Form Methods Syst Des (2018) 53:363–383
https://doi.org/10.1007/s10703-017-0305-8

An improved algorithm for the control synthesis of nonlinear sampled switched systems

Adrien Le Coënt[1] · **Julien Alexandre dit Sandretto**[2] · **Alexandre Chapoutot**[2] · **Laurent Fribourg**[1]

Published online: 11 November 2017

Abstract A novel algorithm for the control synthesis for nonlinear switched systems is presented in this paper. Based on an existing procedure of state-space bisection and made available for nonlinear systems with the help of guaranteed integration, the algorithm has been improved to be able to consider longer patterns of modes with a better pruning approach. Moreover, the use of guaranteed integration also permits to take bounded perturbations and varying parameters into account. It is particularly interesting for safety critical applications, such as in aeronautical, military or medical fields. The whole approach is entirely guaranteed and the induced controllers are *correct-by-design*. Some experimentations are performed to show the important gain of the new algorithm.

Keywords Nonlinear control systems · Reachability · Formal methods · Numerical simulation · Control system synthesis

This work is supported by Institut Farman (through the project SWITCHDESIGN), by the French National Research Agency through the "iCODE Institute project" funded by the IDEX Paris-Saclay, ANR-11-IDEX-0003-02, and by Labex DigiCosme (Project ANR-11-LABEX-0045-DIGICOSME).

✉ Alexandre Chapoutot
chapoutot@ensta.fr

Adrien Le Coënt
adrien.le-coent@ens-cachan.fr

Julien Alexandre dit Sandretto
alexandre@ensta.fr

Laurent Fribourg
fribourg@lsv.fr

[1] CMLA and LSV, CNRS, ENS Paris-Saclay, INRIA, Université Paris-Saclay, 61 av. du Président Wilson, 94230 Cachan, France

[2] U2IS Department, ENSTA ParisTech, Université Paris-Saclay, 828, Boulevard des Maréchaux, 91762 Palaiseau Cedex, France

1 Introduction

In this paper, we present a control synthesis method for a special form of hybrid systems [1] named *switched systems*. Such systems have been recently used in various domains such as automotive industry and, with noteworthy success, power electronics (e.g., power converters). They are continuous-time systems with discrete switching events. More precisely, these systems are described by piecewise continuous dynamics called *modes*; the change of modes takes place instantaneously at so-called *switching instants*. In this paper, we suppose that switching instants occur *periodically* at constant sampling period τ (*sampled* switched systems). The *control synthesis* problem consists in finding a *switching rule* σ in order to satisfy a given specification. At each sampling time τ, 2τ, …, according to the state of the system, the rule σ selects an appropriate mode to fulfill the specification [2,3]. A schematic view of switched systems is given in Fig. 1.

Modes are characterized by nonlinear Ordinary Differential Equations (ODEs). In general, the exact solution of differential equations cannot be obtained, and a numerical integration scheme is used to approximate the state of the system. With the objective of computing a sound control, our approach uses *guaranteed numerical integration methods*, also called "sound reachability" methods. A guaranteed method is a numerical method which provides a formal proof of its result.

Guaranteed numerical integration methods have to consider two kinds of problems. First, the way of representating sets of states (boxes, zonotopes [4,5], Taylor models [6], Support functions [7], etc.). Second, the scheme of the numerical integration (Taylor series [8–11], Runge-Kutta methods [12–15], etc.) which propagate sets of states through the dynamics of the system. In this paper we follow [15] in which sets are represented by zonotopes and propagation of sets is based on the Runge-Kutta numerical scheme.

This work is along the line of the seminal paper [16] on hybrid systems, and can be seen as an optimized application in the context of sampled switched systems. Other guaranteed approaches for control synthesis of switched systems include the construction of a discrete abstraction of the original system on a grid of the state space: e.g., approximately bisimilar models [17], approximately alternatingly similar models [18], or feedback refinement relations [19].

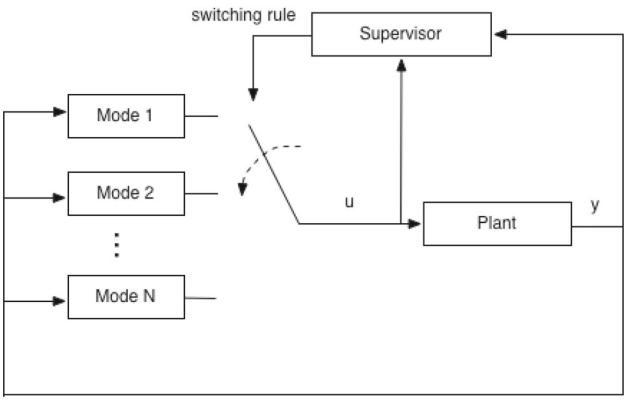

Fig. 1 Sampled switched systems (with u the input and y the state of the system)

In [20], we proposed an algorithm based on guaranteed integration for the synthesis of nonlinear switched system controllers. The specification to fulfill is to reach a target zone R from an initial zone then return iteratively to R while always staying in a safe neighbourhood S of R while avoiding bad states B. This is done be covering R with a set of tiles for which a sequence of modes (pattern) satisfying the specification has to be found. A similar objective has been treated in [21], but the authors use a proof certificate in the form of a robust control Lyapunov-like function instead.

In this paper, we present an improved version of the branch and prune algorithm of [20] which permits to consider more modes and longer patterns, using a suitable pruning approach. This leads to a strong decrease in computation time. The new algorithm allows us to handle harder problems.

The paper is divided as follows. In Sect. 2, we introduce some preliminaries on switched systems and some notation used in the following. In Sect. 3, the guaranteed integration of nonlinear ODEs is presented. In Sect. 4, we present the main algorithm of state-space bisection used for control synthesis. In Sect. 5, the whole approach is tested on four examples of the literature. We give some performance tests and compare our approach with the state-of-the-art tools in Sect. 6. We conclude in Sect. 7.

2 Switched systems

Let us consider nonlinear switched systems such that

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t)) \tag{1}$$

defined for all $t \geq 0$, where $x(t) \in \mathbb{R}^n$ is the state of the system, $\sigma(\cdot)\mathbb{R}^+ \longrightarrow U$ is the switching rule, and $d(t) \in \mathbb{R}^m$ is a bounded perturbation. The finite set $U = \{1, \ldots, N\}$ is the set of switching modes of the system. We focus on sampled switched systems: given a sampling period $\tau > 0$, switchings will occur at times $\tau, 2\tau, \ldots$. Switchings depend only on time, and not on states: this is the main difference with hybrid systems.

