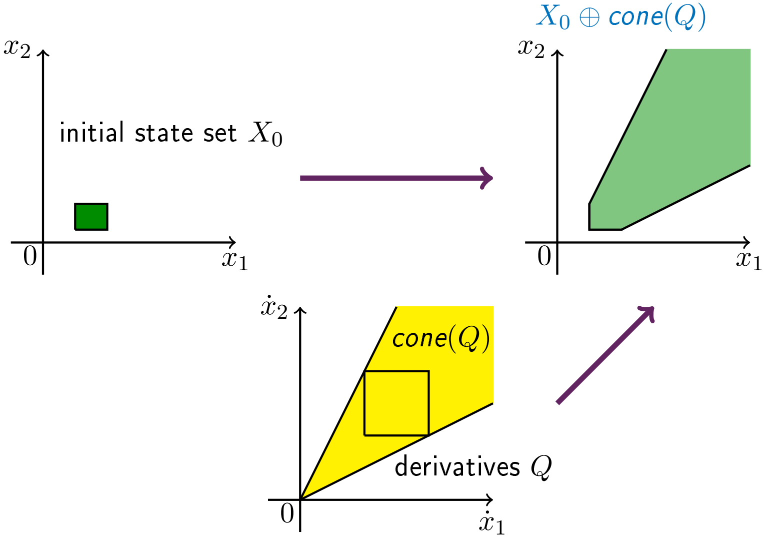
Linear hybrid automata I: Time evolution



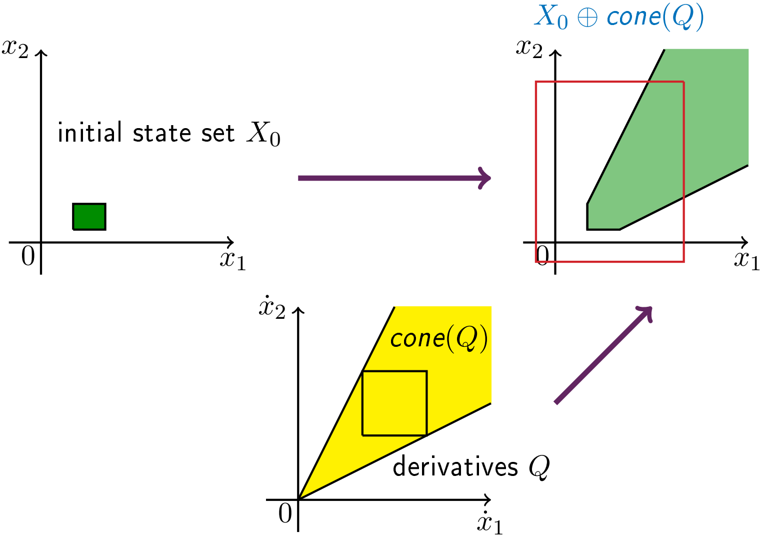
Linear hybrid automata I: Time evolution



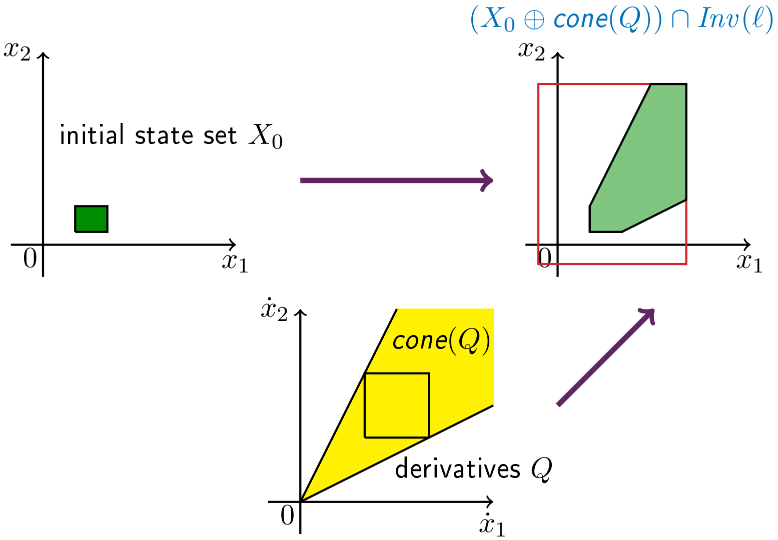
Linear hybrid automata I: Time evolution



