A CSP versus a zonotope-based method for solving guard set intersection in nonlinear hybrid reachability

Moussa Maïga, Nacim Ramdani, Louise Travé-Massuyès, Christophe Combastel

A CSP versus a zonotope-based method for solving guard set intersection in nonlinear hybrid reachability

Moussa Maïga, Nacim Ramdani, Louise Travé-Massuyès and Christophe Combastel

Abstract. Computing the reachable set of hybrid dynamical systems in a reliable and verified way is an important step when addressing verification or synthesis tasks. This issue is still challenging for uncertain nonlinear hybrid dynamical systems. We show in this paper how to combine a method for computing continuous transitions via interval Taylor methods and a method for computing the geometrical intersection of a flowpipe with guard sets, to build an interval method for reachability computation that can be used with truly nonlinear hybrid systems. Our method for flowpipe guard set intersection has two variants. The first one relies on interval constraint propagation for solving a constraint satisfaction problem and applies in the general case. The second one computes the intersection of a zonotope and a hyperplane and applies only when the guard sets are linear. The performance of our method is illustrated on examples involving typical hybrid systems.

Keywords. Intervals analysis, hybrid systems, event detection, event localization, zonotope, reachability analysis.

1. Introduction

Hybrid systems are complex dynamical systems, which involve the interaction of discrete and continuous dynamics. This is the case for instance when real-time digital controllers interact with physical environments in our daily lives. Many complex systems exhibiting piecewise continuous dynamical behaviours, can be regarded as hybrid systems. Hybrid systems are often components of safety-critical systems. It is then necessary to have a thorough and guaranteed insight of their properties, such as performance, safety or stability [5, 44, 24]. The verification of these properties can be achieved through reachability analyses, some of which require an explicit computation of the hybrid state space, i.e. the set of all trajectories of the hybrid system starting from a possible initial set and under all admissible disturbances and variations in parameter values. This is a challenging issue since it is practically impossible to enumerate in a reliable way all the possible systems' behaviours.

Usual algorithms for hybrid reachability computation alternate the computation of continuous and discrete transitions, and most of them must address the challenging issue of event detection and localisation, that is computing the geometrical intersection of the continuous flowpipe with guard sets [2].

This issue has been addressed by many authors recently, although most of them focus on affine dynamics and polyhedral guard sets. Using support functions, [20] investigates ways of reducing the

This work is supported by the French National Research Agency under contract ANR 2011 INS 006 MAGIC-SPS (projects.laas.fr/ANR-MAGIC-SPS)..

over-approximation when computing the discrete transitions. The critical operation in this computation is the intersection of the flowpipe with the guard sets. They propose an approach for computing the intersection for affine dynamics with polyhedral guards. Reference [2] introduced a new approach for avoiding geometric intersection operations in reachability analysis by over-approximating the intersection of affine dynamics with guard sets modeled as halfspaces by nonlinear mappings.

Another family of effective but non-guaranteed methods combine polynomial approximations of the guard condition with root finding algorithms ([43, 8, 19, 18, 21, 39]), but uncertainties (about initial states, model parameters, ...) cannot be taken into account. Other approaches are based on zonotopes (a.o. [23, 2, 1, 3]), support functions ([26, 25, 20]) or polytopes [22] ; they scale well and are well adapted to affine dynamics.

When it comes to hybrid reachability computation with nonlinear hybrid systems, i.e. hybrid dynamical systems with nonlinear dynamics and guard sets described by nonlinear functions, there are very few truly nonlinear methods available, among which our previous work [41] and references [16, 28].

In [41], continuous transitions were addressed via set integration based on interval Taylor methods, and the event detection and localisation problems underlying flow/guard intersection was solved using propagation techniques for constraint-satisfaction problem (CSP). Recently, we introduced technical improvements of the latter method in order to reduce both the computation time while ensuring reduced over-approximation when crossing the guard set conditions. Our method now implements Lohner's QR-factorization [36] within the guaranteed numerical set integration method to control the wrapping effect, applies the standard contractor HC4Revise [9] for solving the CSP at discrete transition steps only, and makes use of bisection in a single dimension to ensure polynomial time complexity [34].

In this paper, we further improve this method to solve the flow/guard intersection directly without using any CSP solver when the guard sets are defined by hyperplanes. When a discrete transition is detected, we solve the event localisation problem as follows : we characterize the reached set as a zonotope, then we compute the zonotope enclosing the intersection between the reached set and the hyperplane defining the guard sets. This zonotope-based approach is compared to the CSP-based approach on two hybrid systems.

The paper is organized as follows. Sect. 2 defines hybrid dynamical systems. Sect. 3 overviews the interval tools used in the paper. Sect. 4 reviews our method for solving continuous transitions. Sect. 5 gathers our two methods for flow-guard sets intersection. Results are presented in Sect. 6 for two hybrid systems.

2. Uncertain nonlinear hybrid systems

Dynamical hybrid systems can be represented by a hybrid automaton [6] given by

$$HA = (\mathbb{Q}, \mathbb{D}, \mathbb{P}, \Sigma, \mathbb{A}, \mathrm{Inv}, \mathbb{F}) \tag{1}$$

and defined as follows:

- $\mathbb{Q}$ is a set of locations $\{q\}$ whose continuous dynamics, i.e. flow transitions, are described by non-autonomous differential equations $f_q \in \mathbb{F}$ of the form

$$\mathrm{flow}(q): \quad \dot{x}(t) = f_q(x, p, t), \tag{2}$$

  where $f_q : \mathbb{D} \times \mathbb{P} \times \mathbb{R}^+ \mapsto \mathbb{D}$ is nonlinear and assumed sufficiently smooth over $\mathbb{D} \subseteq \mathbb{R}^n$, with dimension n that may depend on q, and $p \in \mathbb{P}$, where $\mathbb{P}$ is an uncertainty domain for the parameter vector p.
- Inv is an invariant, which assigns a domain to the continuous state space of each location:

$$\mathrm{Inv}(q): \quad \nu_q(x(t), p, t) < 0, \tag{3}$$

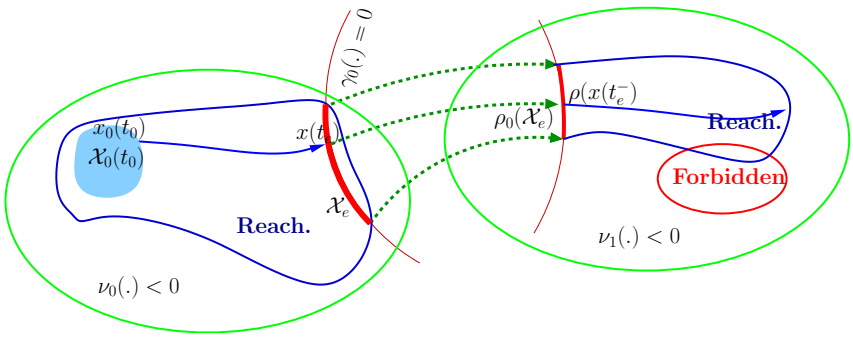


FIGURE 1. Set reachable in finite time by system (2-4).

where inequalities are taken componentwise, $\nu_q : \mathbb{D} \times \mathbb{P} \times \mathbb{R}^+ \mapsto \mathbb{R}^m$ is also nonlinear, and the number m of inequalities may also depend on q.

- $\mathbb{A}$ is the set of discrete transitions $\{e = (q \to q')\}$ given by the 5-tuple $(q, \text{guard}, \sigma, \rho, q')$, where q and q' represent upstream and downstream locations respectively; guard is a condition of the form:

$$\text{guard}(e): \quad \gamma_e(x(t), p, t) = 0; \tag{4}$$

σ is an event, and ρ is a reset function assumed to be affine.

A transition $q \to q'$ may occur when the continuous state flow reaches the guard set, i.e. when the continuous state satisfies condition (4).

