Systematic Simulation using Sensitivity Analysis

Alexandre Donzé and Oded Maler

VERIMAG
2, Avenue de Vignate
38610 Gières, France
Alexandre.Donze@imag.fr

Abstract. In this paper we propose a new technique for verification by simulation of continuous and hybrid dynamical systems with uncertain initial conditions. We provide an algorithmic methodology that can, in most cases, verify that the system avoids a set of bad states by conducting a *finite* number of simulation runs starting from a finite subset of the set of possible initial conditions. The novelty of our approach consists in the use of *sensitivity analysis*, developed and implemented in the context of numerical integration, to efficiently characterize the coverage of sampling trajectories.

1 Introduction

Numerical simulation is a commonly-used method for predicting or validating the behavior of complex dynamical systems. It is often the case that due to incomplete knowledge of the initial conditions or the presence of external disturbances, the system in question may have an infinite and non-countable number of trajectories, only a finite subset of which can be covered by simulation. Two major directions for attacking this coverage problem have been reported in the literature. The first approach, see e.g. [ACH$^+$95,DM98,CK98,ADG03,Gir05] consists of an adaptation of discrete verification techniques to the continuous context via reachability computation, namely computing by geometric means an over-approximation of the set of states reached by all trajectories. The other complementary approach attempts to find conditions under which a finite number of well-chosen trajectories will suffice to prove correctness and cover in some sense all the trajectories of the system [KKMS03,BF04,BCLM05,BCLM06,GP06]. This paper is concerned with the second approach.

The main contribution of the paper is an algorithm whose input consists of an arbitrary dynamical system, an initial set $\mathcal{X}_0$, a set of "bad" states $\mathcal{F}$, a time interval $[0,T]$ and some $\delta > 0$. The algorithm picks a point in $\mathcal{X}_0$ and simulates the trajectory of the system. If a bad state occurs in the trajectory the algorithm stops and declares the system "unsafe"; otherwise, a systematic refinement operator is used to extend the sampling of the initial points from which new trajectories are numerically simulated and so on. The algorithm is guaranteed to terminate in a finite number of steps, either by finding a bad trajectory and declaring the system unsafe, by declaring the system "safe" when the sampled trajectories provably cover the reachable set, or returning an "uncertain" answer when sampled trajectories are less than δ-close to $\mathcal{F}$.

The novelty of this paper w.r.t. similar works is the notion of coverage used, which is based on the concept of an *expansion function* which characterizes how neighboring trajectories are getting closer to or further from one another as time goes by. We show that this notion can be effectively approximated[1] using the concept of *sensitivity function* implemented in standard numerical integrators. The rest of the paper is organized as follows. In Section 2 we define samples, their dispersion and expansion functions and use them to introduce an abstract algorithm for safety verification. The algorithm is then concretized and implemented using a numerical solver, a grid-based sample refinement scheme (Section 3) and sensitivity to approximate expansion (Section 4). In Section 5 we demonstrate the behavior of our implementation of the algorithm on a linear time-varying system and on two nonlinear analog circuits, the tunnel diode oscillator and the voltage controlled oscillator. The extension of sensitivity analysis to hybrid systems is the topic of Section 6, which is followed by discussions of future steps in simulation-based verification.

2 Verification by Simulation

We consider a dynamical system of the form

$$\dot{\mathbf{x}} = f(t, \mathbf{x}),\ \mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{X}_0, \tag{1}$$

where $\mathbf{x}$ is a vector in $\mathbb{R}^n$, $\mathcal{X}_0$ is compact and f is assumed of class C^1. The problem is known to have a unique solution noted $\xi_{\mathbf{x}_0}(t)$. We say that the system is *safe* if all trajectories starting from any $\mathbf{x}_0 \in \mathcal{X}_0$ do not intersect a set $\mathcal{F}$ of bad states. We consider a bounded time horizon $[0, T]$. Let $\text{Reach}_{\leq t}(\mathcal{X}_0)$ (resp. $\text{Reach}_{=t}(\mathcal{X}_0)$) be the set reachable from $\mathcal{X}_0$ in less than (resp. exactly) t units of time. A usual way to prove that the system is safe is to prove emptiness of the intersection of the reachable set $\text{Reach}_{\leq T}(\mathcal{X}_0)$ with the set $\mathcal{F}$. In this section, we introduce the concept of *expansion function* which allows to characterize how a finite set of trajectories covers the reachable set and use such a set of trajectories trying to refute intersection with the bad set $\mathcal{F}$.

2.1 Preliminaries

We use a metric d and extend it to distance from points to set using $d(\mathbf{x}, \mathcal{X}) = \inf_{\mathbf{y}\in\mathcal{X}}\big(d(\mathbf{x}, \mathbf{y})\big)$ When used, the notation $\|\cdot\|$ denotes the infinity norm. It extends to matrix with the usual definition: $\|\mathbf{A}\| = \sup_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|$. A ball $\mathcal{B}_\delta(\mathbf{x})$ is the set of points $\mathbf{x}'$ satisfying $d(\mathbf{x}, \mathbf{x}') \leq \delta$. Given a set $\mathcal{X}$, a cover for $\mathcal{X}$ is a

[1] For linear time varying systems, as we show, these two notions coincide.

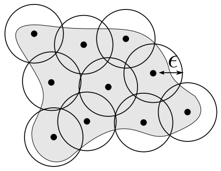
set of sets $\{\mathcal{X}_1, \dots \mathcal{X}_k\}$ such that $\mathcal{X} \subset \bigcup_{i=1}^{k} \mathcal{X}_i$. A ball cover is a cover consisting of balls. We extend the notation $\mathcal{B}_\delta$ to sets and trajectories as follows

$$\mathcal{B}_\delta(\mathcal{S}) = \bigcup_{\mathbf{x} \in \mathcal{S}} \mathcal{B}_\delta(\mathbf{x}) \quad \text{and} \quad \mathcal{B}_\delta(\xi_{\mathbf{x}}) = \bigcup_{t \in [0,T]} \mathcal{B}_{\delta(t)}(\xi_{\mathbf{x}}(t))$$

A *sampling* of $\mathcal{X}$ is a set $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ of points in $\mathcal{X}$. The intuitive notion of the "coverage" of $\mathcal{X}$ by $\mathcal{S}$ is formalized by

Definition 1 (Dispersion). *The dispersion $\alpha_{\mathcal{X}}(\mathcal{S})$ is the smallest radius ϵ such that the union of all ϵ radius closed balls with their center in $\mathcal{S}$ covers $\mathcal{X}$.*

$$\alpha_{\mathcal{X}}(\mathcal{S}) = \min_{\epsilon > 0} \{\epsilon \mid \mathcal{X} \subset \mathcal{B}_\epsilon(\mathcal{S})\} \tag{2}$$



We now define the process of *refining* a sampling, which simply consists in finding a new sampling with a strictly smaller dispersion.