Linear hybrid automata I: Time evolution

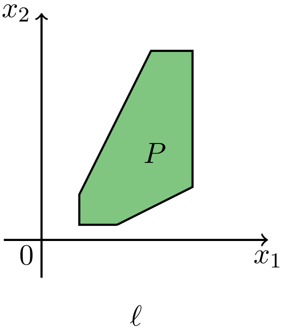


Linear hybrid automata I: Time evolution



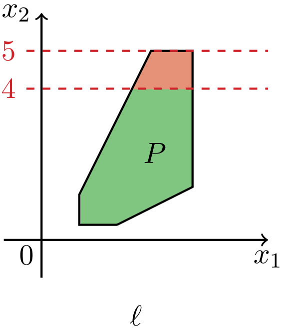
Linear hybrid automata I: Discrete steps (jumps)

Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$



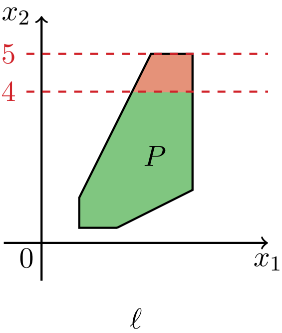
Linear hybrid automata I: Discrete steps (jumps)

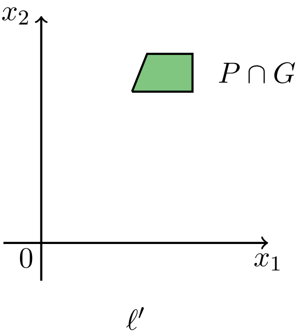
Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$



Linear hybrid automata I: Discrete steps (jumps)

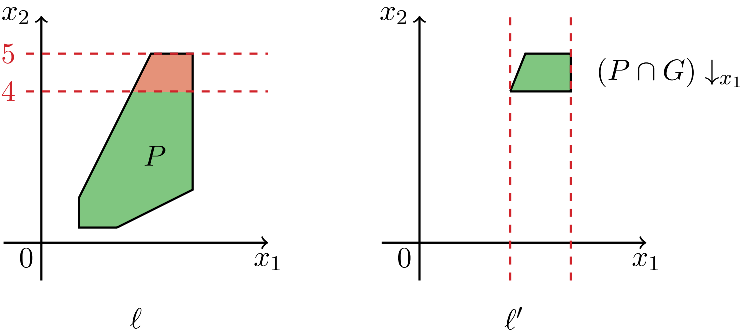
Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$





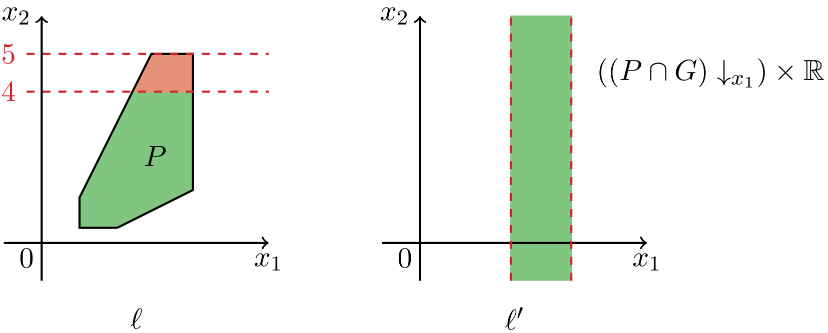
Linear hybrid automata I: Discrete steps (jumps)

Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$



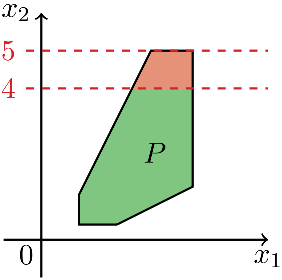
Linear hybrid automata I: Discrete steps (jumps)

Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R}\times[4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R}\times[2,4]}, \quad l' \quad) \in Edge$

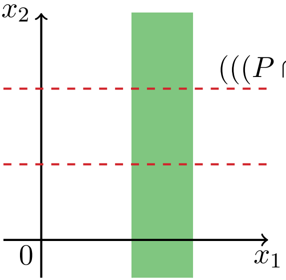


Linear hybrid automata I: Discrete steps (jumps)

Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in \mathit{Edge}$



ℓ

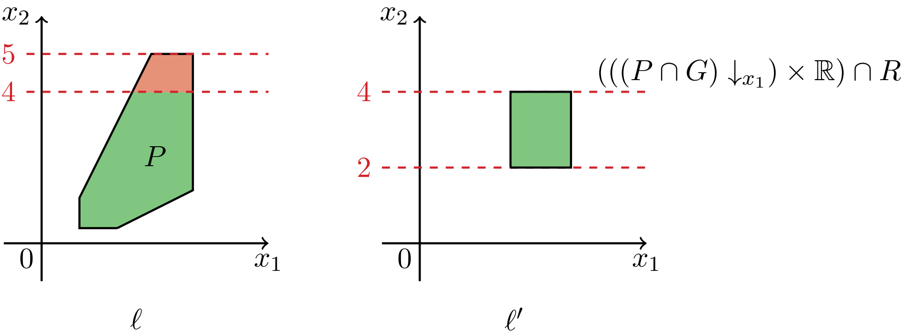


$$(((P \cap G) \downarrow_{x_1}) \times \mathbb{R}) \cap R$$

ℓ'

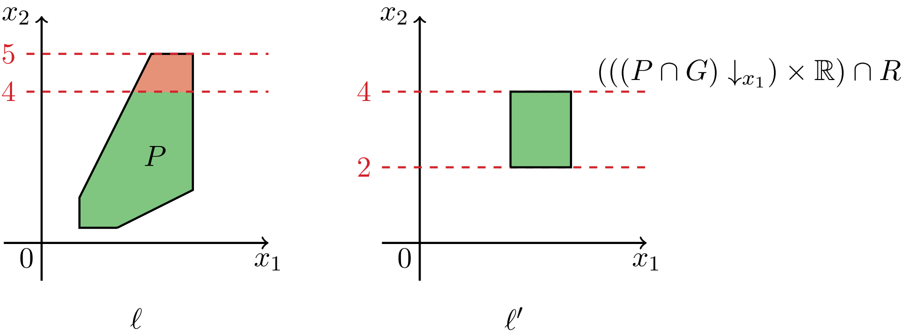
Linear hybrid automata I: Discrete steps (jumps)

Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$



Linear hybrid automata I: Discrete steps (jumps)

Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$



- Additionally, we intersect the result with the target location's invariant.
- Computed via projection, Minkowski sum and intersection.

Contents

Linear hybrid automata II

Linear hybrid automata of type II (LHA II) are hybrid automata with the following restrictions:

- All derivatives are defined by linear ordinary differential equations (ODEs) of the form
$$\dot{x} = Ax + Bu \ ,$$
where $x = (x_1, \ldots, x_n)^T$ are the (continuous) variables, A is a matrix of dimension $n \times n$, $u = (u_1, \ldots, u_m)^T$ are disturbance/input/control variables with rectangular domain U, and B is a matrix of dimension $n \times m$.
- All invariants and jump guards are defined by polytopes.
- All jump resets are defined by linear transformations of the form $x := Ax + b$.

When time passes by in a location, the values of the continuous variables evolve according to linear ODEs.
N.B.: Now the values follow non-linear functions!

Approximating a flowpipe

Consider a dynamical system with state equation

$$\dot{x} = f(x(t)).$$

Approximating a flowpipe

Consider a dynamical system with state equation

$$\dot{x} = f(x(t)).$$

We assume f to be Lipschitz continuous.