The set reachable in finite time by system (2-4) is illustrated in Fig. 1. When starting from an initial state vector $x(t_0)$ taken in $\mathbb{X}(t_0)$, a discrete transition occurs when the continuous flow intersects the guard set at time t_e. Then, the continuous state vector is reset as $x_1(t_e^+) = \rho_0(x_0(t_e^-))$. $\mathbb{X}_e$ is the set of all possible vectors $x(t_e)$ when vector $x(t_0)$ varies in $\mathbb{X}(t_0)$. The reachable set may then intersect a forbidden area as shown in the figure.

Introducing the new state variable $z = (x, p, t)$ with $\dot{z} = (\dot{x}, 0, 1)$, and defining $\mathbb{Z} = \mathbb{D} \times \mathbb{P} \times \mathbb{R}^+$, equations (2–4) are rewritten:

$$\text{flow}(q): \quad \dot{z}(t) = f_q(z), \tag{5}$$

$$\text{Inv}(q): \quad \nu_q(z(t)) < 0 \quad \text{and} \tag{6}$$

$$\text{guard}(e): \quad \gamma_e(z(t)) = 0. \tag{7}$$

so that all uncertain quantities are embedded in the initial state vector.

When analyzing hybrid systems, intersections with guard sets that enable discrete transitions may occur. When a flowpipe of non-zero size reaches a guard condition described by linear or nonlinear constraints, there is a non-empty set of instants during which the constraint is satisfied, leading to a *continuum* of switching times [17].

3. Solving constraint satisfaction problems with interval methods

In this section, we overview key concepts regarding methods based on interval analysis that we use for finding intersections with invariants and guards, and evaluating jump functions.

Consider the system of m (in)equalities over n variables $z \in \mathbb{R}^n$

$$\mathscr{C}: \wedge_{1 \le i \le m}(h_i(z) \prec 0), \prec \in \{=, <\} \tag{8}$$

and denote the domain of z by $\mathscr{Z}$. Here, inequalities are considered when one computes mode invariants or prune solution sets from those parts that are not contained in invariants. Equalities are considered when one addresses flow/guard intersection and the evaluation of jump successors.

System (8) is a *numerical constraint satisfaction problem* CSP : $(\mathscr{Z}, \mathscr{C})$. Denoting by $\mathscr{S}$ its set of solutions, we have

$$\mathscr{S} = \{z \in \mathscr{Z} \mid \wedge_{1 \leq i \leq m} (h_i(z) \prec 0)\}. \tag{9}$$

An enclosure of $\mathscr{S}$ can be computed in a reliable and guaranteed way via branch-and-prune approaches using interval analysis and contractors based on constraint propagation [29].

3.1. Intervals analysis

A real interval $[u] = [\underline{u}, \overline{u}]$ is a closed and connected subset of $\mathbb{R}$ where $\underline{u}$ represents the lower bound of $[u]$ and $\overline{u}$ represents the upper bound. The width of an interval $[u]$ is defined by $\text{wid}([u]) = \overline{u} - \underline{u}$, its midpoint by $\text{mid}([u]) = (\overline{u} + \underline{u})/2$, and its radius by $\text{rad}([u]) = (\overline{u} - \underline{u})/2 = \text{wid}([u])/2$. An interval $[u]$ can be defined by its midpoint and its radius, so $[u] = [\text{mid}([u]) - \text{rad}([u]), \text{mid}([u]) + \text{rad}([u])$, the unitary interval is $\mathbf{B} = [-1, 1]$. The set of all real intervals of $\mathbb{R}$ is denoted $\mathbb{IR}$. Two intervals $[u]$ and $[v]$ are equal if and only if $\underline{u} = \underline{v}$ and $\overline{u} = \overline{v}$. Real arithmetic operations are extended to intervals [35]. Arithmetic operations on two intervals $[u]$ and $[v]$ can be defined as:

$$\circ \in \{+, -, *, /\}, \;\; [u] \circ [v] = \{x \circ y \mid x \in [u], \; y \in [v]\}.$$

An interval vector (or box) $[X]$ is a vector with interval components and may equivalently be seen as a Cartesian product of scalar intervals: $[X] = [x_1] \times [x_2] ... \times [x_n]$. The set of $n-$dimensional real interval vectors is denoted by $\mathbb{IR}^n$. A unitary box in $\mathbb{IR}^n$, denoted by $\mathbf{B}^n$, is a box composed by n unitary intervals.

An interval matrix is a matrix with interval components. The set of $n \times m$ real interval matrices is denoted by $\mathbb{IR}^{n \times m}$. Classical operations for interval vectors (resp. interval matrices) are direct extensions of the same operations for point vectors (resp. point matrices) [35]. Let $f : \mathbb{R}^n \to \mathbb{R}^m$, the range of function f over an interval vector $[u]$ is given by:

$$f([u]) = \{f(x) | x \in [u]\}.$$

The interval function $[f]$ from $\mathbb{IR}^n$ to $\mathbb{IR}^m$ is an inclusion function for f if:

$$\forall [u] \in \mathbb{IR}^n, \;\; f([u]) \subseteq [f]([u]).$$

An inclusion function of f can be obtained by replacing each occurrence of a real variable by its corresponding interval and by replacing each standard function by its interval evaluation. Such a function is called the natural inclusion function. In practice the inclusion function is not unique, it depends on the formal expression of f.

3.2. Branch-and-prune algorithms

Consider system (8) and the case when " $\prec$ " is " $=$ ". Constraint-satisfaction algorithms work as follows.

Algorithm 1: Interval-Solve

input : '$\wedge_{1\leq i\leq m} h_i([z]) = 0$', $\mathscr{Z}, \varepsilon_1, \varepsilon_2$
output: list of boxes $\overline{\mathscr{S}}$

define a running list of boxes $\mathscr{L}$ and initialize it with $[z] = \text{Hull}(\mathscr{Z})$;
while $\mathscr{L} \neq 0$ **do**
  pick first box $[z]$ from the list;
  evaluate $h_i([z])$;
  if $\exists i : 0 \notin h_i([z])$ **then**
    discard box $[z]$;
  else if *((*$\max \text{wid}([z]) < \varepsilon_1$*) or (*$\max_i \text{wid}(h_i([z])) < \varepsilon_2$*))* **then**
    store box $[z]$ in list $\overline{\mathscr{S}}$
  else
    partition $[z]$ and store new boxes in $\mathscr{L}$;
  end if
end while

Clearly, this simple algorithm is of exponential complexity. There are several technical and heuristic improvements, which make it possible to control the overall computation time and memory usage [29, 27]. They also address partitioning strategies, and the possible use of interval narrowing procedures on box $[z]$.

3.3. Contractors

The idea underlying contractors is to use a reduction function that reduces the size of box $[z]$ during the branching scheme of algorithm `Interval-Solve` without using bisection. This reduction can be achieved by an interval narrowing operator, a contractor for (8) on $[z]$, which we write as

$$[z]' = \texttt{Contractor}(\mathscr{C}, [z]).$$

This operator removes from $[z]$ subsets that do not contain a solution to (8) and satisfies the following properties: (*a*) $[z]' \subseteq [z]$ and (*b*) $[z]' \cap \mathscr{S} = [z] \cap \mathscr{S}$, where $\mathscr{S}$ is the solution set (9).

Most contractors use consistency filtering techniques (e.g. [12], see also the review [38]) and/or constraint propagation [29]. Interval propagation techniques are based on the interval extension of the local Waltz filtering [15, 45, 11]. Consistency filtering techniques rely on local consistency properties.

4. Continuous transitions using set integration via Interval Taylor methods

For better readability, we briefly overview in this section, the main ideas underlying continuous reachability analysis using guaranteed set integration via interval Taylor methods.