Definition 2 (Refinement). *Let $\mathcal{S}$ and $\mathcal{S}'$ be samplings of $\mathcal{X}$. We say that $\mathcal{S}'$ refines $\mathcal{S}$ if and only $\alpha_{\mathcal{X}}(\mathcal{S}') < \alpha_{\mathcal{X}}(\mathcal{S})$.*

A refining sampling can be constructed from the set to refine (e.g. by adding sufficiently many points) or be found independently. In both cases, we can assume that it is obtained through a *refinement operator* which we define next.

Definition 3 (Refinement operators). *A refinement operator $\rho : 2^{\mathcal{X}} \mapsto 2^{\mathcal{X}}$ maps a sampling $\mathcal{S}$ to another sampling $\mathcal{S}' = \rho_{\mathcal{X}}(\mathcal{S})$ such that $\mathcal{S}$ refines $\mathcal{S}'$. A refinement operator is* complete *if $\forall \mathcal{S}$,*

$$\lim_{k \to \infty} \alpha_{\mathcal{X}}\big(\rho_{\mathcal{X}}^{(k)}(\mathcal{S})\big) = 0$$

where $\rho_{\mathcal{X}}^{(k)}(\mathcal{S})$ is the result of k application of $\rho_{\mathcal{X}}$ to $\mathcal{S}$.

In other terms, a refinement operator is complete if a sampling of $\mathcal{X}$ which has been infinitely refined is *dense* in $\mathcal{X}$. Until we define one in section 3, we assume the existence of a complete refinement operator ρ.

2.2 Expansion Function

The intuitive idea is to draw "tubes" around trajectories so that the union of these tubes will provide an over-approximation of the reachable set. The expansion function then simply maps time t to the radius of the tube at t, given an initial state $\mathbf{x}_0$ and an initial radius ϵ.

Definition 4 (Expansion function). *Given $\mathbf{x}_0 \in \mathcal{X}_0$, and $\epsilon > 0$, the* expansion function *of $\xi_{\mathbf{x}_0}$, denoted by $\mathcal{E}_{\mathbf{x}_0,\epsilon} : \mathbb{R}^+ \mapsto \mathbb{R}^+$ maps t to the smallest non-negative number δ such that all trajectories with initial state in $\mathcal{B}_\epsilon(\mathbf{x}_0)$ reach a point in $\mathcal{B}_\delta(\xi_{\mathbf{x}_0}(t))$ at time t:*

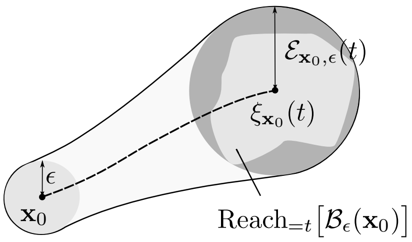
$$\mathcal{E}_{\mathbf{x}_0,\epsilon}(t) = \sup_{d(\mathbf{x}_0,\mathbf{x}) \leq \epsilon} d\big(\xi_{\mathbf{x}_0}(t), \xi_{\mathbf{x}}(t)\big) \tag{3}$$

Clearly, a first property of the expansion functions is that it approaches 0 as ϵ tends toward 0:

$$\forall t > 0, \lim_{\epsilon \longrightarrow 0} \mathcal{E}_{\mathbf{x},\epsilon}(t) = 0 \tag{4}$$

This results directly from the continuity of $\xi_{\mathbf{x}}(t)$ w.r.t. $\mathbf{x}$.

The expansion function value $\mathcal{E}_{\mathbf{x}_0,\epsilon}(t)$ gives the radius of the ball which over-approximate tightly the reachable set from the ball $\mathcal{B}_\epsilon(\mathbf{x}_0)$ at time t. Obviously, if we take several such balls so that the initial set $\mathcal{X}_0$ is covered, we obtain a corresponding cover of $\text{Reach}_{=t}(\mathcal{X}_0)$. This is stated in the following


$\mathcal{E}_{\mathbf{x}_0,\epsilon}(t)$
$\xi_{\mathbf{x}_0}(t)$
ϵ
$\mathbf{x}_0$
$\text{Reach}_{=t}\big[\mathcal{B}_\epsilon(\mathbf{x}_0)\big]$

Proposition 1. *Let $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_k\}$ be a sampling of $\mathcal{X}_0$ such that $\bigcup_{i=1}^k \mathcal{B}_{\epsilon_i}(\mathbf{x}_i)$ is a ball cover of $\mathcal{X}_0$ for some $\{\epsilon_1, \dots, \epsilon_k\}$. Let $t > 0$ and for each $1 \leq i \leq k$, let $\delta_i = \mathcal{E}_{\mathbf{x}_i,\epsilon_i}(t)$. Then $\bigcup_{i=1}^k \mathcal{B}_{\delta_i}(\xi_{\mathbf{x}_i}(t))$ is a ball cover of Reach$_{=t}(\mathcal{X}_0)$.*

Proof. By definition, the ball cover of $\mathcal{X}_0$ contains $\mathcal{X}_0$, and each $\mathcal{B}_{\delta_i}(\xi_{\mathbf{x}_i}(t))$ contains $\text{Reach}_{=t}(\mathcal{B}_{\epsilon_i}(\mathbf{x}_i))$, and the rest follows from the commutativity of the dynamics with set union and containment. □

In particular, if $\mathcal{S}$ is a sampling of $\mathcal{X}_0$ with dispersion ϵ then we are in the case where $\epsilon_i = \epsilon$ for all $1 < i < k$ and since the result is true for all $t \in [0, T]$, we have the following

Corollary 1. *Let $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\}$ be a sampling of $\mathcal{X}_0$ with dispersion $\alpha_{\mathcal{X}_0}(\mathcal{S}) = \epsilon$. Let $\delta > 0$ be an upper bound for $\mathcal{E}_{\mathbf{x}_i,\epsilon}(t)$ for all $1 < i < k$ and $t \in [0, T]$, then the following inclusions hold*

$$\text{Reach}_{[0,T]}(\mathcal{X}_0) \subseteq \bigcup_{\mathbf{x}\in\mathcal{S}} \mathcal{B}_{\mathcal{E}_{\mathbf{x},\epsilon}}(\xi_{\mathbf{x}}) \subseteq \bigcup_{\mathbf{x}\in\mathcal{S}} \mathcal{B}_\delta(\xi_{\mathbf{x}}) \subseteq \mathcal{B}_\delta\big(\text{Reach}_{[0,T]}(\mathcal{X}_0)\big) \tag{5}$$

Proof. The first inclusion is a direct application of the proposition. The second results from the fact that δ is an upper-bound and the third inclusion is due to the fact that $\forall (\mathbf{x}_i, t) \in \mathcal{S} \times [0, T],\ \xi_{\mathbf{x}_i}(t) \in \text{Reach}_{[0,T]}(\mathcal{X}_0)$. □

In other terms, if we bloat the sampling trajectories starting from $\mathcal{S}$ with a radius δ, which is an upper bound for expansion functions of these trajectories, then we get an over-approximation of the reachable set which is between the exact reachable set and the reachable set bloated with δ. Because of (4), it is clear that δ, and then the over-approximation error, decreases when ϵ gets smaller.
The second corollary of proposition 1 underlies our verification strategy.