Approximating a flowpipe

Consider a dynamical system with state equation

$$\dot{x} = f(x(t)).$$

We assume f to be Lipschitz continuous.

Wikipedia: "Intuitively, a Lipschitz continuous function is limited in how fast it can change: for every pair of points on the graph of this function, the absolute value of the slope of the line connecting them is no greater than a definite real number [...]".

Approximating a flowpipe

Consider a dynamical system with state equation

$$\dot{x} = f(x(t)).$$

We assume f to be Lipschitz continuous.

Wikipedia: "Intuitively, a Lipschitz continuous function is limited in how fast it can change: for every pair of points on the graph of this function, the absolute value of the slope of the line connecting them is no greater than a definite real number [...]".

Lipschitz continuity implies the existence and uniqueness of the solution to an initial value problem, i.e., for every initial state x_0 there is a unique solution $x(t, x_0)$ to the state equation.

Approximating a flowpipe

The set of reachable states at time t from a set of initial states X_0 is defined as

$$\mathcal{R}_t(X_0) = \{x_t \mid \exists x_0 \in X_0.\ x_t = x(t, x_0)\}.$$

Approximating a flowpipe

The set of reachable states at time t from a set of initial states X_0 is defined as

$$\mathcal{R}_t(X_0) = \{x_t \mid \exists x_0 \in X_0.\ x_t = x(t, x_0)\}.$$

The set of reachable states, the flowpipe, from X_0 in the time interval $[0, T]$ is defined as

$$\mathcal{R}_{[0,T]}(X_0) = \cup_{t\in[0,T]}\mathcal{R}_t(X_0).$$

Approximating a flowpipe

The set of reachable states at time t from a set of initial states X_0 is defined as

$$\mathcal{R}_t(X_0) = \{x_t \mid \exists x_0 \in X_0.\ x_t = x(t, x_0)\}.$$

The set of reachable states, the flowpipe, from X_0 in the time interval $[0, T]$ is defined as

$$\mathcal{R}_{[0,T]}(X_0) = \cup_{t \in [0,T]} \mathcal{R}_t(X_0).$$

We describe a solution which approximates the flowpipe by a sequence of convex polytopes.

Problem statement for polyhedral approximation of flowpipes

Given

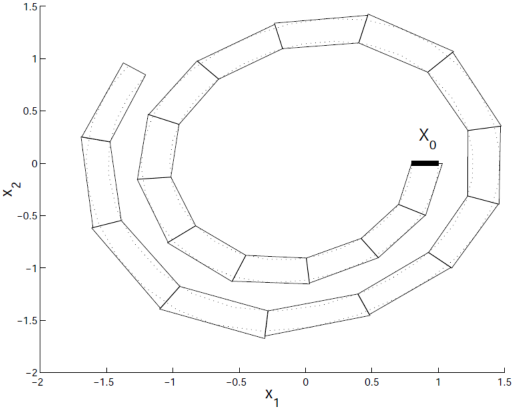
- a set X_0 of initial states which is a polytope, and
- a final time T,

compute a polyhedral approximation $\hat{\mathcal{R}}_{[0,T]}(X_0)$ to the flowpipe $\mathcal{R}_{[0,T]}(X_0)$ such that

$$\mathcal{R}_{[0,T]}(X_0) \subseteq \hat{\mathcal{R}}_{[0,T]}(X_0).$$

Flowpipe segmentation

Since a single convex polyhedron would strongly overapproximate the flowpipe, we compute a sequence of convex polyhedra, each approximating a flowpipe segment.



Segmented flowpipe approximation

Let the time interval $[0, T]$ be divided into $0 < N \in \mathbb{N}$ time segments

$$[0, t_1],\ [t_1, t_2],\ \ldots,\ [t_{N-1}, T]$$

with $t_i = i \cdot \delta$ for $\delta = \frac{T}{N}$.