We work in the *synchronous* setting for discrete events. This means that all the discrete events are supposed to occur at periodic instants: $\tau, 2\tau, 3\tau, \ldots$ The switching rule $\sigma(\cdot)$ is thus piecewise constant, we will consider that $\sigma(\cdot)$ is constant on the time interval $[(k-1)\tau, k\tau)$ for $k \geq 1$. We call "*pattern*" a finite sequence of modes $\pi = (i_1, i_2, \ldots, i_k) \in U^k$. With such a control input, and under a given perturbation d, we will denote by $x(t; t_0, x_0, d, \pi)$ the solution at time t of the system

$$\begin{aligned} \dot{x}(t) &= f_{\sigma(t)}(x(t), d(t)), \\ x(t_0) &= x_0, \\ \forall j \in \{1, \ldots, k\},\ \sigma(t) &= i_j \in U \text{ for } t \in [(j-1)\tau, j\tau[. \end{aligned} \tag{2}$$

We address the problem of synthesizing a state-dependent switching rule $\sigma(\cdot)$ for Eq. (2) in order to verify some properties. This important problem is formalized as follows:

Problem 1 (*Control Synthesis Problem*) Let us consider a sampled switched system as defined in Eq. (2). Given three sets R, S, and B, with $R \cup B \subset S$ and $R \cap B = \emptyset$, find a rule $\sigma(\cdot)$ such that, for any $x(0) \in R$

– $\tau - stability$[1] $x(t)$ returns in R infinitely often, at some multiples of sampling time τ.
– *safety* $x(t)$ always stays in $S \backslash B$.

Under the above-mentioned notation, we propose the main procedure of our approach which solves this problem by constructing a law $\sigma(\cdot)$, such that for all $x_0 \in R$, and under the unknown bounded perturbation d, there exists $\pi = \sigma(\cdot) \in U^k$ for some k such that:

$$\begin{cases} x(t_0 + k\tau; t_0, x_0, d, \pi) & \in R \\ \forall t \in [t_0, t_0 + k\tau], \quad x(t; t_0, x_0, d, \pi) & \in S \\ \forall t \in [t_0, t_0 + k\tau], \quad x(t; t_0, x_0, d, \pi) & \notin B. \end{cases}$$

Such a law permits to perform an infinite-time state-dependent control. The synthesis algorithm is described in Sect. 4 and involves guaranteed set-based integration presented in the next section, the main underlying tool is interval analysis [8]. To tackle this problem, we introduce some definitions.

In the following, we will often use the notation $[x] \in \mathbb{IR}$ (the set of intervals with real bounds) where

$$[x] = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leqslant x \leqslant \overline{x}\}$$

denotes an interval. By an abuse of notation $[x]$ will also denote a vector of intervals, i.e., a Cartesian product of intervals, a.k.a. a *box*. In the following, the sets R, S and B are given in the form of boxes. With interval values, it comes with an associated interval arithmetic.

Interval arithmetic extends to $\mathbb{IR}$ elementary functions over $\mathbb{R}$. For instance, the interval sum, i.e., $[x_1] + [x_2] = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}]$, encloses the image of the sum function over its arguments. The enclosing property basically defines what is called an *interval extension* or an *inclusion function*.

Definition 1 (*Inclusion Function*) Consider a function $f : \mathbb{R}^n \to \mathbb{R}^m$, then $[f] : \mathbb{IR}^n \to \mathbb{IR}^m$ is said to be an extension of f to intervals if

$$\forall [x] \in \mathbb{IR}^n, \quad [f]([x]) \supseteq \{f(x), x \in [x]\}.$$

It is possible to define inclusion functions for all elementary functions such as $\times$, $\div$, sin, cos, exp, and so on. The *natural* inclusion function is the simplest to obtain: all occurrences of the real variables are replaced by their interval counterpart and all arithmetic operations are evaluated using interval arithmetic. More sophisticated inclusion functions such as the centered form, or the Taylor inclusion function may also be used (see [22] for more details).

We now introduce the Initial Value Problem, which is one of main ingredients of our approach.

Definition 2 [*Initial Value Problem (IVP)*] Consider an ODE with a given initial condition

$$\dot{x}(t) = f(t, x(t), d(t)) \quad \text{with} \quad x(0) \in X_0, \ d(t) \in [d], \tag{3}$$

with $f : \mathbb{R}^+ \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ assumed to be continuous in t and d and globally Lipschitz in x. We assume that parameters d are bounded (used to represent a perturbation, a modeling error, an uncertainty on measurement, …). An *IVP* consists in finding a function $x(t)$ described by Eq. (3) for all $d(t)$ lying in $[d]$ and for all the initial conditions in X_0.

[1] This definition of stability is different from the stability in the Lyapunov sense.

Definition 3 Let $X \subset \mathbb{R}^n$ be a box of the state space. Let $\pi = (i_1, i_2, \ldots, i_k) \in U^k$. The *successor set* of X via π, denoted by $Post_\pi(X)$, is the (over-approximation of the) image of X induced by application of the pattern π, i.e., the solution at time $t = k\tau$ of

$$\begin{aligned} \dot{x}(t) &= f_{\sigma(t)}(x(t), d(t)), \\ x(0) &= x_0 \in X, \\ \forall t \geq 0, \quad & d(t) \in [d], \\ \forall j \in \{1, \ldots, k\}, \quad & \sigma(t) = i_j \in U \text{ for } t \in [(j-1)\tau, j\tau). \end{aligned} \tag{4}$$

Definition 4 Let $X \subset \mathbb{R}^n$ be a box of the state space. Let $\pi = (i_1, i_2, \ldots, i_k) \in U^k$. We denote by $Tube_\pi(X)$ the union of boxes covering the trajectories of IVP (4), whose construction is detailed in Sect. 3.

Remark 1 $Post_\pi(X)$ is an over-approximation at time $t = k\tau$ of the set of states originated from the set X at time $t = 0$, while $Tube_\pi(X)$ is an over-approximation of the whole set of states between $t = 0$ and $t = k\tau$ originated from the set X at time $t = 0$.

3 Guaranteed integration

In this section, we describe our approach for guaranteed integration based on Runge-Kutta methods [14,15]. The goal being obviously to obtain a solution of the differential equations describing the modes of the nonlinear switched systems.