Corollary 2. *Let $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ be a sampling of $\mathcal{X}$ such that $\bigcup_{i=1}^{k} \mathcal{B}_{\epsilon_i}(\mathbf{x}_i)$ is a ball cover of $\mathcal{X}_0$. For $t \in [0, T]$ and $1 \leq i \leq k$, let $\delta_i(t) = \mathcal{E}_{\mathbf{x}_i, \epsilon_i}(t)$. If for all $t \in [0, T]$,*

$$\mathcal{B}_{\delta_i(t)}(\xi_{\mathbf{x}_i}(t)) \cap \mathcal{F} = \emptyset,$$

then for all trajectory $\xi_{\mathbf{x}}$ starting from $\mathbf{x} \in \mathcal{X}_0$, the intersection of $\xi_{\mathbf{x}}$ and the bad set $\mathcal{F}$ is empty and thus the system is safe.

2.3 A Verification Algorithm

In theory, then, from previous proposition and corollaries, a unique trajectory could be sufficient to verify the system: take one point $\mathbf{x}_0$, find ϵ such that $\mathcal{X}_0 \subset \mathcal{B}_{\epsilon}(\mathbf{x}_0)$, then check for all $t \in [0, T]$ that $\mathcal{B}_{\delta(t)}(\xi_{\mathbf{x}_i}(t)) \cap \mathcal{F} = \emptyset$, where $\delta(t) = \mathcal{E}_{\mathbf{x}_0, \epsilon}(t)$. If this is the case, then the system is safe. Obviously, the opposite case does not mean that the system is unsafe since $\mathcal{B}_{\delta(t)}(\xi_{\mathbf{x}_i}(t))$ is actually an over-approximation of $\text{Reach}_{=t}(\mathcal{X}_0)$, rather it indicates that the distance between the reachable set and $\mathcal{F}$ is less than $\delta(t)$. If this indication is not sufficient, then more trajectories have to be simulated until a sufficiently dense sampling of $\mathcal{X}_0$ is found. This is what Algorithm 1 does.

Algorithm 1 Safety Verification. The algorithm takes $\delta > 0$ as input.

```
$\mathcal{U} \leftarrow \emptyset$, $\mathcal{S}_0 \leftarrow \{\mathbf{x}_0\}$ with $\mathbf{x}_0 \in \mathcal{X}_0$, $k \leftarrow 0$
loop
    /* For $\mathcal{S}_k$, check each trajectory and compute an upper bound $\delta_k$ for $\mathcal{E}$ */
    $\epsilon_k \leftarrow \alpha_{\mathcal{X}_0}(\mathcal{S}_k, \mathcal{X}_0)$, $\delta_k \leftarrow 0$
    for all $\mathbf{x} \in \mathcal{S}_k$ do
        if $\xi_{\mathbf{x}} \cap \mathcal{F} \neq \emptyset$ then
            return (unsafe, $\{\mathbf{x}\}$)
        else if $\mathcal{B}_{\mathcal{E}_{\mathbf{x},\epsilon_k}}(\xi_x) \cap \mathcal{F} \neq \emptyset$ then
            $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathbf{x}\}$
        end if
        $\delta_k \leftarrow \max\left(\sup_{t \in [0,T]} (\mathcal{E}_{\mathbf{x},\epsilon_k}(t)), \delta_k\right)$
    end for
    /* Stop either if no uncertain trajectory was found (safe) or if the upper
    bound is smaller than precision $\delta$. Else refine the sampling and loop */
    if $\mathcal{U} = \emptyset$ then
        return safe
    else if $\delta_k < \delta$ then
        return (uncertain, $\mathcal{U}$)
    else
        $\mathcal{S}_{k+1} \leftarrow \rho_{\mathcal{X}_0}(\mathcal{S}_k)$, $\mathcal{U} \leftarrow \emptyset$, $k \leftarrow k + 1$ /* Refine the sample */
    end if
end loop
```

Theorem 1. *Under assumptions mentioned above, algorithm 1 terminates and its output satisfies:*

- *it is* `safe` *only if the system is safe.*
- *it is* (`unsafe`, $\{\mathbf{x}\}$) only if the system is unsafe and $\{\mathbf{x}\}$ is a counter-example, i.e.: ξ_x intersects $\mathcal{F}$.
- it is ($\texttt{uncertain}, \mathcal{U}$) only if all the points in $\mathcal{U}$ induce uncertain trajectories: $\forall\ \mathbf{x} \in\ \mathcal{U},\ d(\xi_{\mathbf{x}}, \mathcal{F}) \leq \delta$.

Proof. For `unsafe`, the result is obvious from the algorithm. For `safe` and `uncertain`, the algorithm terminates because ρ is complete, $\lim\limits_{k\to 0} \epsilon_k = 0$ and $\lim\limits_{\epsilon_k \to 0} \delta_k = 0$. Consequently for some k, $\delta_k < \delta$. Now, if $\mathcal{U}$ was found empty, at or before iteration k, this means that corollary 2 applies which proves that the system is safe, while if $\mathcal{U}$ is still not empty at iteration k i.eif the algorithm has returned ($\texttt{unsafe}, \mathcal{U}$), then $\mathcal{U}$ contains states $\mathbf{x}$ for which

$$d(\xi_{\mathbf{x}}, \mathcal{F}) \leq \sup_{t\in[0,T]} (\mathcal{E}_{\mathbf{x},\epsilon_k}(t)) \ \leq\ \delta_k \ \leq\ \delta.$$

Note that moreover, the inclusions (5) are true for $\mathcal{S}_k$. □

The algorithm requires some $\delta > 0$ as input to guarantee termination. In fact, the problematic case is when the distance between the reachable set and the bad set is exactly 0. In this case, there is no way to get an answer other than `uncertain`. On the other hand, we can state the following theorem:

Theorem 2. *If $d\big(Reach_{\leq T}(\mathcal{X}_0), \mathcal{F}\big) > 0$, then there exist a $\delta > 0$ for which algorithm 1 returns* `safe`*.*

Proof. This is true for any $\delta < d\big(\mathrm{Reach}_{\leq T}(\mathcal{X}_0), \mathcal{F}\big)$. Indeed, since for some k, the inclusions (5) in corollary 1 are true for $\mathcal{S}_k$ then

$$\mathcal{B}_\delta\big(\mathrm{Reach}_{[0,T]}(\mathcal{X}_0)\big) \cup \mathcal{F} = \emptyset \Rightarrow \mathcal{B}_\delta\Big(\bigcup_{\mathbf{x}\in\mathcal{S}} \xi_{\mathbf{x}}\Big) \cup \mathcal{F} = \emptyset$$

so $\mathcal{U}$ is empty at the end of the **for** loop and the algorithm will return `safe`. □

3 An Efficient Sampling Strategy

In this section, we focus on the sampling strategy, that is, on the sample refinement operator.