Consider the *uncertain* dynamical system described by (5)–(7) with $z(t_0) \in \mathbb{Z}_0$ at time $t_0 \geq 0$ and denote by $\mathbb{Z}(t; t_0, \mathbb{Z}_0)$ the set of solutions of (5) at time t originating from each initial condition in $\mathbb{Z}_0$ at t_0. $\mathbb{Z}(t; t_0, \mathbb{Z}_0)$ is abbreviated as $\mathbb{Z}(t)$ when there is no ambiguity.

Define a time grid $t_0 < t_1 < t_2 < \ldots < t_{n_T}$, which needs not to be equally spaced in this paper, and assume initial domain is an interval vector; i.e. $\mathbb{Z}_0 = [z_0] = [\underline{z}_0, \overline{z}_0]$.

Then, guaranteed set integration via interval Taylor methods computes interval vectors $[z_j], j = 1, \ldots, n_T$, that are *guaranteed* to contain the set of solutions $\mathbb{Z}(t_j; t_0, \mathbb{Z}_0)$ of (5) at times t_j, $j = 1, \ldots, n_T$ in three stages:

- verify the existence and uniqueness of the solution using the Banach fixed point theorem and the Picard-Lindelöf operator [35, 14, 37].

- compute an a priori enclosure $[\widetilde{z}_j]$ such that $[\widetilde{z}_j] \supseteq \mathbb{Z}(t)$ for all t in $[t_j, t_{j+1}]$. Hence, $[\widetilde{z}_j]$ is indeed an over-approximation of the reachable set over $[t_j, t_{j+1}]$. It can be made as tight as possible using the following stage.
- compute a tighter enclosure for the set of solutions of (5) at t that can be taken as t_{j+1} or any $t \in [t_j, t_{j+1}]$ not necessarily on the time-grid, using a Taylor series expansion of order k of the solution at t_j, and where $[\widetilde{z}_j]$ is used to enclose the remainder term:

$$\mathbb{Z}(t;t_j,[z_j]) \subseteq [z](t;t_j,[z_j]) = [z_j] + \sum_{i=1}^{k-1} (t-t_j)^i \mathsf{f}_q^{[i]}([z_j]) + (t-t_j)^k \mathsf{f}_q^{[k]}([\widetilde{z}_j]), \tag{10}$$

and where the $\mathsf{f}_q^{[i]}([z_j])$ are the Taylor coefficients.

Remark 1. *Eq. (*10*) written for any t in $[t_j, t_{j+1}]$ is an extension of what is classically done for guaranteed set integration, since latter methods aim at computing tight enclosures for time instants taken on the grid. Here, for solving the flow/guard intersection we need to obtain an explicit characterization of the solution for any time instant taken between two time grid points. Eq. (*10*) can be viewed as a conservative polynomial interpolation, hence acts as an analytical solution for the flowpipe for t in $[t_j, t_{j+1}]$, since $\mathsf{f}_q^{[k]}([\widetilde{z}_j])$ encloses the remainder of the Taylor series for any t in $[t_j, t_{j+1}]$ [40].*

Extending the results from [36], (10) can be turned into a computationally acceptable scheme that controls the wrapping effect[1] by using the *mean-value* approach complemented by QR-factorization as proposed by Lohner. The solution enclosure at time $t \in [t_j, t_{j+1}]$ can be computed in the mean-value form :

$$\mathbb{Z}(t;t_j,\mathbb{Z}_0) \in \{v + A(t)r \,|\, v \in [v](t), r \in [r](t)\},$$

and

$$\mathbb{Z}_{j+1} = \mathbb{Z}(t_{j+1};t_j,\mathbb{Z}_0) \in \{v + A_{j+1}r \,|\, v \in [v_{j+1}], r \in [r_{j+1}]\},$$

Defining:

$$[\chi](t) \equiv \{[z](t), \hat{z}(t), [v](t), [r](t), A(t)\}, \tag{11}$$

where $\hat{z}(t) := \text{mid}([z](t))$, the algorithm $\varphi^{QR}(.)$ given in **Algorithm** 2 is used to compute the solution set of (5) at time $t \in [t_j, t_{j+1}]$ [41]. The solution enclosure at time t_{j+1} is given by $[\chi_{j+1}] = \varphi^{QR}([\chi_j], t_j, t_{j+1}, [\widetilde{z}_j])$.

Algorithm 2: Algorithm φ^{QR}

input : $[\chi_j]$, t_j, t, $[\widetilde{z}_j]$
output: $[\chi](t)$

1 $[v](t) := \hat{z}_j + \sum_{i=1}^{k-1} (t-t_j)^i \mathsf{f}_q^{[i]}(\hat{z}_j) + (t-t_j)^k \mathsf{f}_q^{[k]}([\widetilde{z}_j])$;
2 $[S](t) := \mathbb{I} + \sum_{i=1}^{k-1} (t-t_j)^i \frac{\partial \mathsf{f}_q^{[i]}}{\partial z}([z_j])$;
3 $[q](t) := ([S](t)A_j)[r_j] + [S](t)([v_j] - \hat{z}_j)$;
4 $[z](t) := [v](t) + [q](t)$;
5 obtain $A(t)$ via QR-factorization of $\text{mid}([S](t)A_j)$ [33];
6 $[r](t) := A(t)^{-1}([S](t)A_j)[r_j] + (A(t)^{-1}[S])([v_j] - \hat{z}_j)$;
7 $\hat{z}(t) := \text{mid}([v](t))$;
8 $[\chi](t) := \{[z](t), \hat{z}(t), [v](t), [r](t), A(t)\}$;

[1] The wrapping effect is the over-approximation induced by enclosing a set of any shape in an axis-aligned box.

5. Computing the geometrical intersection of a flowpipe and guard sets

We now show how to compute the geometrical intersection of a continuous flowpipe with the guard sets[2]. For completeness, we overview a preliminary version of an algorithm based on solving a CSP, then we introduce a variant using zonotopes. These two methods will be compared on the same examples in the next section.

5.1. Event detection and localization

The issue is first to detect if the flowpipe intersects a guard set, then to compute when and where the intersection occurs, in other words we need to compute the time instants t_e and solution state vector $z(t_e)$ such that (7) is satisfied, i.e. $\gamma_e(z(t_e)) = 0$.

Because the flowpipe has a non-zero width, there is a continuum of time instants where the intersection occurs. Hence, we need to characterize the set of all such solutions, i.e.

$$\mathscr{T}^\star \times \mathscr{Z}^\star = \left\{ t_e \times z(t_e) \text{ s. t. } (t_e \in [t_j, t_{j+1}]) \wedge (\gamma_e(z(t_e)) = 0) \wedge (\dot{z}(t) = f_q(z)) \wedge (z(t_j) \in [z_j]) \right\} \quad (12)$$

Let us assume an event exists for t_e in $[t_j, t_{j+1}]$, then the methods described in the sequel are able to detect the existence of such event.

Computing solution set (12) is now an analytical problem since algorithm $\varphi^{QR}(t)$ yields for any t in $[t_j, t_{j+1}]$ an analytical solution for the flowpipe over the time interval $[t_j, t_{j+1}]$, hence the methods described in section 3 applies directly. To obtain a tight characterization of (12), we need to partition the search space. Therefore, to curb computational complexity and keep a polynomial computation time, we partition only in the single direction of the time variable, and use contractors to reduce the solution set at a given time instant.

Let us denote $[\underline{t}^\star, \overline{t}^\star]_l \subseteq [t_j, t_{j+1}]$ a sub-interval over which (7) is satisfied, and $\mathscr{Z}_l^\star$ the set of state vectors for which t exists in $[\underline{t}^\star, \overline{t}^\star]_l$ that satisfies (7). Solution set (12) can then be over-approximated by

$$\mathscr{T}^\star \times \mathscr{Z}^\star \subseteq \bigcup_{l=1}^{L} [\underline{t}^\star, \overline{t}^\star]_l \times \mathscr{Z}_l^\star \quad (13)$$

where L is the number of solution sub-boxes.