A numerical integration method computes a sequence of values (t_n, x_n) approximating the solution $x(t; t_0, x_0, d)$ of the IVP defined in Eq. (3) such that $x_n \approx x(t_n; x_{n-1})$. The simplest method is Euler's method in which $t_{n+1} = t_n + h$ for some step size h and $x_{n+1} = x_n + h \times f(t_n, x_n, d)$; so the derivative of x at time t_n, $f(t_n, x_n, d)$, is used as an approximation of the derivative on the whole time interval to perform a linear interpolation. This method is very simple and fast, but requires small step sizes. More advanced methods, coming from the Runge-Kutta family, use a few intermediate computations to improve the approximation of the derivative. The general form of an explicit s-stage Runge-Kutta formula, that is using s evaluations of f, is

$$\begin{aligned} x_{n+1} &= x_n + h \sum_{i=1}^{s} b_i k_i, \\ k_1 &= f\left(t_n,\ x_n, d\right), \\ k_i &= f\left(t_n + c_i h,\ x_n + h \sum_{j=1}^{i-1} a_{ij} k_j, d\right),\ i = 2, 3, \ldots, s. \end{aligned} \tag{5}$$

The coefficients c_i, a_{ij} and b_i fully characterize the method. To make Runge-Kutta validated, the challenging question is how to compute guaranteed bounds of the distance between the true solution and the numerical solution, defined by $x(t_n; t_{n-1}, x_{n-1}, d) - x_n$. This distance is associated to the *local truncation error* (LTE) of the numerical method.

To bound the LTE, we rely on *order condition* [23] respected by all Runge-Kutta methods. This condition states that a method of this family is of order p if and only if the $p + 1$ first coefficients of the Taylor expansion of the solution and the Taylor expansion of the numerical

methods are equal. In consequence, LTE is proportional to the Lagrange remainders of Taylor expansions. Formally, LTE is defined by (see [14]):

$$x(t_n; x_{n-1}) - x_n = \frac{h^{p+1}}{(p+1)!}\left(f^{(p)}\left(\xi, x(\xi; x_{n-1}), d\right) - \frac{d^{p+1}\phi}{dt^{p+1}}(\eta)\right) \quad \xi \in]t_n, t_{n+1}[\text{ and } \eta \in]t_n, t_{n+1}[. \tag{6}$$

The function $f^{(n)}$ stands for the n-th derivative of function f with respect to time t that is $\frac{d^n f}{dt^n}$ and $h = t_{n+1} - t_n$ is the step size. The function $\phi : \mathbb{R} \to \mathbb{R}^n$ is defined by $\phi(t) = x_n + h\sum_{i=1}^{s} b_i k_i$ where k_i are defined as Eq. (5).

The challenge to make Runge-Kutta integration schemes safe with respect to the true solution of IVP is then to compute a bound (or also an over-approximation) of the result of Eq. (6). In other words, we do have to bound the value of $f^{(p)}\left(\xi, x(\xi; t_{n-1}, x_{n-1}, d)\right), d)$ and the value of $\frac{d^{p+1}\phi}{dt^{p+1}}(\eta)$ with numerical guarantee. The latter expression is straightforward to bound because the function ϕ only depends on the value of the step size h, and so does its $(p+1)$-th derivative. The bound is then obtained using the affine arithmetic [24,25].

However, the expression $f^{(p)}\left(\xi, x(\xi; x_{n-1}), d\right)$ is not so easy to bound as it requires to evaluate f for a particular value of the IVP solution $x(\xi; t_{n-1}, x_{n-1}, d)$ at an unknown time $\xi \in]t_{n-1}, t_n[$. The solution used is the same as the one found in [9,12] and it requires to bound the solution of IVP on the interval $[t_{n-1}, t_n]$. This bound is usually computed using the Banach's fixpoint theorem applied with the Picard-Lindelöf operator, see [9]. This operator is used to compute an enclosure of the solution $[\tilde{x}]$ of IVP over a time interval $[t_{n-1}, t_n]$, that is for all $t \in [t_{n-1}, t_n]$, $x(t; t_{n-1}, x_{n-1}, d) \in [\tilde{x}]$. In its simple interval form, Picard-Lindelöf operator is defined by

$$[p_f]([r]) \stackrel{\text{def}}{=} [x_n] + [0, h]f([t_{n-1}, t_n], [r], [d]), \tag{7}$$

with h the integration step size. This is usually implemented by an iterative process and if $[\mathbf{r}]$ is found such that $[p_f]([r]) \subseteq [r]$ then $[\tilde{x}] \subseteq [r]$ by the Banach fixed-point theorem. More sophisticated versions of the Picard-Lindelöf operator exist, we refer to [9] for more details. We can hence bound $f^{(p)}$ substituting $x(\xi; t_{n-1}, x_{n-1}, d)$ by $[\tilde{x}]$. This general approach used to solve IVPs in a validated way is called the Lohner two step approach [26].

Remark 2 Note that to apply guaranteed numerical intergation aglorithms, we restrict perturbation d in Definition 2 to be constant over the integration step size h. Indeed, the computation of the truncation error implies high order time derivatives of f which is defined as a combination of time derivatives of x and d. As time derivatives of d are usually unkown, a simplification has been made to be able to compute a bound of Eq. (6). Note however that in the framework of differential inclusion [27], a time varying disturbance d can be taken into account and such adaptations could be used straightforwardly in our proposed algorithms.

With guaranteed numerical integration methods and for a given pattern of switched modes $\pi = (i_1, \ldots, i_k) \in U^k$ of length k, we are able to compute, for $j \in \{1, .., k\}$, the enclosures:

- $[x_j] \ni x(j\tau)$;
- $[\tilde{x}_j] \ni x(t)$, for $t \in [(j-1)\tau, j\tau]$