3.1 Local Refinement

First, we can remark that when Algorithm 1 ends with the `uncertain` answer, one choice is to try a smaller δ. In that case, it is not necessary to restart the algorithm from scratch; the set $\mathcal{U}$ can be used. Indeed, it is clear that any trajectory starting from the ball $\mathcal{B}_{\epsilon_k}(\mathbf{x})$, where $\mathbf{x} \in \mathcal{S}_k$ has not been inserted into the uncertain set $\mathcal{U}$, is safe. Thus the set $B_{\epsilon_k}\big(\mathcal{S}_k \setminus \mathcal{U}\big)$ is safe and it is enough to verify the set $\mathcal{B}_{\epsilon_k}\big(\mathcal{U}\big)$. In fact, this observation is also relevant at each iteration of the **for** loop inside Algorithm 1. Instead of refining *globally* $\mathcal{S}_k$ by the instruction $\mathcal{S}_{k+1} \leftarrow \rho_{\mathcal{X}_0}(\mathcal{S}_k)$, $\mathcal{S}_k$ could be refined *locally* only around uncertain states. This can be done simply by replacing $\mathcal{X}_0$ with $B_{\epsilon_k}(\mathcal{U})$ and refine this new initial set. Line 19 thus becomes:

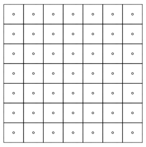
$\mathcal{X}_0 \leftarrow \mathcal{B}_{\epsilon_k}(\mathcal{U})$
$\mathcal{S}_{k+1} \leftarrow \rho_{\mathcal{X}_0}(\mathcal{U})$, $\mathcal{U} \leftarrow \emptyset$, $k \leftarrow k+1$

Now we proceed to describe the particular sampling method that we use in the implementation of the algorithm.

3.2 Hierarchical Grid Sampling

We saw that algorithm 1 terminates as soon as δ_k gets sufficiently small, which requires also the dispersion ϵ_k to be sufficiently small. Then we need that the convergence of sequence $(\epsilon_k)_{k\in\mathbb{N}}$ towards 0 to be as fast as possible. This requires that for a given number of points, our sampling strategy tries to minimize its dispersion. In this section, we assume that with an appropriate change in variables, sampling $\mathcal{X}_0$ is equivalent to sampling the unit hypercube $[0,1]^n$. In case we use the L_∞ metrics, i.e. $d(\mathbf{x},\mathbf{y}) = \max_i(|x_i - yi|)$, the solution of the problem of minimizing dispersion of N points in $[0,1]^n$ is known (see e.g. [LaV06]): the minimum possible dispersion is $\frac{1}{2\lfloor N^{1/n}\rfloor}$ and is obtained by placing the points at the center of smaller hypercubes of size $\frac{1}{\lfloor N^{1/n}\rfloor}$, partitioning the unit hypercube.

Note that there are $(\lfloor N^{1/n}\rfloor)^n$ such hypercubes, which may be less than N. In this case, the remaining points can placed anywhere without affecting the dispersion. Obtained grids are referred to as *Sukharev grids*. The picture on the right gives an example of such a grid for $n = 2$ and $N = 49$.



Sukharev grids have optimal dispersion but for a fixed number of points while in our verification algorithm, we do not know in advance how many points in the initial set we will have to use. In fact, we need to implement the refinement operator to be used throughout Algorithm 1, starting from the singleton sampling set $\mathcal{S}_0$. An elegant way to do it is to superpose *hierarchically* Sukharev grids, the refinement process being defined simply in a recursive fashion. Let $\mathcal{X}$ be a hypercube of size $\frac{1}{2^l}$ (we say that such a cube is part of the grid of resolution l) and $\mathcal{S}$ be a sampling of $\mathcal{X}$, then:

- if $\mathcal{S} = \emptyset$ then $\rho_{\mathcal{X}}(\mathcal{S}) = \{\mathbf{x}\}$, where $\mathbf{x}$ is the center of the hypercube $\mathcal{X}$;
- if $\mathcal{S} \neq \emptyset$ then $\rho_{\mathcal{X}}(\mathcal{S}) = \mathcal{S} \cup \bigcup_{i=1}^{2^n} \rho_{\mathcal{X}_i}(\mathcal{S}_i)$ where the sets $\mathcal{X}_i$ are the 2^n hypercubes of size $\frac{1}{2^{l+1}}$ partitioning $\mathcal{X}$ and the sets $\mathcal{S}_i$ contain the points of $\mathcal{S}$ that are inside $\mathcal{X}_i$.

On figure 1, we show the effect of three iterations in dimension 2 and 3 along with an example of successive local refinements. From this definition, it is clear that the operator ρ is a refinement operator and that it is complete since we have:

$$\alpha_{\mathcal{X}}(\rho_{\mathcal{X}}(\mathcal{S})) = \frac{1}{2}\alpha_{\mathcal{X}}(\mathcal{S}).$$

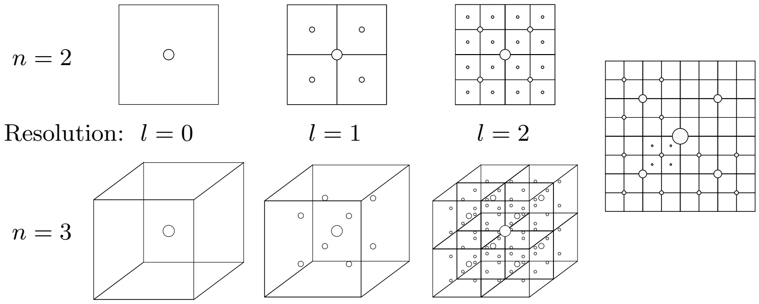


Fig. 1. Refinements for $n = 2$ and $n = 3$ dimensions for resolutions from $l = 0$ to $l = 2$. On the right, local refinements until resolution 3.

In [LYL04], a simple procedure is described for choosing the order in which the partitioning cubes are processed so that the mutual distance between two consecutive cubes is maximized. This is an interesting feature from the point of view of fast falsification since it means that every two consecutive simulation runs will be far from each other and that for any $k \in \mathbb{N}$, the initial states of the first k trajectories will constitute a good coverage of the initial set $\mathcal{X}_0$.