Allowing some over-approximation when computing t_e, we say that an event occurs over the subinterval $[\underline{t}^\star, \overline{t}^\star]_l$ if $\text{wid}\,([\underline{t}^\star, \overline{t}^\star]_l)$ is smaller than a given threshold ε_T, as suggested in [41].

Now the question remains on how to compute a tight over-approximation of $\mathscr{Z}_l^\star$. In the sequel, we show two ways to do this.

The first one solves a constraint satisfaction problem as in [34] and applies to nonlinear guard sets. The second one computes directly the intersection of the enclosure (11) with the guard set and applies when the latter is described as a hyperplane.

5.2. Computing the intersection by solving a CSP

Recall that the enclosure of the solution of the IVP ODE (5) can be computed for the small time interval $[t^\star]_l = [\underline{t}^\star, \overline{t}^\star]_l$ in the compound form (11), denoted $[\chi]_l$, using an inclusion function for algorithm $\varphi^{QR}(t)$, which we obtain by using an interval $[t]$ as input. We will save the notation $[\chi^\star]_l$ for the compound form characterizing the tight over-approximation of $\mathscr{Z}_l^\star$. Removing the l subscript for simplicity, we have

$$[\chi] = \{[z], \hat{z}, [v], [r], A^\star\}. \quad (14)$$

Therefore

$$(\exists z \in [z] \text{ s.t. } \gamma_e(z) = 0) \Rightarrow (\exists v \times r \in [v] \times [r] \text{ s.t. } \gamma_e(v + A^\star r) = 0), \quad (15)$$

hence, computing the intersection over the time interval $[t^\star]$ boils down to solving the CSP

$$\mathscr{C} : \left([v] \times [r], \text{“}\gamma_e\left(v + A^\star r\right) = 0\text{”}\right). \quad (16)$$

[2]The same computational methods apply for the intersection with invariant sets.

Using a contractor, we can obtain a tight over-approximation of $\mathscr{Z}^\star$ as a compound form $[\chi^\star]_l$ where

$$[v^\star] \times [r^\star] = \texttt{Contractor}(\mathscr{E}, [v] \times [r]). \tag{17}$$

If the solution set for CSP (17) is not empty, we assume that the event $e = q \to q'$ occurs at $t_e = \underline{t}^\star$ and that $[\chi](t_e^-) = [\chi^\star]$, as suggested in [41]. The discrete transition can then be computed from there, using the reset function ρ.

5.3. Computing the intersecting from a zonotope and a hyperplane

We now show how to compute the flowpipe/guard set intersection by using the intersection of a zonotope and a hyperplane. We extend the approach initially given in [13] for computing the intersection of two zonotopes. Compared to the previous scheme as presented in 5.2, it reduces to simple linear algebra computations (mainly, evaluation of a null space), which ensures polynomial complexity and small computation time as the space dimension increases. Let us first review zonotope definitions and then introduce our main results.

5.3.1. Zonotopes. Given a vector $c \in \mathbb{R}^n$ and a matrix $R \in \mathbb{R}^{n\times p}$, a zonotope [32] $\mathtt{Z}$ is the set $\mathtt{Z} = c \oplus R\mathbf{B}^p = \{c + Rx : x \in \mathbf{B}^p\}$. The vector c is the center of the zonotope, and the matrix R defines the shape of the (centrally symmetric) zonotopic domain. $\mathtt{Z}$ is the Minkowski sum[3] of the m-segments defined as m columns of matrix R in $\mathbb{R}^{n\times p}$.

Property 1. *The Minkowski sum of two zonotopes* $\mathtt{Z}_1 = c_1 \oplus R_1\mathbf{B}^{p_1} \in \mathbb{R}^n$ *and* $\mathtt{Z}_2 = c_2 \oplus R_2\mathbf{B}^{p_2} \in \mathbb{R}^n$ *is also a zonotope, defined as* $\mathtt{Z}_1 \oplus \mathtt{Z}_2 = (c_1 + c_2) \oplus [R_1\ R_2]\mathbf{B}^{p_1+p_2}$.

Property 2. *The image of a zonotope* $\mathtt{Z} = c \oplus R\mathbf{B}^p \in \mathbb{R}^n$ *by a linear mapping L can be computed by a standard matrix product* $L\mathtt{Z} = Lc \oplus (LR)\mathbf{B}^p$.

5.3.2. Computing the intersection of a zonotope and a hyperplane.

Proposition 5.1. *Consider a zonotope* $\mathtt{Z} = c \oplus R\mathbf{B}^p$ *and a hyperplane* $\mathscr{H}(\eta, d) = \{x | \eta^\top x = d\}$ *with non empty intersection (which can easily be tested using support vector* η*). Let us define :* $\sigma = R^\top\eta$, $d' = d - \eta^\top c$, $c_s = \sigma d'/\|\sigma\|_2^2$, $V_0 = N(\sigma^\top)$, *i.e. a basis of the null space (or kernel) of* $\sigma^\top$, $R_s = V_0V_0^\top$, $c_\cap = c + Rc_s$, *and* $R_\cap = RR_s$, *then a zonotope enclosing the intersection between* $\mathtt{Z}$ *and the hyperplane* $\mathscr{H}(\eta, d)$ *is given by*

$$\mathtt{Z} \cap \mathscr{H} \subseteq \mathtt{Z}_\cap = c_\cap \oplus R_\cap\mathbf{B}^p. \tag{18}$$

Proof. Let $x \in \mathtt{Z}$, then $s \in \mathbf{B}^p$ exists such that $x = c + Rs$. The point vector x also belongs to the hyperplane $\mathscr{H}(\eta, d)$ if $\eta^\top x = d$, hence if $\eta^\top c + \eta^\top Rs = d$. Then, introducing $\sigma = R^\top\eta$ and $d' = d - \eta^\top c$, we can deduce the necessary and sufficient condition

$$x \in \mathtt{Z} \cap \mathscr{H} \Leftrightarrow x \in \{c \oplus Rs \,|\, s \in \mathbf{B}^p \wedge \sigma^\top s = d'\}. \tag{19}$$

Bounding $\mathtt{Z} \cap \mathscr{H}(\eta, d)$ boils down to finding all $s \in \mathbf{B}^p$ such that $\sigma^\top s = d'$. This equation can be solved using the singular value decomposition (SVD) of vector $\sigma^\top$, which takes the particular form

$$\sigma^\top = U \cdot S \cdot V^\top = \begin{bmatrix} U_1 \end{bmatrix} . \begin{bmatrix} S_1 & 0_{1\times p-1} \end{bmatrix} . \begin{bmatrix} V_1^\top \\ V_0^\top \end{bmatrix}, \tag{20}$$

where $U_1 = 1$ is scalar, $S_1 = \|\sigma\|_2$, $V_1 = \frac{\sigma}{\|\sigma\|_2}$, $V_0 = N(\sigma^\top)$, and where $\|\sigma\|_2$ is the Euclidean norm of σ ($\|\sigma\|_2^2 = \sigma^\top\sigma$) and $N(\sigma^\top)$ is an orthonormal basis of the kernel of $\sigma^\top$.
Let $V = [V_1, V_0]$ be viewed as a change of coordinates, and define point vector $\xi = (\xi_1, \xi_0)$ which satisfies $s = V\xi = V_1\xi_1 + V_0\xi_0$. Since $V^\top V = I$, we have $V_1^\top s = \xi_1$ and $V_0^\top s = \xi_0$. From $\sigma^\top s = d'$ and $\sigma^\top = U_1S_1V_1^\top$, it comes $U_1S_1\xi_1 = d'$ and $\xi_1 = \frac{d'}{\|\sigma\|_2}$.

[3] Let $\xi_1, \xi_2 \subset \mathbb{R}^n$, the Minkowski sum of ξ_1 and ξ_2 is: $\xi_1 \oplus \xi_2 = \{s_1 + s_2 | s_1 \in \xi_1, s_2 \in \xi_2\}$.