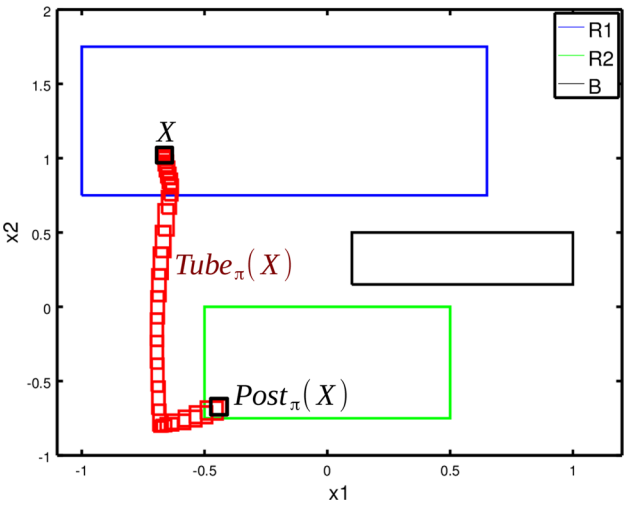

Fig. 2 Illustration of functions $Post_\pi(X)$ and $Tube_\pi(X)$ from the Example 5.2, for the initial box $X = \text{x1} \times \text{x2} = [-0.69, -0.64] \times [1, 1.06]$, with a pattern $\pi = (1, 3, 0)$

with respect to the system of IVPs:

$$\begin{cases} \dot{x}(t) = f_{\sigma(t)}(t, x(t), d(t)), \\ x(t_0 = 0) \in [x_0], d(t) \in [d], \\ \sigma(t) = i_1, \forall t \in [0, t_1], t_1 = \tau \\ \quad \vdots \\ \dot{x}(t) = f_{\sigma(t)}(t, x(t), d(t)), \\ x(t_{k-1}) \in [x_{k-1}], d(t) \in [d], \\ \sigma(t) = i_k, \forall t \in [t_{k-1}, t_k], t_k = k\tau. \end{cases} \tag{8}$$

Thereby, the enclosure $Post_\pi([x_0])$ is included in $[x_k]$ and $Tube_\pi([x_0])$ is included in $\bigcup_{j=1,..,k}[\tilde{x}_j]$. This applies for all initial states in $[x_0]$ and all disturbances $d(t) \in [d]$. A view of enclosures computed by guaranteed integration for one solution obtained for Example 5.2 is shown in Fig. 2.

4 The state space bisection algorithm

4.1 Principle of the algorithm

We describe the algorithm solving the control synthesis problem for nonlinear switched systems (see Problem 1, Sect. 2).
Given the input boxes R, S, B, and given two positive integers K and D, the algorithm provides, when it succeeds, a decomposition Δ of R of the form $\{V_i, \pi_i\}_{i \in I}$, with the properties:

- $\bigcup_{i \in I} V_i = R$,
- $\forall i \in I,\ Post_{\pi_i}(V_i) \subseteq R$,
- $\forall i \in I,\ Tube_{\pi_i}(V_i) \subseteq S$,
- $\forall i \in I,\ Tube_{\pi_i}(V_i) \bigcap B = \emptyset$.

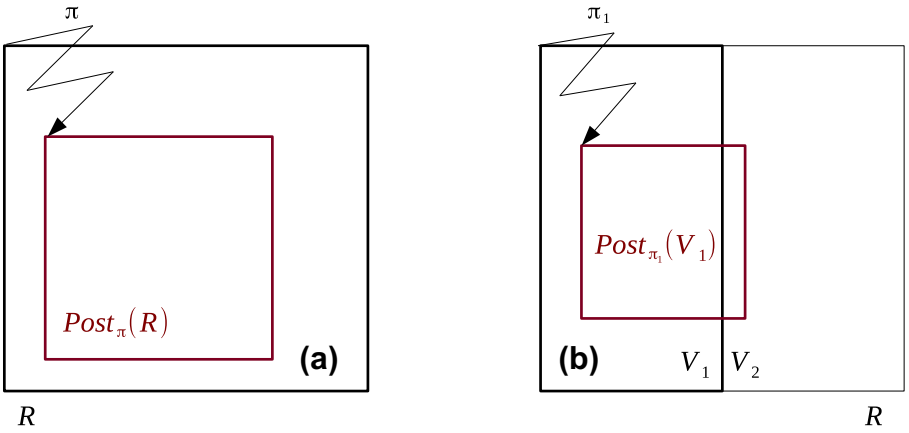

Fig. 3 Principle of the bisection method

The sub-boxes $\{V_i\}_{i\in I}$ are obtained by repeated bisection. At first, function $Decomposition$ calls sub-function $Find\_Pattern$ which looks for a pattern π of length at most K such that $Post_\pi(R) \subseteq R$, $Tube_\pi(R) \subseteq S$ and $Tube_\pi(R) \bigcap B = \emptyset$. If such a pattern π is found, then a uniform control over R is found (see Fig. 3a). Otherwise, R is divided into two sub-boxes V_1, V_2, by bisecting R with respect to its longest dimension. Patterns are then searched to control these sub-boxes (see Fig. 3b). If for each V_i, function $Find\_Pattern$ manages to get a pattern π_i of length at most K verifying $Post_{\pi_i}(V_i) \subseteq R$, $Tube_{\pi_i}(V_i) \subseteq S$ and $Tube_{\pi_i}(V_i) \bigcap B = \emptyset$, then it is a success and algorithm stops. If, for some V_j, no such pattern is found, the procedure is recursively applied to V_j. It ends with success when every sub-box of R has a pattern verifying the latter conditions, or fails when the maximal degree of decomposition D is reached. The algorithmic form of functions $Decomposition$ and $Find\_Pattern$ are given in Algorithms 1 and 2 respectively. Note that a special form of Algorithm 2 for linear ODEs can be found in [2].

Algorithm 1 $Decomposition$.

Function: $Decomposition(W, R, S, B, D, K)$

Input: A box W, a box R, a box S, a box B, a degree D of bisection, a length K of input pattern
Output: $\langle\{(V_i, \pi_i)\}_i, True\rangle$ or $\langle\_, False\rangle$

$(\pi, b) := Find\_Pattern(W, R, S, B, K)$
if $b = True$ **then**
  return $\langle\{(W, Pat)\}, True\rangle$
else
  if $D = 0$ **then**
    return $\langle\_, False\rangle$
  else
    Divide equally W into (W_1, W_2)
    for $i = 1, 2$ **do**
      $(\Delta_i, b_i) := Decomposition(W_i, R, S, B, D-1, K)$
    end for
    return $(\bigcup_{i=1,2} \Delta_i, \bigwedge_{i=1,2} b_i)$
  end if
end if

Our control synthesis method being well defined, we introduce the main result of this paper, stated as follows:

Proposition 1 *Algorithm 1 with input* (R, R, S, B, D, K) *returns, when it successfully terminates, a decomposition* $\{V_i, \pi_i\}_{i\in I}$ *of* R *which solves Problem 1.*