4 Implementation using Sensitivity Analysis

4.1 Sensitivity Analysis Theory

Recall that we consider dynamics of the general form: $\dot{\mathbf{x}} = f(t, \mathbf{x}),\ \mathbf{x}(0) \in \mathcal{X}_0$. As a function of $\mathbf{x}_0$, the flow $\xi_{\mathbf{x}_0}$ is differentiable w.r.t $\mathbf{x}_0$. Thus the *sensitivity to initial conditions* at time t is well defined by:

$$\mathbf{s}_{\mathbf{x}_0}(t) \triangleq \frac{\partial \xi_{\mathbf{x}_0}}{\partial \mathbf{x}_0}(t). \tag{6}$$

where $\mathbf{s}_{\mathbf{x}_0}(t)$ is a square matrix of order n. To compute the sensitivity matrix, we first apply the chain rule to get the derivative of $\mathbf{s}_{\mathbf{x}_0}$ w.r.t. time:

$$\frac{\partial}{\partial t}\frac{\partial \xi_{\mathbf{x}_0}}{\partial \mathbf{x}_0}(t) = \frac{\partial}{\partial \mathbf{x}_0} f\big(t, \xi_{\mathbf{x}_0}(t)\big) = D_f(\xi_{\mathbf{x}_0}(t))\ \frac{\partial \xi_{\mathbf{x}_0}}{\partial \mathbf{x}_0}(t)$$

which gives the following *sensitivity equation*:

$$\dot{\mathbf{s}}_{\mathbf{x}_0}(t) = D_{f,\mathbf{x}_0}(t)\ \mathbf{s}_{\mathbf{x}_0}(t) \tag{7}$$

where $D_{f,\mathbf{x}_0}$ is the Jacobian matrix of f along trajectory $\xi_{\mathbf{x}_0}$. Hence, this equation is a linear time-varying ordinary differential equation (ODE). Note that this is a *matrix* differential equation but it can be viewed as a system of n ODEs of order n. The ij^{th} element of $\mathbf{s}_{\mathbf{x}_0}(t)$ basically represents the influence of variations in the i^{th} coordinate x_0^i of $\mathbf{x}_0$ on the j^{th} coordinate $x^j(t)$ of $\xi_{\mathbf{x}_0}(t)$. Then it is clear that the initial value $\mathbf{s}_{\mathbf{x}_0}(0)$ of $\mathbf{s}_{\mathbf{x}_0}$ must be the identity matrix, $\mathbf{I}_n$. Efficient solvers exist that implement the computation of sensitivity functions (in our implementations, we use the tool suite described in [SH05]).
A particularly interesting case is when the dynamics is linear time-varying, i.e. when $f(t,\mathbf{x}) = A(t)\ \mathbf{x}$. Indeed, in this case, we know that the Jacobian matrix of f is just the matrix A which means that sensitivity matrix $\mathbf{s}_{\mathbf{x}_0}(t)$ share *the same dynamics* as the flow $\xi_{\mathbf{x}_0}$. In fact, the columns of sensitivity matrix are solutions of the system dynamics equation where initial conditions are the canonical vectors of $\mathbb{R}^n$.

4.2 Sensitivity Functions and Expansion Functions

The following important result relates sensitivity functions to expansion functions:

Theorem 3. *Let $\mathbf{x}_0 \in \mathcal{X}_0$, $t \in [0,T]$ and assume that f is C^2. Then there exists a real $M > 0$ such that $\forall \epsilon > 0$:*

$$|\mathcal{E}_{\mathbf{x}_0,\epsilon}(t) - \|\mathbf{s}_{\mathbf{x}_0}(t)\|\ \epsilon\ | \leq M\epsilon^2 \tag{8}$$

Proof. Since f is C^2, the flow $\xi_{\mathbf{x}_0}$ is also C^2 w.r.t. $\mathbf{x}_0$ ([HS74]). Let $\mathbf{x} \in \mathcal{X}_0$. Then the Taylor expansion of $\xi_{\mathbf{x}_0}(t)$ around $\mathbf{x}_0$ shows that there exist a bounded function φ_t such that:

$$\begin{aligned} \xi_{\mathbf{x}}(t) &= \xi_{\mathbf{x}_0}(t) + \frac{\partial \xi_{\mathbf{x}_0}}{\partial \mathbf{x}_0}(t)\ (\mathbf{x}-\mathbf{x}_0) + \|\mathbf{x}-\mathbf{x}_0\|^2\ \varphi_t(\mathbf{x}-\mathbf{x}_0) \\ \Leftrightarrow \xi_{\mathbf{x}}(t) - \xi_{\mathbf{x}_0}(t) &= \mathbf{s}_{\mathbf{x}_0}(t)\ (\mathbf{x}-\mathbf{x}_0) + \|\mathbf{x}-\mathbf{x}_0\|^2\ \varphi_t(\mathbf{x}-\mathbf{x}_0) \end{aligned} \tag{9}$$

Equation (9) implies that $\forall \mathbf{x} \in \mathcal{B}_\epsilon(\mathbf{x}_0)$,

$$\|\xi_{\mathbf{x}}(t) - \xi_{\mathbf{x}_0}(t)\| \leq \|\mathbf{s}_{\mathbf{x}_0}(t)\|\|\mathbf{x}-\mathbf{x}_0\| + \|\mathbf{x}-\mathbf{x}_0\|^2\ \|\varphi_t(\mathbf{x}-\mathbf{x}_0)\| \leq \|\mathbf{s}_{\mathbf{x}_0}(t)\|\epsilon + \epsilon^2\ M$$

which implies in turn that

$$\mathcal{E}_{\mathbf{x}_0,\epsilon} - \|\mathbf{s}_{\mathbf{x}_0}(t)\|\epsilon \leq M\epsilon^2 \tag{10}$$

On the other hand, 9 can be rewritten as

$$\begin{aligned} \mathbf{s}_{\mathbf{x}_0}(t)\ (\mathbf{x}_0 - \mathbf{x}) &= \xi_{\mathbf{x}}(t) - \xi_{\mathbf{x}_0}(t) - \|\mathbf{x}-\mathbf{x}_0\|^2\ \varphi_t(\mathbf{x}-\mathbf{x}_0) \\ \Rightarrow \|\mathbf{s}_{\mathbf{x}_0}(t)\ (\mathbf{x}_0 - \mathbf{x})\| &\leq \|\xi_{\mathbf{x}}(t) - \xi_{\mathbf{x}_0}(t)\| + \|\mathbf{x}-\mathbf{x}_0\|^2\ \|\varphi_t(\mathbf{x}-\mathbf{x}_0)\| \\ &\leq \mathcal{E}_{\mathbf{x}_0,\epsilon}(t) + \epsilon^2 M \end{aligned} \tag{11}$$

From the definition of matrix norm, we know that we can find a unit vector $\mathbf{y}$ such that $\|\mathbf{s}_{\mathbf{x}_0}(t)\| = \|\mathbf{s}_{\mathbf{x}_0}(t)\ \mathbf{y}\|$. The inequality (11) is true for all $\mathbf{x} \in \mathcal{B}_\epsilon(\mathbf{x}_0)$ so in particular for $\mathbf{x} = \mathbf{x}_0 + \epsilon\mathbf{y}$ in which case

$$\|\mathbf{s}_{\mathbf{x}_0}(t)\ (\mathbf{x}_0 - \mathbf{x})\| = \|\mathbf{s}_{\mathbf{x}_0}(t)\ (\epsilon\mathbf{y})\| = \|\mathbf{s}_{\mathbf{x}_0}(t)\|\epsilon.$$