Recall that $\xi_0 = V_0^\top s$. Since $s \in \mathbf{B}^p$, then $\xi_0 \in V_0^\top \mathbf{B}^p$. From $s = V_1\xi_1 + V_0\xi_0$ we have $s \in c_s \oplus V_0 V_0^\top \mathbf{B}^p$. Finally, an over-approximation $[s]$ of the set of all $s \in \mathbf{B}^p$ that satisfies $\sigma^\top s = d'$ is given by

$$[s] = c_s \oplus R_s \mathbf{B}^p. \tag{21}$$

By combining (19) and (21), it comes:

$$x \in \mathtt{Z} \cap \mathscr{H} \Rightarrow x \in c \oplus R(c_s \oplus R_s \mathbf{B}^p) \Leftrightarrow x \in (c + Rc_s) \oplus RR_s \mathbf{B}^p. \tag{22}$$

Introducing $c_\cap = c \oplus Rc_s$, and $R_\cap = RR_s$, we have

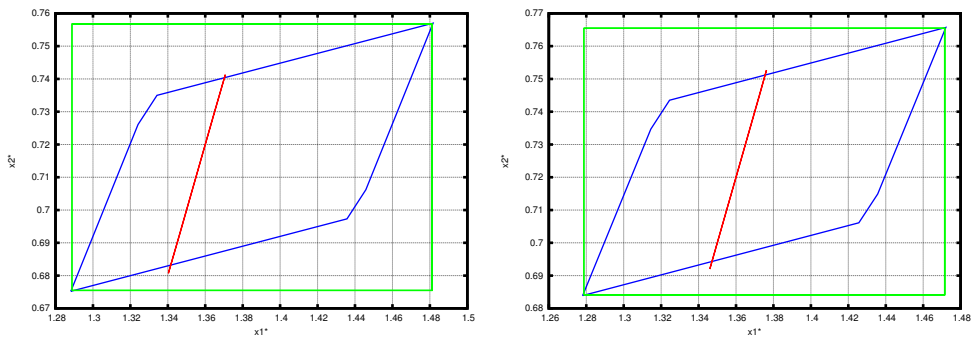
$$x \in \mathtt{Z} \cap \mathscr{H} \Rightarrow x \in \mathtt{Z}_\cap = c_\cap \oplus R_\cap \mathbf{B}^p. \tag{23}$$

□

Corollary 1. *By construction, the zonotope* $\mathtt{Z}_\cap$ *enclosing the intersection as defined in* (19)*, is included in the hyperplane* $\mathscr{H}$*.*

Proof. Let $x \in c_\cap \oplus R_\cap \mathbf{B}^p$ then $\exists s \in \mathbf{B}^p$ such that $x = c_\cap + R_\cap s$, then $\eta^\top x = \eta^\top (c + R\sigma d'/\|\sigma\|_2^2 + RV_0V_0^\top s) = \eta^\top c + \|\sigma\|_2^2 d'/\|\sigma\|_2^2 + \sigma^\top V_0 V_0^\top s$, because $\sigma^T = \eta^T R$. Since $\sigma^\top V_0 = 0$ by construction of V_0, and since $d' = d - \eta^\top c$, $\eta^\top x = d$ which completes the proof. □

Finally, proposition 5.1 and equation (18) give a computational scheme for computing an over-approximation of $\mathtt{Z} \cap \mathscr{H}$ summarized in **Algorithm** 3. This approximation may feature some conservativeness but we found that it is reasonably tight for our applications, as illustrated by Fig. (2).



(a) Instant $t_1^\star$ (b) Instant $t_2^\star$

FIGURE 2. Initial zonotope in blue color, enclosed by an axis-aligned box in green color, the zonotope enclosing the intersection with the hyperplane is depicted in red, for two time instants

5.3.3. Main algorithm. As in section 5.2, we consider the flowpipe $\mathscr{Z}_l$ computed over a subinterval $[t^\star]_l = [\underline{t}^\star, \overline{t}^\star]_l$ using an inclusion function for algorithm $\varphi^{QR}(t)$ and given in the compound form (11). Removing again l subscript for simplicity, we have

$$\mathscr{Z} = \{v + A^\star r \,|\, v \in [v], r \in [r]\} \subseteq A^\star [r] \oplus [v]. \tag{24}$$

It is easy to see that domain $\mathscr{Z}$ is a MSPB i.e the Minkowski sum of the parallelotope $A^\star[r]$ and the axis-aligned box $[v]$ (see Fig. 3). Furthermore, it is worth noticing that each domain $\mathscr{Z}$ is a particular zonotope. Indeed, introducing point vector

$$c = A^\star \mathrm{mid}([r]) + \mathrm{mid}([v]) \in \mathbb{R}^n, \tag{25}$$

Algorithm 3: `Intersection_Z_H`

input : c, R, η, d
output: $c_\cap, R_\cap$

$\sigma = R^\top \eta$;
$d' = d - \eta^\top c$;
$V_0 = N(\sigma^\top)$;
$c_s = \sigma d' / \|\sigma\|_2^2$;
$R_s = V_0 V_0^\top$;
$c_\cap = c + R c_s$;
$R_\cap = R R_s$;

and point matrix

$$R = \left[\begin{array}{cc} A^\star dr([r]) & dr([v]) \end{array} \right] \in \mathbb{R}^{n \times 2n} \tag{26}$$

where $dr(.)$ is a short notation for $\mathrm{diag}(\mathrm{rad}\,(.))$, we can write the flowpipe as a zonotope as follows

$$\mathscr{Z} = c \oplus R\mathbf{B}^{2n} \tag{27}$$

Let us assume that the guard sets are characterized by hyperplane $\mathscr{H}$, i.e. equation (7) simplifies as

$$\gamma_e(z(t)) = 0 \Leftrightarrow \eta^\top z(t) = d \tag{28}$$

then, the MSBP $\mathscr{Z}^\star = \mathscr{Z}_\cap$ that over-approximates $\mathscr{Z} \cap \mathscr{H}$ can directly be obtained using proposition 5.1 with $p = 2n$.

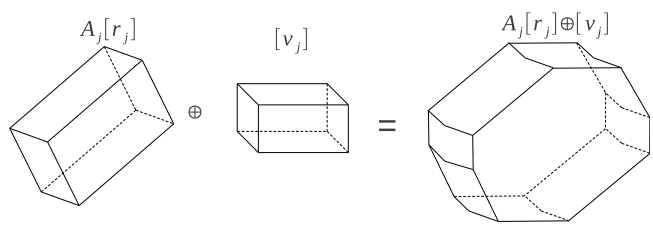


FIGURE 3. Minkowski sum of oriented box $A_j[r_j]$ and aligned box $[v_j]$

Example 1. *Consider the zonotope* $\mathtt{Z}_1 = c \oplus R\mathbf{B}^p$ *and the hyperplane* $\mathscr{H} = \{x | \eta^\top x = d\}$ *with*

$$c = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 2 & 3 & 0.5 & 0 \\ 3 & 2 & 1 & 0 & -0.5 \end{bmatrix}, \quad \eta = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \quad d = -0.5;$$

In Fig. 4, red line shows the obtained zonotope $\mathtt{Z}_\cap$ that bounds the intersection $\mathtt{Z}_1 \cap \mathscr{H}$

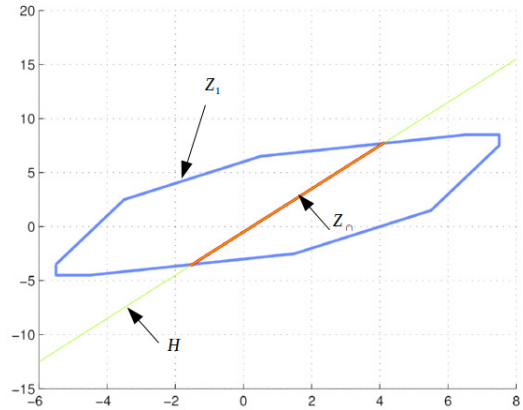


FIGURE 4. $Z_1 \cap \mathscr{H} \subseteq Z_\cap$

Algorithm 4: `GetGEN`

input : $[z], \hat{z}, [v], [r], A$
output: c, R
1 $c_{=} Amid([r]) + mid([v])$;
2 $R = [A.diag(rad([r])), diag(rad([v]))]$

5.4. Nonlinear hybrid reachability approach

Finally, we can combine the method for computing continuous transitions described in section 4 with the method for computing the intersection between the flowpipe and the guard sets described in this section to derive a complete method for nonlinear hybrid reachability computation. The combined methods are gathered in the algorithms described in **Algorithm** (4) and (5).