Proof Let $x_0 = x(t_0 = 0)$ be an initial condition belonging to R. If the decomposition has terminated successfully, we have $\bigcup_{i\in I} V_i = R$, and x_0 thus belongs to V_{i_0} for some $i_0 \in I$. We can thus apply the pattern π_{i_0} associated to V_{i_0}. Let us denote by k_0 the length of π_{i_0}. We have:

- $\mathbf{x}(k_0\tau; 0, x_0, d, \pi_{i_0}) \in R$
- $\forall t \in [0, k_0\tau], \quad \mathbf{x}(t; 0, x_0, d, \pi_{i_0}) \in S$
- $\forall t \in [0, k_0\tau], \quad \mathbf{x}(t; 0, x_0, d, \pi_{i_0}) \notin B.$

Let $x_1 = \mathbf{x}(k_0\tau; 0, x_0, d, \pi_{i_0}) \in R$ be the state reached after application of π_{i_0} and let $t_1 = k_0\tau$. State x_1 belongs to R, it thus belongs to V_{i_1} for some $i_1 \in I$, and we can apply the associated pattern π_{i_1} of length k_1, leading to:

- $\mathbf{x}(t_1 + k_1\tau; t_1, x_1, d, \pi_{i_1}) \in R$
- $\forall t \in [t_1, t_1 + k_1\tau], \quad \mathbf{x}(t; t_1, x_1, d, \pi_{i_1}) \in S$
- $\forall t \in [t_1, t_1 + k_1\tau], \quad \mathbf{x}(t; t_1, x_1, d, \pi_{i_1}) \notin B.$

We can then iterate this procedure from the new state

$$x_2 = \mathbf{x}(t_1 + k_1\tau; t_1, x_1, d, \pi_{i_1}) \in R.$$

This can be repeated infinitely, yielding a sequence of points belonging to R $x_0, x_1, x_2, \ldots$ attained at times $t_0, t_1, t_2, \ldots$, when the patterns $\pi_{i_0}, \pi_{i_1}, \pi_{i_2}, \ldots$ are applied.

We furthermore have that all the trajectories stay in S and never cross B:

$$\forall t \in \mathbb{R}^+, \exists k \geq 0, \ t \in [t_k, t_{k+1}]$$

and

$$\forall t \in [t_k, t_{k+1}], \ \mathbf{x}(t; t_k, x_k, d, \pi_{i_k}) \in S, \ \mathbf{x}(t; t_k, x_k, d, \pi_{i_k}) \notin B.$$

The trajectories thus return infinitely often in R, while always staying in S and never crossing B. □

Remark 3 Note that it is possible to perform reachability from a set R_1 to another set R_2 by computing $Decomposition(R_1, R_2, S, B, D, K)$. The set R_1 is thus decomposed with the objective to send its sub-boxes into R_2, i.e., for a sub-box V of R_1, patterns π are searched with the objective $Post_\pi(V) \subseteq R_2$ (see Example 5.2).

4.2 The search of patterns

We propose in this paper an improvement of the function $Find\_Pattern$ given in [2,20], which is a naive testing of all the patterns of growing length (up to K).

The improved function, denoted here by $Find\_Pattern2$, exploits heuristics to prune the search tree of patterns. The algorithmic form of $Find\_Pattern2$ is given in Algorithm 3. It relies on a new data structure consisting of a list of triplets containing:

Algorithm 2 *Find_Pattern*.

Function: $Find\_Pattern(W, R, S, B, K)$

Input: A box W, a box R, a box S, a box B, a length K of input pattern
Output: $\langle \pi, True \rangle$ or $\langle \_, False \rangle$

for $i = 1 \dots K$ **do**
  $\Pi :=$ set of input patterns of length i
  while Π is non empty **do**
    Select π in Π
    $\Pi := \Pi \setminus \{\pi\}$
    if $Post_\pi(W) \subseteq R$ **and** $Tube_\pi(W) \subseteq S$ **and** $Tube_\pi(W) \bigcap B = \emptyset$ **then**
      return $\langle \pi, True \rangle$
    end if
  end while
end for
return $\langle \_, False \rangle$

Algorithm 3 *Find_Pattern2*.

Function: $Find\_Pattern2(W, R, S, B, K)$

Input: A box W, a box R, a box S, a box B, a length K of input pattern
Output: $\langle \pi, True \rangle$ or $\langle \_, False \rangle$

$\mathcal{S} = \{\emptyset\}$
$\mathcal{L} = \{(W, W, \emptyset)\}$
while $\mathcal{L} \neq \emptyset$ **do**
  e_{current} = takeHead($\mathcal{L}$)
  for $i \in U$ **do**
    if $Post_i(e_{\text{current}}.Y_{\text{current}}) \subseteq R$ **and** $Tube_i(e_{\text{current}}.Y_{\text{current}}) \bigcap B = \emptyset$ **and** $Tube_i(e_{\text{current}}.Y_{\text{current}}) \subseteq S$ **then**
      putTail($\mathcal{S}$, $e_{\text{current}}.\Pi + i$) /* or also "**return** $\langle e_{\text{current}}.\Pi + i, True \rangle$" */
    else
      if $Tube_i(e_{\text{current}}.Y_{\text{current}}) \bigcap B \neq \emptyset$ **or** $Tube_i(e_{\text{current}}.Y_{\text{current}}) \not\subseteq S$ **then**
        discard e_{current}
      end if
    else
      if $Tube_i(e_{\text{current}}.Y_{\text{current}}) \bigcap B = \emptyset$ **and** $Tube_i(e_{\text{current}}.Y_{\text{current}}) \subseteq S$ **then**
        if Length(Π) $+ 1 < K$ **then**
          putTail($\mathcal{L}$, $(e_{\text{current}}.Y_{\text{init}}, Post_i(e_{\text{current}}.Y_{\text{current}}), e_{\text{current}}.\Pi + i)$)
        end if
      end if
    end if
  end for
end while
return $\langle \_, False \rangle$ if no solution is found, or $\langle \pi, True \rangle$, π being any pattern validated in *Solution*.

- An initial box $V \subset \mathbb{R}^n$,
- A *current* box $Post_\pi(V)$, image of V by the pattern π,
- The associated pattern π.

For any element e of a list of this type, we denote by $e.Y_{\text{init}}$ the initial box, $e.Y_{\text{current}}$ the *current* box, and by $e.\Pi$ the associated pattern. We denote by $e_{\text{current}} = takeHead(\mathcal{L})$ the element on top of a list $\mathcal{L}$ (this element is removed from list $\mathcal{L}$). The function $putTail(\cdot, \mathcal{L})$ adds an element at the end of the list $\mathcal{L}$.