Segmented flowpipe approximation

Let the time interval $[0, T]$ be divided into $0 < N \in \mathbb{N}$ time segments

$$[0, t_1],\ [t_1, t_2],\ \ldots,\ [t_{N-1}, T]$$

with $t_i = i \cdot \delta$ for $\delta = \frac{T}{N}$.

We generate an approximation $\hat{\mathcal{R}}_{[t_1,t_2]}(X_0)$ for each flowpipe segment:

$$\mathcal{R}_{[t_1,t_2]}(X_0) \subseteq \hat{\mathcal{R}}_{[t_1,t_2]}(X_0).$$

Segmented flowpipe approximation

Let the time interval $[0, T]$ be divided into $0 < N \in \mathbb{N}$ time segments

$$[0, t_1],\ [t_1, t_2],\ \ldots,\ [t_{N-1}, T]$$

with $t_i = i \cdot \delta$ for $\delta = \frac{T}{N}$.

We generate an approximation $\hat{\mathcal{R}}_{[t_1,t_2]}(X_0)$ for each flowpipe segment:

$$\mathcal{R}_{[t_1,t_2]}(X_0) \subseteq \hat{\mathcal{R}}_{[t_1,t_2]}(X_0).$$

The complete flowpipe approximation is the union of the approximation of all N pipe segments:

$$\mathcal{R}_{[0,T]}(X_0) \subseteq \hat{\mathcal{R}}_{[0,T]}(X_0) = \bigcup_{k=1,\ldots,N} \hat{\mathcal{R}}_{[t_{k-1},t_k]}(X_0)$$

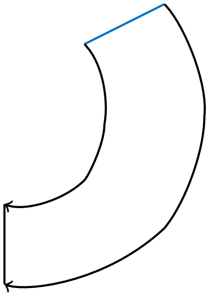
Approaches

Next we discuss one possible approach for flowpipe approximation, but there are different other techniques, too.

Linear hybrid automata II: Time evolution

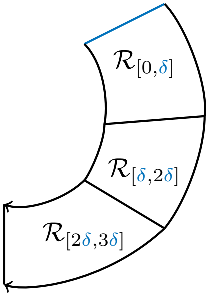
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$



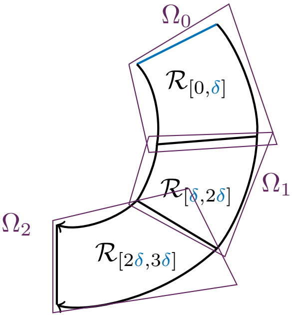
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$



Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$

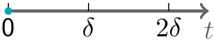


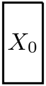
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:

Linear hybrid automata II: Time evolution

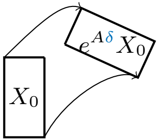
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:





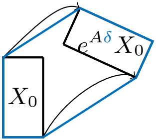
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



Linear hybrid automata II: Time evolution

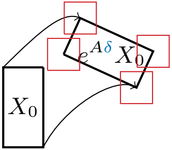
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



convex hull of $X_0 \cup e^{A\delta}X_0$

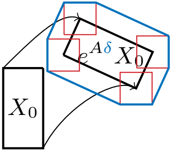
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



bloating with B_1 to include non-linear behaviour

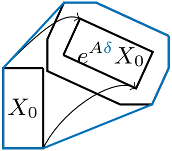
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



$(e^{A\delta}X_0) \oplus B_1$

Linear hybrid automata II: Time evolution

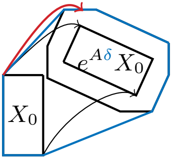
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



convex hull of $X_0 \cup ((e^{A\delta}X_0) \oplus B_1)$
covers the behaviour $\mathcal{R}_{[0,\delta]}$ under $\dot{x} = Ax$

Linear hybrid automata II: Time evolution

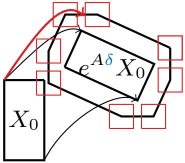
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



disturbance!