If we substitute in the right hand side of (11) and subtract $\mathcal{E}_{\mathbf{x}_0,\epsilon}(t)$, we get:

$$\|\mathbf{s}_{\mathbf{x}_0}(t)\|\epsilon - \mathcal{E}_{\mathbf{x}_0,\epsilon}(t) \leq M\epsilon^2 \tag{12}$$

The conjunction of inequalities (10) and (12) proves the result. □

When the dynamics of the system is affine, i.e. when $f(t, \mathbf{x}) = A(t)\mathbf{x} + b(t)$, where $A(t)$ and $b(t)$ are time varying matrices of appropriate dimensions, then expansion function can be computed exactly.

Theorem 4. *Let $\mathbf{x}_0 \in \mathcal{X}_0$, $t \in [0, T]$ and assume that f is affine. Then $\forall\epsilon > 0$:*

$$\mathcal{E}_{\mathbf{x}_0,\epsilon}(t) = \|\mathbf{s}_{\mathbf{x}_0}(t)\|\epsilon \tag{13}$$

Proof. This follows immediately from the fact that if f is affine, φ_t in equation (9) is null. Indeed, following the remark at the end of the previous subsection, we know from (7) that the lines of matrix $\mathbf{s}_{\mathbf{x}_0}(t)$ are solutions of the homogeneous system $\dot{\mathbf{x}} = A(t)\mathbf{x}$. Since this is a linear system, the vector $\mathbf{s}_{\mathbf{x}_0}(t)\ (\mathbf{x} - \mathbf{x}_0)$ is also solution of this system. Then $\xi_{\mathbf{x}_0}(t) + \mathbf{s}_{\mathbf{x}_0}(t)\ (\mathbf{x} - \mathbf{x}_0)$ is solution of the full system $\dot{\mathbf{x}} = A(t)\mathbf{x} + b(t)$. Furthermore, as $\mathbf{s}_{\mathbf{x}_0}(0)$ is the identity matrix,

$$\xi_{\mathbf{x}_0}(0) + \mathbf{s}_{\mathbf{x}_0}(0)\ (\mathbf{x} - \mathbf{x}_0) = \mathbf{x}_0 + (\mathbf{x} - \mathbf{x}_0) = \mathbf{x}.$$

In other words, $\xi_{\mathbf{x}_0} + \mathbf{s}\ (\mathbf{x} - \mathbf{x}_0)$ and $\xi_{\mathbf{x}}$ are two trajectories of the system with the same initial conditions so by uniqueness, they are equal. Then clearly,

$$\begin{aligned} \xi_{\mathbf{x}}(t) - \xi_{\mathbf{x}_0}(t) &= \mathbf{s}_{\mathbf{x}_0}(t)\ (\mathbf{x}_0 - \mathbf{x}) \\ \Rightarrow \sup_{\mathbf{x}\in\mathcal{B}_\epsilon(\mathbf{x}_0)} \|\xi_{\mathbf{x}}(t) - \xi_{\mathbf{x}_0}(t)\| &= \sup_{\mathbf{x}\in\mathcal{B}_\epsilon(\mathbf{x}_0)} \mathbf{s}_{\mathbf{x}_0}(t)\ (\mathbf{x}_0 - \mathbf{x}) \\ \Leftrightarrow \mathcal{E}_{\mathbf{x}_0,\epsilon}(t) &= \|\mathbf{s}_{\mathbf{x}_0}(t)\|\epsilon.\ \square \end{aligned}$$

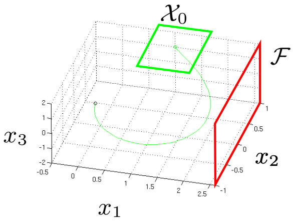
From what precedes, then, we can approximate $\mathcal{E}_{\mathbf{x}_0,\epsilon}(t)$ with the quantity $\|\mathbf{s}_{\mathbf{x}_0}\|\epsilon$ and use it to implement Algorithm 1. In the case of affine systems, the implementation is exact and Theorem 1 applies to the concrete implantation. In the general case, when f may be nonlinear, we know that the error is quadratic with respect to ϵ. In order to take this error into account, we can force the algorithm to guarantee that the initial set is sampled with a sufficiently small dispersion ϵ. The new algorithm then takes an additional input parameter $\epsilon > 0$, refine globally the initial sampling until the dispersion ϵ is reached (while checking only for unsafe trajectories), and then continues and terminates as Algorithm 1 with local refinements.

5 Examples

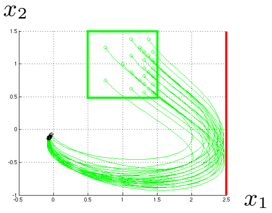
We have implemented the techniques described in the preceding sections on top of a numerical simulation tool that supports sensitivity analysis and have applied it to several examples.

5.1 A High Dimensional Affine Time-varying System

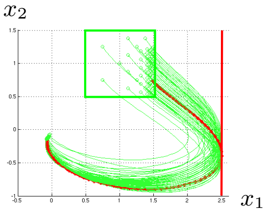
We consider a system with affine dynamics of the form $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{b}(t)$ with $\mathbf{A}(t) = e^{-t}\mathbf{M} - \mathbf{I}_{50}$ and $\mathbf{b}(t) = \mathbf{b}_0 e^{-t} \sin t$ where $\mathbf{M}$ and $\mathbf{b}_0$ are respectively 50×50 and 50×1 matrices with random coefficients in $[0,1]$. We used a 2-dimensional $\mathcal{X}_0 = [0.5, 1.5] \times [0.5, 1.5] \times \{1\}^{48}$. The bad set $\mathcal{F}$ is the half plane given by an inequality of the form $x_1 \leq d$. The figure below illustrates the behavior of the verification algorithm in different scenarios (projected on the three first coordinates). In all cases, a small number of trajectories was needed to obtain the answer.



$d = 2.6$: One trajectory was enough to prove that the system is safe.



$d = 2.5$: The system is declared uncertain using $\delta = 0.1$ after 25 trajectories.

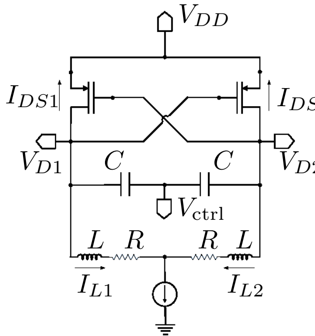


$d = 2.5$: The system was found unsafe with $\delta = 0.01$ after 63 trajectories.