Let us suppose one wants to control a box $X \subseteq R$. The list $\mathcal{L}$ of Algorithm 3 is used to store the intermediate computations leading to possible solutions (patterns sending X in R while never crossing B or $\mathbb{R}^n \setminus S$). It is initialized as $\mathcal{L} = \{(X, X, \emptyset)\}$. First, a testing of all the control modes is performed (a set simulation starting from X during time τ is computed for all the modes in U). The first level of branches is thus tested exhaustively. If a branch leads to crossing B or $\mathbb{R}^n \setminus S$, the branch is cut. Indeed, no following branch can be accepted if a previous one crosses B. It is one of the improvements presented in this paper. Otherwise, either a solution is found or an intermediate state is added to $\mathcal{L}$. The next level of branches (patterns of length 2) is then explored from branches that are not cut. This process continues iteratively. At the end, either the tree is explored up to level K (avoiding the cut branches), or all the branches have been cut at lower levels. List $\mathcal{L}$ is thus of the form $\{(X, Post_{\pi_i}(X), \pi_i)_{i \in I_X}\}$, where for each $i \in I_X$ we have $Post_{\pi_i}(X) \subseteq S$ and $Tube_{\pi_i}(X) \bigcap B = \emptyset$. Here, I_X is the set of indexes associated to the stored intermediate solutions, $|I_X|$ is thus the number of stored intermediate solutions for the initial box X. The number of stored intermediate solutions grows as the search tree of patterns is explored, then decreases as solutions are validated, branches are cut, or the maximal level K is reached.

The storage of the intermediate solutions $Post_{\pi_i}(X)$ allows us to reuse the computations already performed. Even if the search tree of patterns is visited exhaustively, it already allows us to obtain much better computation times than with Function $Find\_Pattern$.

A second list, denoted by $\mathcal{S}$ in Algorithm 3, is used to store the validated patterns associated to X, i.e., a list of patterns of the form $\{\pi_j\}_{j \in I'_X}$, where for each $j \in I'_X$ we have $Post_{\pi_j}(X) \subseteq R$, $Tube_{\pi_j}(X) \bigcap B = \emptyset$ and $Tube_{\pi_j}(X) \subseteq S$. Here, I'_X is the set of indexes associated the the stored validated solutions, $|I'_X|$ is thus the number of stored validated solutions for the initial box X. The number of stored validated solutions can only increase, and we hope that at least one solution is found, otherwise, the initial box X is split in two sub-boxes.

Remark 4 Several solutions can be returned by $Find\_Pattern2$, so further optimizations could be performed, such as returning the pattern minimizing a given cost function. In practice, and in the examples given below, we return the first validated pattern and stop the computation as soon as it is obtained (see commented line 12 in Algorithm 3).

Compared to [2], this new function highly improves the computation times, even though the complexity of the two functions is theoretically the same, at most in $O(N^K)$. A comparison between functions $Find\_Pattern$ and $Find\_Pattern2$ is given in Sect. 6.

5 Experimentations

In this section, we apply our approach to different case studies taken from the literature. Our solver prototype is written in C++ and based on DynIBEX [28]. The computations times given in the following have been performed on a 2.80 GHz Intel Core i7-4810MQ CPU with 8 GB of memory. Note that our algorithm is mono-threaded so all the experimentation only uses one core to perform the computations. The results given in this section have been obtained with Function $Find\_Pattern2$ with regards to our previous algorithm $Find\_Pattern$ [20]. We compare our approach with two tools in the state of the art of control synthesis: PESSOA [29] and SCOTS [30]. Note that such a comparison is necessarily somehow unfair: in our approach, we allow the trajectories starting at R to temporarily exit R (as far as they stay within S) before reentering R; in contrast, in PESSOA and SCOTS, the trajectories are strictly forbidden to exit R, even temporarily. This explains why, in the examples below, our approach may be

able to control the whole region R while PESSOA and SCOTS can only control subregions of R.

5.1 A linear example: boost DC–DC converter

This linear example (without disturbance) is taken from [31] and has already been treated with the state space bisection method in a linear framework in [2]. This running example is used to verify that our approach is still valid for a simple linear case without disturbance, and also to show the strong improvement in term of computation time.

The system is a boost DC–DC converter with one switching cell. There are two switching modes depending on the position of the switching cell. The dynamics is given by the equation $\dot{x}(t) = A_{\sigma(t)}x(t) + B_{\sigma(t)}$ with $\sigma(t) \in U = \{1, 2\}$. The two modes are given by the matrices:

$$A_1 = \begin{pmatrix} -\frac{r_l}{x_l} & 0 \\ 0 & -\frac{1}{x_c}\frac{1}{r_0+r_c} \end{pmatrix}, \quad B_1 = \begin{pmatrix} \frac{v_s}{x_l} \\ 0 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} -\frac{1}{x_l}\left(r_l + \frac{r_0.r_c}{r_0+r_c}\right) & -\frac{1}{x_l}\frac{r_0}{r_0+r_c} \\ \frac{1}{x_c}\frac{r_0}{r_0+r_c} & -\frac{1}{x_c}\frac{r_0}{r_0+r_c} \end{pmatrix}, \quad B_2 = \begin{pmatrix} \frac{v_s}{x_l} \\ 0 \end{pmatrix}.$$

with $x_c = 70$, $x_l = 3$, $r_c = 0.005$, $r_l = 0.05$, $r_0 = 1$, $v_s = 1$. The sampling period is $\tau = 0.5$. The parameters are exact and there is no perturbation. We want the state to return infinitely often to the region R, set here to $[1.55, 2.15] \times [1.0, 1.4]$, while never going out of the safety set $S = [1.54, 2.16] \times [0.99, 1.41]$. The goal of this example is then to synthesize a controller with intrinsic stability.

The decomposition was obtained in less than 1 s with a maximum length of pattern set to $K = 6$ and a maximum bisection depth of $D = 3$. A simulation is given in Fig. 4.