Linear hybrid automata II: Time evolution

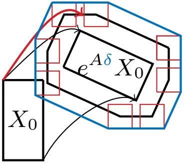
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



bloating with B_2

Linear hybrid automata II: Time evolution

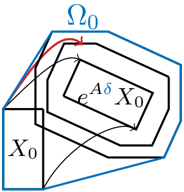
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



$$(e^{A\delta}X_0) \oplus B_1 \oplus B_2$$

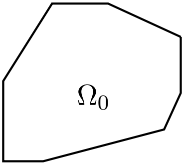
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



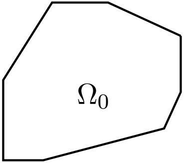
$\Omega_0 = conv(X_0 \cup ((e^{A\delta}X_0) \oplus B_1 \oplus B_2))$
covers the behaviour $\mathcal{R}_{[0,\delta]}$ under $\dot{x} = Ax + Bu$

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



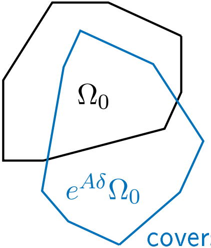
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:



Linear hybrid automata II: Time evolution

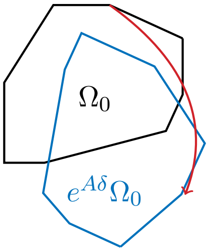
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:



covers the behaviour $\mathcal{R}_{[\delta,2\delta]}$ under $\dot{x} = Ax$

Linear hybrid automata II: Time evolution

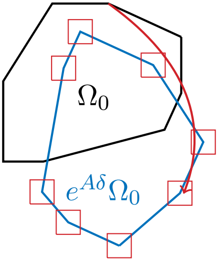
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:



disturbance!

Linear hybrid automata II: Time evolution

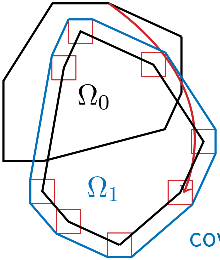
- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:



bloating with B_2

Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:

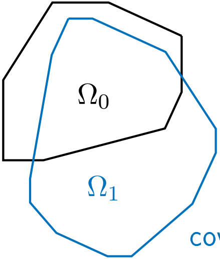


$\Omega_1 = (e^{A\delta}\Omega_0) \oplus B_2$

covers the behaviour $\mathcal{R}_{[\delta,2\delta]}$ under $\dot{x} = Ax + Bu$

Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute polytopes $\Omega_0, \Omega_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:

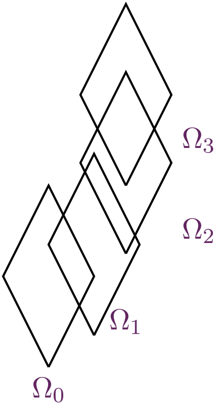


$$\Omega_1 = (e^{A\delta}\Omega_0) \oplus B_2$$

covers the behaviour $\mathcal{R}_{[\delta,2\delta]}$ under $\dot{x} = Ax + Bu$

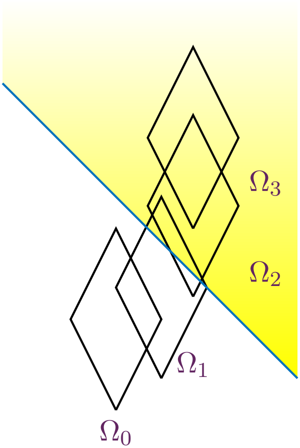
Linear hybrid automata II: Discrete steps (jumps)

The same procedure as for LHA I, extended with possibilities for aggregation and/or clustering.



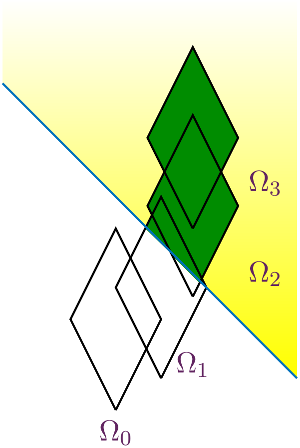
Linear hybrid automata II: Discrete steps (jumps)

The same procedure as for LHA I, extended with possibilities for aggregation and/or clustering.