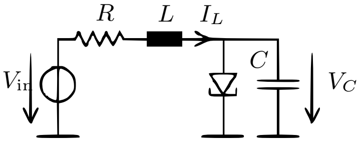
5.2 Verifying the invariant of two oscillator circuits

For the following examples, our goal is to prove that a set is invariant for an unbounded horizon. To do this, the classical idea is to show that for a certain $T > 0$, the set $\text{Reach}_{=T}$ is contained in $\text{Reach}_{\leq t}$ with $t < T$ which implies that $\text{Reach}_{\leq T}$ is the reachable set for unbounded horizon. Our method is to use our verification algorithm slightly modified so that *every* trajectory is considered as uncertain. As previously, the algorithm stops when $\delta_k < \delta$. At this point, then, we can characterize the reachable sets thanks to inclusions (5) of Corollary 1.

We applied this idea to analyze the periodic steady state behavior of two analog oscillator circuits. The first one is a *tunnel-diode oscillator* (TDO) whose second-order nonlinear dynamics is given in Figure 2. The second circuit is a *voltage controlled oscillator* (VCO) circuit, the schema of which is given on the right. Its dynamics is governed by a third-order nonlinear equation. A fully-detailed model can be found in [FKR06].

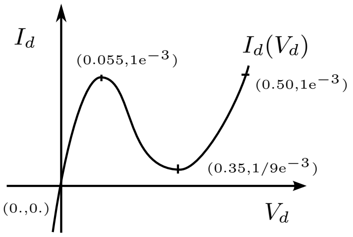


VCO schema



$$\dot{V}_d = \frac{1}{C}(-I_d(V_d) + I_L)$$
$$\dot{I}_L = \frac{1}{C}(-I_d(V_d) + I_L)$$

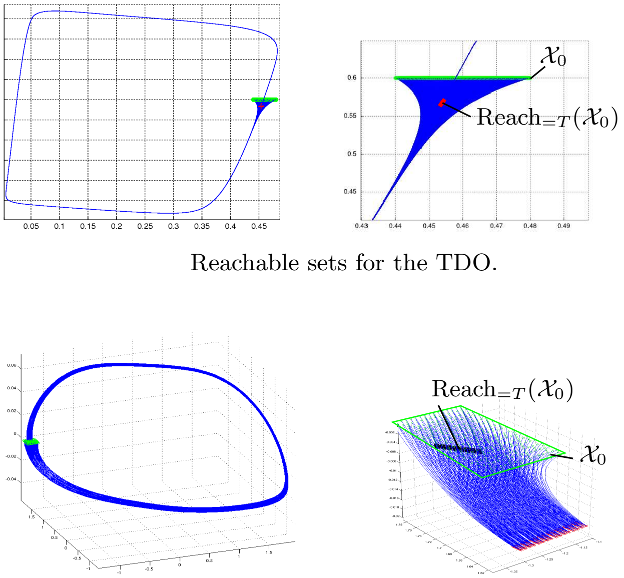
Diode characteristic:



where $C = 1pF$, $L = 1\mu H$, $G = 5m\Omega^{-}1$, $V_{\rm in} = 0.3V$.

Fig. 2. Tunnel Diode Oscillator Circuit

What makes this problem difficult for traditional tools performing reachability is that most often, the reachable set is computed step by step forward in time and each step increases the error of over-approximation. This error after one period may have become too large to prove the invariant property. This is particularly serious for the VCO for which the limit cycle is much less contractive than for the TDO. In [FKR06], this problem is addressed using forward-backward refinement. Our method, however, does not suffer from this problem. If we neglect the inherent error of the numerical solver used to compute the trajectories, the quality of the over-approximation that we get with the sensitivity function does not depend on the time we measure it. If the dynamics is neither really contractive nor diverging as for the VCO, this means that the norm of the sensitivity function will remain near 1 and then we know from Theorem 1 and Theorem 3 that if we sample the initial set with a precision ϵ then we will get an approximation of the reachable set at time T with a precision which is quadratic w.r.t. ϵ. We show some results we obtained in Figure 3. In both cases, we sampled the initial set with $\epsilon = 1 \times 10^{-3}$. Since ϵ^2 is then negligible before the size of the reachable sets $\text{Reach}_{=T}(\mathcal{X}_0)$ that we obtain, we can conclude that the sets are invariant. Since, we must note that with this method, we did not yet obtain a *formal* proof of the result because of the remaining indetermination of constant M in Theorem 3. To get this formal bound, a deeper analysis of the dynamics equations still needs to be done. Techniques and tools recently developed are related to this issue (see [SB06]).



Reachable sets for the TDO.

Reachable sets for the VCO.

Fig. 3. Verifying the invariant of two oscillator circuits.

6 Extension to Hybrid Systems

Extending sensitivity analysis to the hybrid case is not straightforward and even in the simplest case of a transition with state continuity, a discontinuity often appears in the sensitivity function that needs to be evaluated. The most general setting for sensitivity analysis includes hybrid systems for which dynamics in each mode is governed by differential algebraic equations [HP00,BL02] To simplify the presentation and get the intuition of what are the changes induced by the hybridicity of the dynamics, we restrict the study to the case of a unique transition between two modes. Let us assume that transitions are governed by crossings of an hyper-surface $\mathcal{G}$ implicitly defined by a continuous function g and that the flow is continuous. The dynamics of this system is described by:

$$\dot{\mathbf{x}} = \begin{cases} f_1(t, \mathbf{x}) \text{ if } g(\mathbf{x}) < 0 \\ f_2(t, \mathbf{x}) \text{ if } g(\mathbf{x}) \geq 0 \end{cases}, \mathbf{x}(0) \in \mathcal{X}_0 \tag{14}$$

We consider a trajectory $\xi_{\mathbf{x}_0}$ performing a first transition at time $\tau > 0$, i.e. such that $g(\xi_{\mathbf{x}_0}(t)) < 0 \ \forall t \in [0, \tau[$ and $g(\xi_{\mathbf{x}_0}(t)) = 0$. We make the following standard assumption:

Assumption 1 *At the crossing point* $\mathbf{x}$*,* $\langle \nabla_{\mathbf{x}} g(\mathbf{x}), f_1(\tau^-, \mathbf{x}) \rangle \neq 0$*, where* $\langle , \rangle$ *is the Euclidean cross product. Moreover, there exists a neighborhood* $\mathcal{N}$ *of* $\mathbf{x}_0$ *such that for all* $\mathbf{x} \in \mathcal{N}$*, this assumption is also true for the flow* $\xi_{\mathbf{x}}$*.*

Assumption 1 prevents the trajectory to cross the frontier tangentially and ensures that there exists a tube of trajectories around $\xi_{\mathbf{x}_0}$ which also crosses the frontier under the same condition. In this setting, we consider the most standard behavior of a hybrid system, i.e. it follows a continuous trajectory for some time, then switches to another continuous mode for again some time and so on. During a continuous evolution, we know how sensitivity evolves. The remaining question is about its continuity at transition times. We have the following