5.2 A polynomial example

We consider the polynomial system taken from [32], presented as a difficult example:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -x_2 - 1.5x_1 - 0.5x_1^3 + u_1 + d_1 \\ x_1 + u_2 + d_2 \end{bmatrix}. \tag{9}$$

The control inputs are given by $u = (u_1, u_2) = K_{\sigma(t)}(x_1, x_2), \sigma(t) \in U = \{1, 2, 3, 4\}$, which correspond to four different state feedback controllers $K_1(x) = (0, -x_2^2 + 2)$, $K_2(x) = (0, -x_2)$, $K_3(x) = (2, 10)$, $K_4(x) = (-1.5, 10)$. We thus have four switching modes. The disturbance $d = (d_1, d_2)$ lies in $[-0.005, 0.005] \times [-0.005, 0.005]$. The objective is to visit infinitely often two zones R_1 and R_2, without going out of a safety zone S, and while never crossing a forbidden zone B. Two decompositions are performed:

- a decomposition of R_1 which returns $\{(V_i, \pi_i)\}_{i \in I_1}$ with:
  - $\bigcup_{i \in I_1} V_i = R_1$,
  - $\forall i \in I_1,\ Post_{\pi_i}(V_i) \subseteq R_2$,
  - $\forall i \in I_1,\ Tube_{\pi_i}(V_i) \subseteq S$,
  - $\forall i \in I_1,\ Tube_{\pi_i}(V_i) \bigcap B = \emptyset$.
- a decomposition of R_2 which returns $\{(V_i, \pi_i)\}_{i \in I_2}$ with:
  - $\bigcup_{i \in I_2} V_i = R_2$,
  - $\forall i \in I_2,\ Post_{\pi_i}(V_i) \subseteq R_1$,
  - $\forall i \in I_2,\ Tube_{\pi_i}(V_i) \subseteq S$,

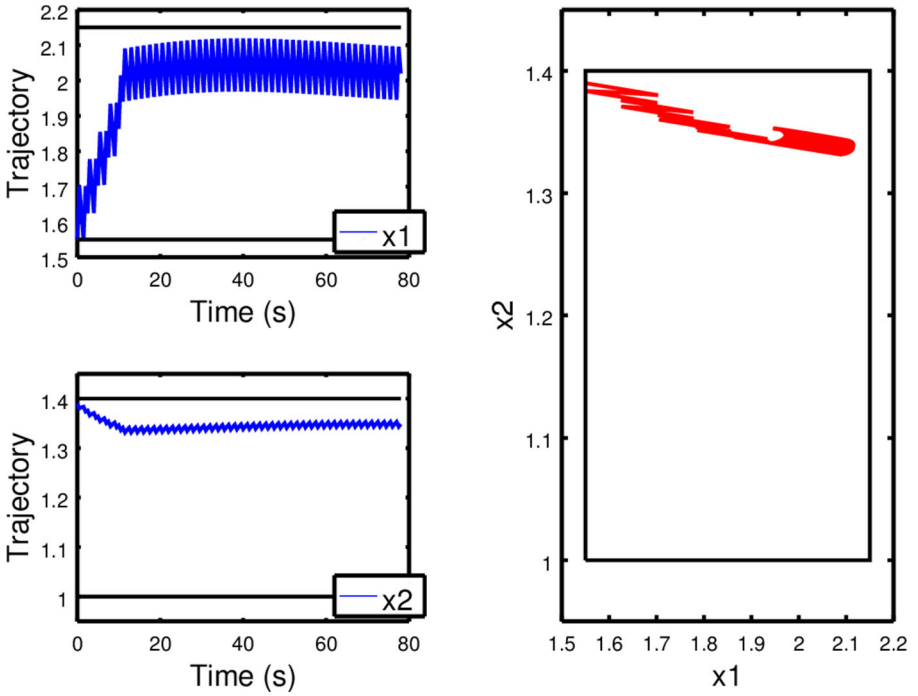

Fig. 4 Example 5.1: simulation from the initial condition (1.55, 1.4). The box R is in plain black. The trajectory is plotted within time for the two state variables on the left, and in the state space plane on the right

– $\forall i \in I_2,\ Tube_{\pi_i}(V_i) \bigcap B = \emptyset$.

The input boxes are the following:

$$\begin{aligned} R_1 &= [-0.5, 0.5] \times [-0.75, 0.0], \\ R_2 &= [-1.0, 0.65] \times [0.75, 1.75], \\ S &= [-2.0, 2.0] \times [-1.5, 3.0], \\ B &= [0.1, 1.0] \times [0.15, 0.5]. \end{aligned}$$

The sampling period is set to $\tau = 0.15$. The decompositions were obtained in 2 min and 30 s with a maximum length of pattern set to $K = 12$ and a maximum bisection depth of $D = 5$. A simulation is given in Fig. 5 in which the disturbance d is chosen randomly in $[-0.005, 0.005] \times [-0.005, 0.005]$ at every time step.

5.3 Building ventilation

We consider a building ventilation application adapted from [33]. The system is a four room apartment subject to heat transfer between the rooms, with the external environment, with the underfloor, and with human beings. The dynamics of the system is given by the following equation:

$$\frac{dT_i}{dt} = \sum_{j \in \mathcal{N}^* \setminus \{i\}} a_{ij}(T_j - T_i) + \delta_{s_i} b_i (T_{s_i}^4 - T_i^4) + c_i \max\left(0, \frac{V_i - V_i^*}{\bar{V}_i - V_i^*}\right)(T_u - T_i).$$