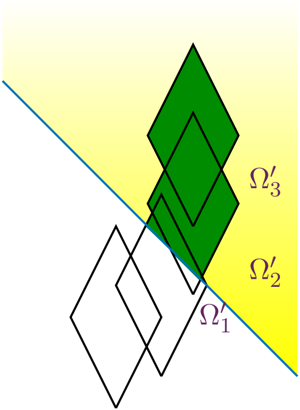
Linear hybrid automata II: Discrete steps (jumps)

The same procedure as for LHA I, extended with possibilities for aggregation and/or clustering.



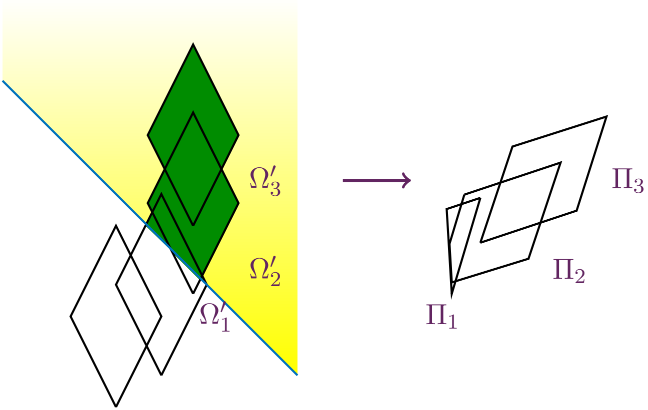
Linear hybrid automata II: Discrete steps (jumps)

The same procedure as for LHA I, extended with possibilities for aggregation and/or clustering.



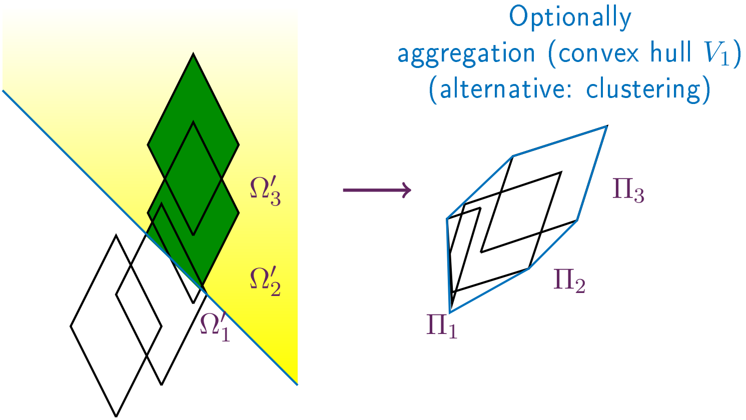
Linear hybrid automata II: Discrete steps (jumps)

The same procedure as for LHA I, extended with possibilities for aggregation and/or clustering.



Linear hybrid automata II: Discrete steps (jumps)

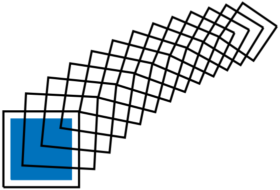
The same procedure as for LHA I, extended with possibilities for aggregation and/or clustering.



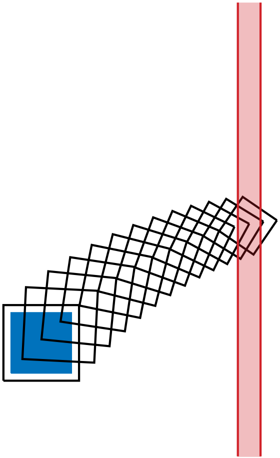
Linear hybrid automata II: The global picture