Remark 2. *Note that in Algorithm* 5 *(*`Hybrid-Transition`*) the inclusion test at line 20 checks whether the guard condition is satisfied for a state vector $z \in [\widetilde{z}]$, where $[\widetilde{z}]$ is a flowpipe enclosure computed over the tiny time interval $[\underline{t}^\star, \overline{t}^\star]$. A nice consequence is that the reachable set needs not to cross completely the guard condition over the latter time interval in order to activate the discrete jump using the reset function ρ.*

Remark 3. *Note also that when the guard condition is a sliding surface, small over-approximations introduced when computing enclosures $[\widetilde{z}]$ may introduce spurious guard crossing. However, in the general case, as illustrated in the examples below, our approach correctly handles discrete transition without introducing spurious guard crossing.*

Algorithm 5: `Hybrid-Transition`

input : $\mathscr{L}_j^F, t_{j+1}, \{\varphi_q^{l_{qr}}(), \widetilde{\varphi}_q(), \nu_q()\}_{q\in\mathscr{Q}}, \{\gamma_e(), \rho_e()\}_{e\in\mathscr{E}}, \varepsilon_T, \eta, d$
output: $\mathscr{L}_{j+1}^F, \mathscr{L}_{j+1}^R$

Initialization : initialize running frontier list $\mathscr{L} := \mathscr{L}_j^F$;
while $\mathscr{L} \neq 0$ **do**
  pick up $\mathscr{L}$ list element $(q, t_0, [\chi_0])$;
  /*Continuous transition */
    compute continuous expansion over $[t_0, t_{j+1}] \rightarrow [\widetilde{\mathbf{z}}]_j$;
    update reached set list $\mathscr{L}_{j+1}^R \leftarrow (q, t_0, t_{j+1}, [\widetilde{\mathbf{z}}_j])$;
    compute new solution at time $t_{j+1} \rightarrow [\chi_{j+1}]$;
    solve CSP to compute $[\chi'_{j+1}] := [\chi_{j+1}] \cap inv(q)$;
  if $[\chi_{j+1}]' \neq \emptyset$ **then**
    update frontier list $\mathscr{L}_{j+1}^F \leftarrow (q, t_{j+1}, [\chi_{j+1}]')$;
  end if
  /*Discrete transition */
  forall the *elements* $e \leftarrow \mathscr{E}$ **do**
    initialize running jump list $\mathscr{L}_e := \{(q, t_0, [\chi_0], t_{j+1})\}$;
    while $\mathscr{L}_e \neq 0$ **do**
      compute flow over $[t_0, t_1] \rightarrow [\widetilde{\mathbf{z}}]$;
      if $e = (q, q')$ *exists* **then**
        if $(\gamma_e([\widetilde{\mathbf{z}}]) \ni 0)$ **then**
          $\underline{t}^\star = t_0$; $\overline{t}^\star = t_1$;
          if $((\overline{t}^\star - \underline{t}^\star) \leq \varepsilon_T)$ **then**
            compute flow enclosure over $[\underline{t}^\star, \overline{t}^\star] \rightarrow [\widetilde{\chi}]^\star$;
            $[c, R] \leftarrow$ `GetGEN`$([\chi])$;
            $(c_\cap, R_\cap) =$ `Intersection_Z_H`(c, R, η, d) ;
            jump and update $\mathscr{L} \leftarrow (q', \underline{t}^\star, \rho_e(\mathscr{Z}_\cap))$;
          else
            compute solution set at $\underline{t}^\star \rightarrow [\chi]_1^\star$;
            compute solution set at $t_1 := (\underline{t}^\star + \overline{t}^\star)/2 \rightarrow [\chi]_2^\star$; update running jump list $\mathscr{L}_e \leftarrow (q, \underline{t}^\star, [\chi]_1^\star, t_1)$; update running jump list $\mathscr{L}_e \leftarrow (q, t_1, [\chi]_2^\star, \overline{t}^\star)$;
          end if
        end if
      end if
    end while
  end forall
end while

6. Numerical experiments

For the purpose of numerical experimentation, the above system-solving methods have been implemented in the `IBEX` C++ library [9] and we have used the standard contractor `HC4Revise` it includes. We have also used `Profil/Bias` C++ class library [31] for interval computation, `FABDAB++` package [7] for automatic differentiation and computing the Taylor coefficients, `AML++` [30]

and `Armadillo` [42] package for Linear algebra. All experiments were tested on a $i5-2430$M 2.4GHz CPU with 3.8GB RAM running Ubuntu Linux.

6.1. Example 1

Consider the hybrid transition for a hybrid dynamical system (Brusselator) with two modes $q = 1, 2$ and one jump transition $e = 1 \to 2$ given by :

$$\begin{cases} \text{flow}(1) : f_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1-(b_1+1)x_1 + a_1 x_1^2 x_2 \\ b_1 x_1 - a_1 x_1^2 x_2 \end{pmatrix} \\ \text{inv}(1) : \nu_1(x_1, x_2) = -2x_1 + x_2 + 2 \\ \text{flow}(2) : f_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1-(b_2+1)x_1 + a_2 x_1^2 x_2 \\ b_2 x_1 - a_2 x_1^2 x_2 \end{pmatrix} \\ \text{inv}(2) : \nu_2(x_1, x_2) = -\nu_1(x_1, x_2) \\ \text{guard}(1) : \gamma_1(x_1, x_2) = \nu_1(x_1, x_2) \\ \text{reset}(1) : \rho_1(x_1, x_2) = L \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} l_1 \\ l_2 \end{pmatrix} \end{cases} \tag{29}$$

with $L = \mathbb{I}_2$, $l_1 = l_2 = -0.5$, $a_1 = 1.5$, $a_2 = 3.5$, $b_1 = 1$, $b_2 = 3.5$ and $x_0 \in [2, 2.15] \times [0.1, 0.15]$. We use a constant integration time step $h = 0.05$, and time interval was bisected until a threshold $\varepsilon_T = 0.005$. When we compare the results obtained in (Fig.s 5), we see that the method using the intersection of a zonotope with a hyperplane exhibits smaller computation time (CPU time = 0.16s) than the one using a contactor (CPU time = 0.20s). Furthermore, when we focus on the over-approximation of the flowpipe/guard set intersection that is represented by black boxes, we also see that the two approaches yield similar results (see Fig.5(b)).