Proposition 2. *Under assumptions 1, the sensitivity function at time* τ *satisfies:*

$$\mathbf{s}(\tau^+) - \mathbf{s}(\tau^-) = \frac{d\tau}{d\mathbf{x}_0} \big(f_2(\tau, \xi_{\mathbf{x}_0}(\tau)) - f_1(\tau, \xi_{\mathbf{x}_0}(\tau)) \big) \tag{15}$$

$$\textit{where } \frac{d\tau}{d\mathbf{x}_0} = \frac{\langle \nabla_{\mathbf{x}} g(\xi_{\mathbf{x}_0}(\tau)), \mathbf{s}_{\mathbf{x}_0}(\tau) \rangle}{\langle \nabla_{\mathbf{x}} g(\xi_{\mathbf{x}_0}(\tau)), f_1(\tau, \xi_{\mathbf{x}_0}) \rangle} \tag{16}$$

We omit the proof for brevity (it can be found in [HP00,BL02]). Rather, we provide a picture in Figure 4 giving an intuition of why a a discontinuity happens.

Proposition 2 provides a constructive formula to compute the values of the jumps. This means that sensitivity functions can be computed for hybrid trajectories and thus Algorithm 1 can be implemented. The assumption made is reasonable in the sense that it is very likely that the set of points for which the frontier is crossed tangentially has a zero measure. Still, current work investigates the behavior of our algorithm around such points, along with the adaptation of Theorem 3 and 4 to the hybrid case.

7 Conclusion

We have developed a novel and general simulation-based method for proving safety of arbitrary continuous systems and demonstrated its effectiveness on several non trivial examples. This method can treat arbitrary nonlinear systems and can be particularly efficient for affine time-varying systems. As shown in Section 6, it can, under reasonable assumptions, be extended to hybrid systems as well. The use of the tunable tolerance parameter δ gives an elegant solution

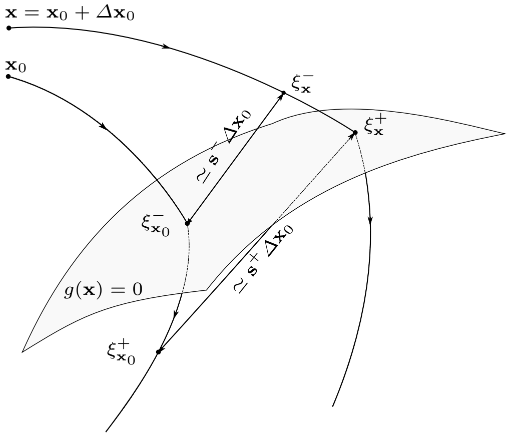

$\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x}_0$
$\mathbf{x}_0$
$\xi_{\mathbf{x}}^-$
$\xi_{\mathbf{x}}^+$
$\tilde{\mathbf{s}}^- \Delta\mathbf{x}_0$
$\xi_{\mathbf{x}_0}^-$
$\tilde{\mathbf{s}}^+ \Delta\mathbf{x}_0$
$g(\mathbf{x}) = 0$
$\xi_{\mathbf{x}_0}^+$

Fig. 4. Illustration of sensitivity discontinuity. The jump condition 15 results from the fact that between τ^- and τ^+ ($-$ and $+$ superscripts in the figure), the flows $\xi_{\mathbf{x}_0}$ and $\xi_{\mathbf{x}}$ evolve with different dynamics f_1 and f_2.

to the eternal tension between finite algorithmic termination and the potential infinite precision of the real numbers. In addition to its theoretical properties and efficiency, our method is more likely to be accepted by practitioners who already use simulation as a key validation tool.

Future work will extend the implementation to hybrid automata, look at the problem of unbounded horizon and, most importantly, deal with *non-autonomous* systems with *bounded inputs*. An appropriate sampling of the input space combined with the use of search heuristics (branch and bound, RRT etc.), and/or optimal control techniques should provide interesting results which could be incorporated naturally into the design process of complex control systems.

References

ACH$^+$95. 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.

ADG03. E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In Oded Maler and Amir Pnueli, editors, *Hybrid Systems: Computation and Control*, LNCS 2623, pages 20–35. Springer-Verlag, 2003.

BCLM05. M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan. Sampling-based reachability algorithms for control and verification of complex systems. In *Proc. Thirteenth Yale Workshop on Adaptive and Learning Systems*, June 2005.

BCLM06. M.S. Branicky, M.M. Curtiss, J. Levine, and S. Morgan. Sampling-based planning, control and verification of hybrid systems. *Control Theory and Applications, IEE Proceedings*, 153:575–590, Sept. 2006.

BF04. A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In R. Alur and G. J. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 142–156. Springer, 2004.

BL02. P. I. Barton and C. Kun Lee. Modeling, simulation, sensitivity analysis, and optimization of hybrid systems. *ACM Trans. Model. Comput. Simul.*, 12(4):256–289, 2002.

CK98. A. Chutinan and B.H. Krogh. Computing polyhedral approximations to dynamic flow pipes. In *Proc. of the 37th Annual International Conference on Decision and Control, CDC'98.* IEEE, 1998.

DM98. T. Dang and O. Maler. Reachability analysis via face lifting. In T.A. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, LNCS 1386, pages 96–109. Springer-Verlag, 1998.

FKR06. G. Frehse, B. H. Krogh, and R. A. Rutenbar. Verifying analog oscillator circuits using forward/backward abstraction refinement. In *DATE 2006: Design, Automation and Test in Europe*, March 2006.

Gir05. A. Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems : Computation and Control*, LNCS 3414, pages 291–305. Springer, 2005.

GP06. A. Girard and G. J. Pappas. Verification using simulation. In *Hybrid Systems : Computation and Control*, volume 3927 of *LNCS*, pages 272–286. Springer-Verlag, 2006.

HP00. I.A. Hiskens and M.A. Pai. Trajectory sensitivity analysis of hybrid systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(2):204–220, February 2000.

HS74. M.W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems and Linear Algebra.* Academic Press, 1974.

KKMS03. J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg. On systematic simulation of open continuous systems. In *HSCC*, pages 283–297, 2003.

LaV06. S. M. LaValle. *Planning Algorithms*, chapter 5. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

LYL04. S. R. Lindemann, A. Yershova, and S. M. LaValle. Incremental grid sampling strategies in robotics. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, pages 297–312, 2004.

SB06. A. B. Singer and P. I. Barton. Bounding the solutions of parameter dependent nonlinear ordinary differential equations. *SIAM Journal on Scientific Computing*, 27(6):2167–2182, 2006.

SH05. R. Serban and A. C. Hindmarsh. Cvodes: the sensitivity-enabled ode solver in sundials. In *Proceedings of IDETC/CIE 2005*, Long Beach, CA., Sept. 2005.