Linear hybrid automata II: The global picture

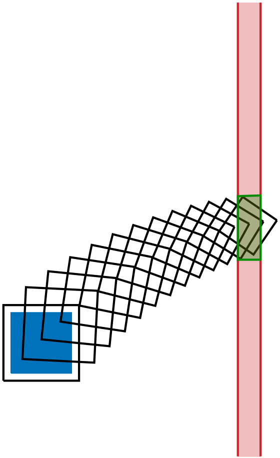
Linear hybrid automata II: The global picture



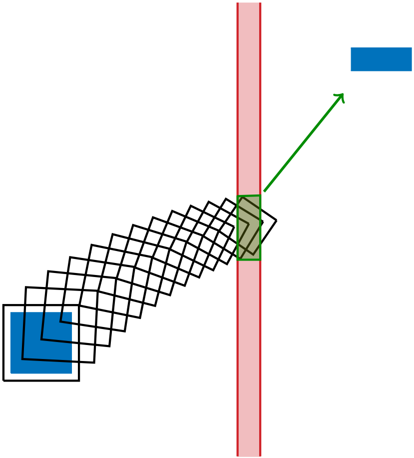
Linear hybrid automata II: The global picture



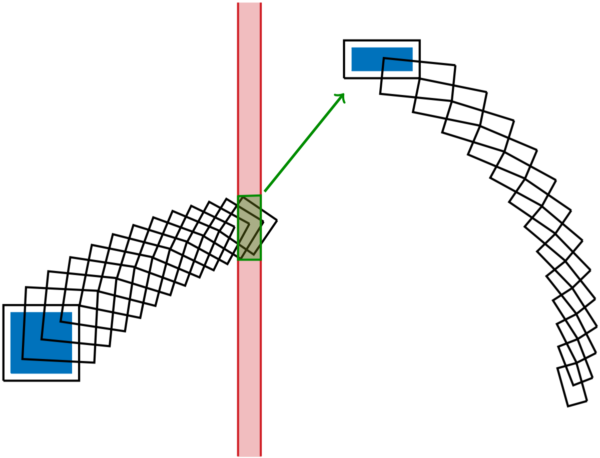
Linear hybrid automata II: The global picture



Linear hybrid automata II: The global picture



Linear hybrid automata II: The global picture

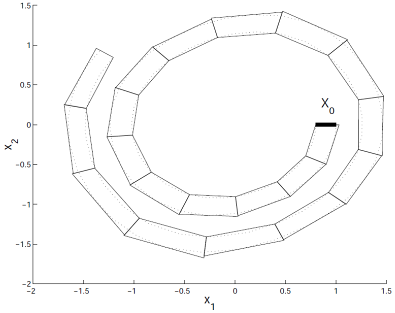


Example

- Van der Pol equation:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -0.2(x_1^2 - 1)x_2 - x_1.\end{aligned}$$

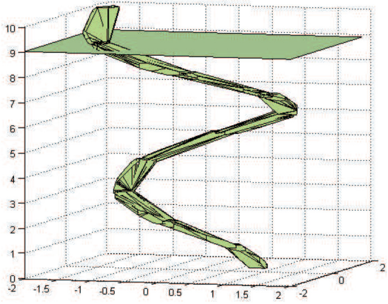
- Intial set: $X_0 = \{(x_1, x_2) \mid 0.8 \leq x_1 \leq 1 \wedge x_2 = 0\}$.
- Time: $T = 10$.
- Segments: 20



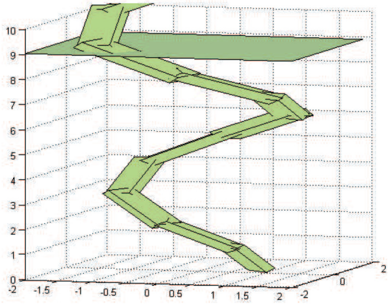
Other geometries for approximation

- Van der Pol equation with a third variable being a clock.
- Approximation

  with convex polyhedra and



  with oriented rectangular hull:



Partitioning the initial set

Var der Pol system with initial set $X_0 = \{(x_1, x_2) \mid 5 \leq x_1 \leq 45 \wedge x_2 = 0\}$.