6.2. Example 2 : Vehicle Model

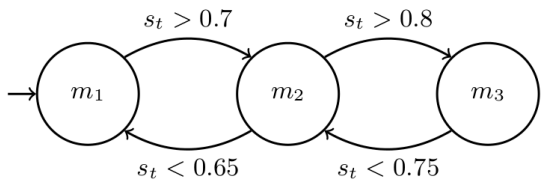


FIGURE 6. Hybrid automaton of the vehicle model

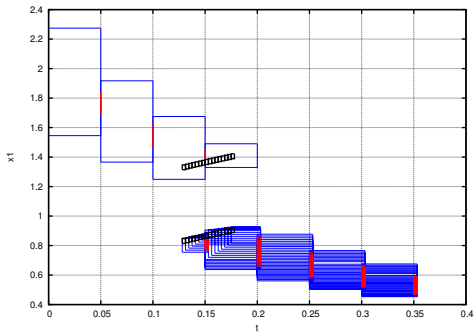
The dynamics of a non-holonomic vehicle [10] is given as follows :

$$\begin{array}{lll} \frac{dx}{dt} = vc_t; & \frac{dy}{dt} = vs_t; & \frac{dv}{dt} = u_1 \\ \frac{dc_t}{dt} = \sigma v^2 s_t; & \frac{ds_t}{dt} = -\sigma v^2 c_t; & \frac{d\sigma}{dt} = u_2 \end{array} \tag{30}$$

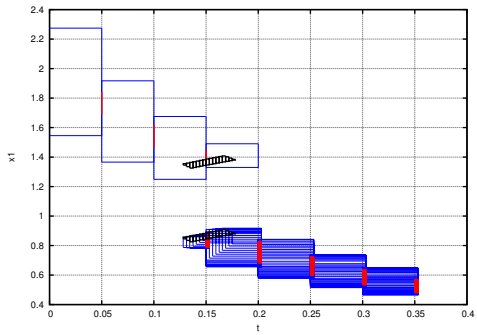
where u_1, u_2 are control inputs. We consider the case of a vehicle with three control modes m_1, m_2, m_3. The control inputs are given by $(u_1, u_2) = (-0.05, -0.1)$ for mode m_1, $(0,0)$ for m_2 and $(0.05, 0.1)$ for m_3. The transitions between modes are shown in Fig. 6. We start our simulation from m_1 with the initial variable values :

$$\begin{array}{ccccc} x \in [1, 1.2] & & y \in [1, 1.2] & & v \in [0.8, 0.81] \\ s_t \in [0.6, 0.61] & & c_t \in [0.7, 0.71] & & \sigma = [0, 0.01] \end{array}$$

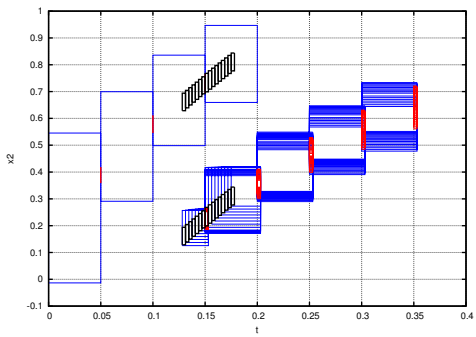
We took for this simulation a constant integration time step $h = 1$, the time interval was bisected until a threshold $\varepsilon_T = 0.05$. All results are plotted in Fig.7(a) and Fig.7(c) when using the zonotope-based method, and in Fig.7(b) and Fig.7(d) for the CSP-based one.



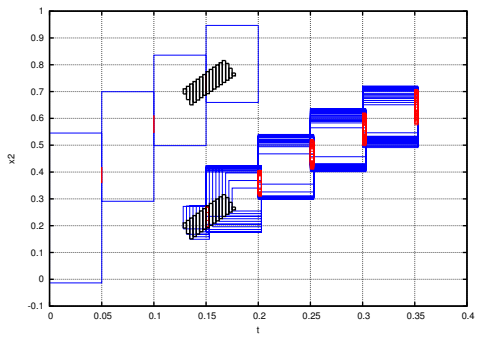
(a) Zonotope-based method. Time history of x_1.



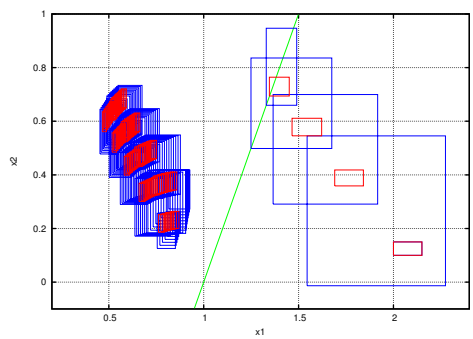
(b) CSP-based method. Time history of x_1.



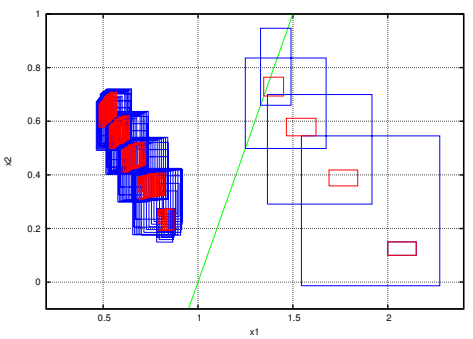
(c) Zonotope-based method. Time history of x_2.



(d) CSP-based method. Time history of x_2.



(e) Zonotope-based method. Projections onto $x1 \times x2$.

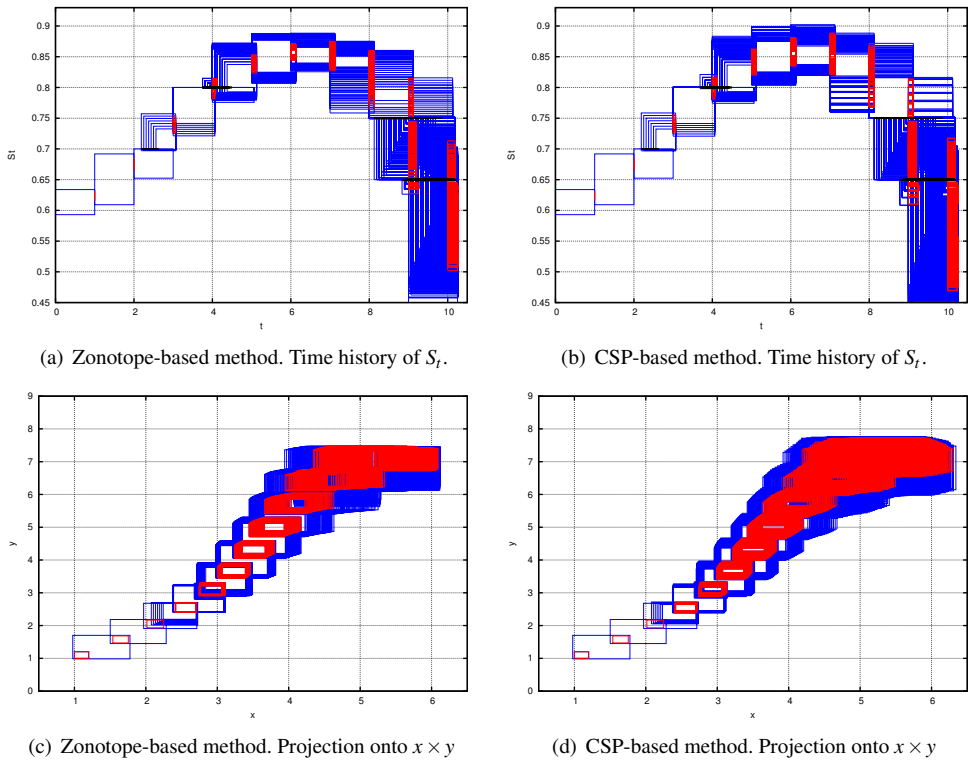


(f) CSP-based method. Projections onto $x1 \times x2$.

FIGURE 5. Reachable set computed for (29). Zonotope-based method, CPU times=0.16s. CSP-based method, CPU times=0.20s.

These figures show that the zonotope-based method yields tighter reachable set obtained in a smaller computation time (CPU time = 36s) than the CSP-based method (CPU time = 46s).

Fig.7(a)-7(c) shows that our algorithm can perform numerical model simulations with large integration time step $h = 1$, hence illustrating the benefit of the proposed method. Note also that our method can produce hybrid reachable sets that correctly cover two guard sets conditions ($s_t < 0.75$, $s_t < 0.65$) simultaneously.



(a) Zonotope-based method. Time history of S_t.

(b) CSP-based method. Time history of S_t.

(c) Zonotope-based method. Projection onto $x \times y$

(d) CSP-based method. Projection onto $x \times y$

FIGURE 7. Reachable set of (30). Zonotope-based method, CPU time = 36s. CSP-based method, CPU time = 46s.

7. Concluding remarks

We have shown that nonlinear hybrid reachability computation can be achieved efficiently by combining interval Taylor methods, interval constraint propagation and zonotope geometrical tools. Evaluating our approach on a 6-dim non-holonomic vehicle benchmark we emphasized its nice performance. Future work will focus on how to extend the zonotope-based method to nonlinear guards. Furthermore, even though bisection is performed on a single direction only, our method generates in practice a large number of boxes to cover the hybrid reachable set, for both variants. The natural idea to reduce this number is to merge these boxes. Ways to achieve this box merging are under study.

References

[1] M. Althoff and B. H. Krogh. Zonotope bundles for the efficient computation of reachable sets. In *CDC-ECE*, pages 6814–6821, 2011.

[2] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *HSCC*, pages 45–54, 2012.

[3] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *CDC*, pages 4042–4048, 2008.

[4] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[5] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *THEORETICAL COMPUTER SCIENCE*, 138:3–34, 1995.

[6] C. Bendtsen and O. Stauning. FADBAD, a flexible C++ package for automatic differentiation. Technical Report IMM–REP–1996–17, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, aug 1996.

[7] L. G. Birta, T. I. Oren, and D. L. Kettenis. A robust procedure for discontinuity handling in continuous system simulation. *Trans. Soc. Comput. Simul. Int.*, 2(3):189–205, Sept. 1985.

[8] G. Chabert. IBEX: Interval Based EXplorer, available at `http://www.ibex-lib.org/`, 2007.

[9] X. Chen, E. Abraham, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *33rd IEEE Real-Time Systems Symposium (RTSS 12)*, pages 183–192, 2012.

[10] J. C. Cleary. Logical arithmetic. *Future Computing Systems*, 2:125–149, 1987.

[11] H. Collavizza, F. Delobel, and M. Rueher. Comparing partial consistencies. *Reliable computing*, 5:213–228, 1999.

[12] C. Combastel, Q. Zhang, and A. Lalami. Fault diagnosis based on the enclosure of parameters estimated with an adaptive observer. In *17th IFAC World Congress*, Seoul, Korea, July 6-11 2008.

[13] G. F. Corliss and R. Rihm. Validating an a priori enclosure using high-order taylor series. In *In Scientific Computing, Computer Arithmetic, and Validated Numerics*, pages 228–238. Akademie Verlag, 1996.

[14] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.

[15] A. Eggers, M. Fränzle, and C. Herde. SAT Modulo ODE: A direct SAT approach to hybrid systems. In *Cha et al. (Eds.): ATVA 2008, LNCS 5311*, pages 171–185, 2008.

[16] A. Eggers, N. Ramdani, N. S. Nedialko, and M. Fränzle. Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. *Software & Systems Modeling*, pages 1–28, 2012.

[17] J. M. Esposito and V. Kumar. A state event detection algorithm for numerically simulating hybrid systems with model singularities. *ACM Trans. Model. Comput. Simul.*, 17(1), 2007.

[18] J. M. Esposito, V. Kumar, and G. J. Pappas. Accurate event detection for simulating hybrid systems. In *HSCC*, pages 204–217, 2001.

[19] G. Frehse and R. Ray. Flowpipe-guard intersection for reachability computations with support functions. In *IFAC Conf. Analysis and Design of Hybrid Systems (ADHS)*, pages 94–101, 2012.

[20] A. Girard. Detection of event occurrence in piecewise linear hybrid systems. Dec. 2002.

[21] A. Girard. Reachability of uncertain linear systems using zonotopes. In *HSCC, vol 3414 in LNCS*, pages 291–305, 2005.

[22] A. Girard and C. L. Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *HSCC*, pages 215–228, 2008.

[23] H. Guéguen and J. Zaytoon. On the formal verification of hybrid systems. *Control Engineering Practice*, 12:1253–1267, 2004.

[24] C. Guernic and A. Girard. Reachability analysis of hybrid systems using support functions. In *Proceedings of the 21st International Conference on Computer Aided Verification*, CAV '09, pages 540–554, Berlin, Heidelberg, 2009. Springer-Verlag.

[25] C. L. Guernic and A. Girard. Reachability analysis of hybrid systems using support functions. In *CAV*, pages 540–554, 2009.

[26] E. Hansen and G. Walster. *Global optimization using interval analysis*. Marcel Dekker, 2nd edition, 2004.

[27] D. Ishii, K. Ueda, and H. Hosobe. An interval-based sat modulo ode solver for model checking nonlinear hybrid systems. *International Journal on Software Tools for Technology Transfer*, 13:449–461, 2011.

[28] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis: with examples in parameter and state estimation, robust control and robotics*. Springer-Verlag, London, 2001.

[29] G. Klima. AML++: Another Matrix Library in c++, available at `http://amlpp.sourceforge.net/`, 2008-2010.

[30] O. Knüppel. PROFIL/BIAS a fast interval library. *Computing*, 53(3-4):277–287, 1994.

[31] A. Lalami and C. Combastel. A state bounding algorithm for linear systems with bounded input and bounded slew-rate. *European Control Conference*, 2007.

[32] R. J. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. In E. W. Kaucher, U. W. Kulisch, and C. Ullrich, editors, *Computer Arithmetic: Scientific Computation and Programming Languages*, pages 255–286. Wiley-Teubner, Stuttgart, 1987.

[33] M. Maïga, N. Ramdani, and L. Travé-Massuyès. A fast method for solving guard set intersection in nonlinear hybrid reachability. In *Proc. of 52nd IEEE Conference on Decision and Control, CDC 2013.*, pages 508–513, December 2013.

[34] R. E. Moore. *Interval Analysis*. Englewood Cliffs, prentice-hall., 1996.

[35] N. Nedialkov, K. Jackson, and G. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21 – 68, 1999.

[36] N. S. Nedialkov and K. R. Jackson. An effective high-order interval method for validating existence and uniqueness of the solution of an ivp for an ode. *Computing*, 17:pages., 2001.

[37] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. In A. Iserles, editor, *Acta Numerica*, chapter 4. Cambridge University Press, 2004.

[38] T. Park and P. I. Barton. State event location in differential-algebraic models. *ACM Trans. Model. Comput. Simul.*, 6(2):137–165, 1996.

[39] N. Ramdani, N. Meslem, and Y. Candau. A hybrid bounding method for computing an over-approximation for the reachable set of uncertain nonlinear systems. *IEEE Trans. Automat. Contr.*, 54(10):2352–2364, 2009.

[40] N. Ramdani and N. S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. *Nonlinear Analysis: Hybrid Systems*, 5(2):149 – 162, 2011.

[41] C. Sanderson. Armadillo: C++ linear algebra library, available at `http://arma.sourceforge.net/`, 2013.

[42] L. F. Shampine, I. Gladwell, and R. W. Brankin. Reliable solution of special event location problems for ODEs. *ACM transactions on Mathematical Software*, 17:11–25, 1987.

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

[44] D. L. Waltz. *Generating semantic descriptions from drawings of scenes with shadows*, pages 19–91. McGraw-Hill, New York, 1975.

Moussa Maïga
Univ. Orléans, INSA-CVL, PRISME, EA 4229, F45072, Orléans, and CNRS, LAAS, Toulouse, France
e-mail: `mmaiga@laas.fr`

Nacim Ramdani
Univ. Orléans, INSA-CVL, PRISME, EA 4229, F45072, Orléans, France
e-mail: `nacim.ramdani@univ-orleans.fr`

Louise Travé-Massuyès
CNRS, LAAS, Toulouse, France
e-mail: louise@laas.fr

Christophe Combastel
ENSEA, ECS-Lab, Paris, France
e-mail: christophe.combastel@ensea.fr