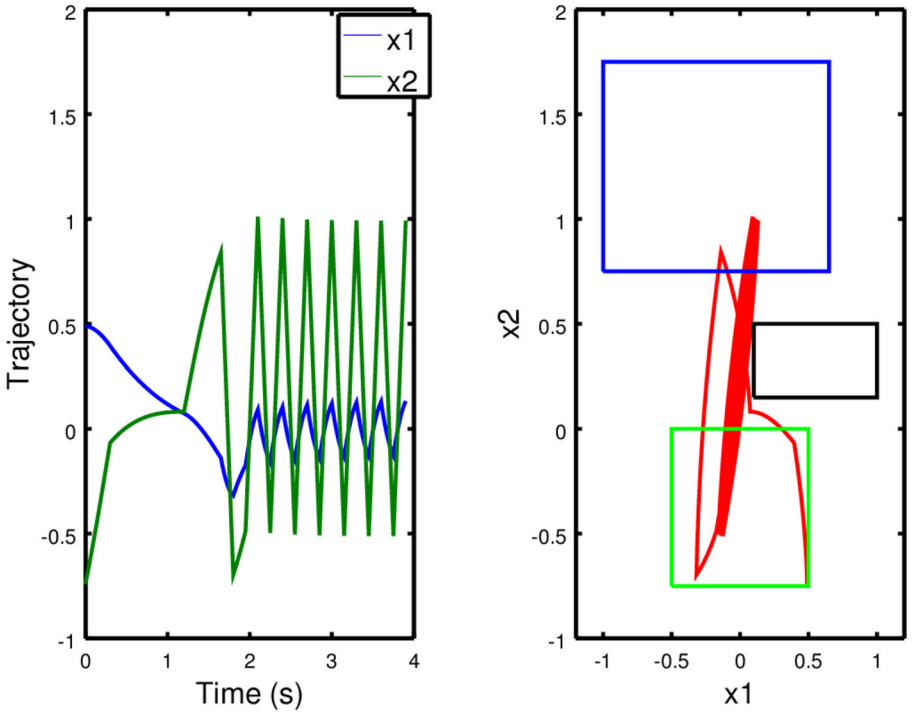

Fig. 5 Example 5.2: simulation from the initial condition $(0.5, -0.75)$. The trajectory is plotted within time on the left, and in the state space plane on the right. In the sate space plane, the set R_1 is in plain green, R_2 in plain blue, and B in plain black. (Color figure online)

The state of the system is given by the temperatures in the rooms T_i, for $i \in \mathcal{N} = \{1, \ldots, 4\}$. Room i is subject to heat exchange with different entities stated by the indexes $\mathcal{N}^* = \{1, 2, 3, 4, u, o, c\}$.

The heat transfer between the rooms is given by the coefficients a_{ij} for $i, j \in \mathcal{N}^2$, and the different perturbations are the following:

- The external environment: it has an effect on room i with the coefficient a_{io} and the outside temperature T_o, varying between 27 and 30 °C.
- The heat transfer through the ceiling: it has an effect on room i with the coefficient a_{ic} and the ceiling temperature T_c, varying between 27 and 30 °C.
- The heat transfer with the underfloor: it is given by the coefficient a_{iu} and the underfloor temperature T_u, set to 17 °C (T_u is constant, regulated by a PID controller).
- The perturbation induced by the presence of humans: it is given in room i by the term $\delta_{s_i} b_i (T_{s_i}^4 - T_i^4)$, the parameter δ_{s_i} is equal to 1 when someone is present in room i, 0 otherwise, and T_{s_i} is a given identified parameter.

The control $V_i, i \in \mathcal{N}$, is applied through the term $c_i \max(0, \frac{V_i - V_i^*}{\bar{V}_i - V_i^*})(T_u - T_i)$. A voltage V_i is applied to force ventilation from the underfloor to room i, and the command of an underfloor fan is subject to a dry friction. Because we work in a switched control framework, V_i can take only discrete values, which removes the problem of dealing with a "max" function in interval analysis. In the experiment, V_1 and V_4 can take the values 0 or 3.5 V, and V_2 and V_3 can take the values 0 or 3 V. This leads to a system of the form of Eq. (1) with $\sigma(t) \in U = \{1, \ldots, 16\}$,

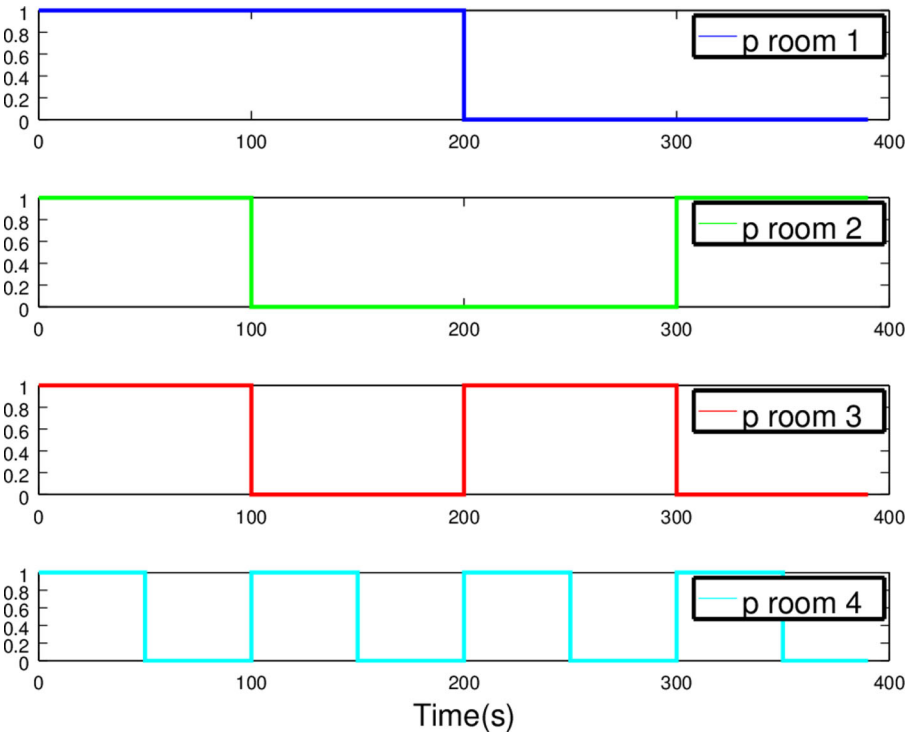

Fig. 6 Example 5.3: perturbation (presence of humans) imposed within time in the different rooms

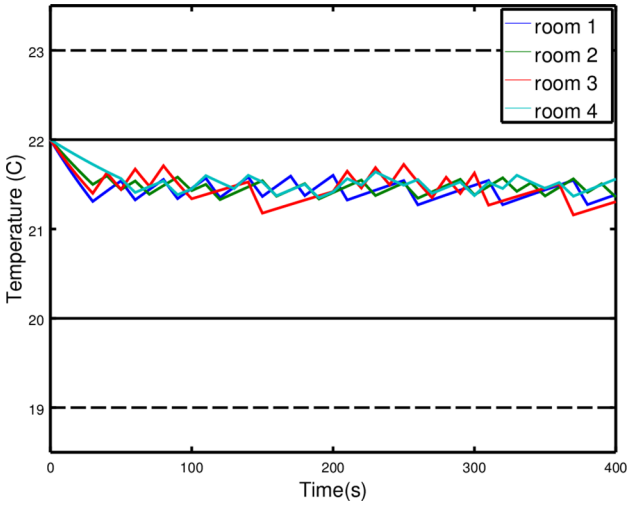

Fig. 7 Example 5.3: simulation from the initial condition (22, 22, 22, 22). The objective set R is in plain black and the safety set S is in dotted black

the 16 switching modes corresponding to the different possible combinations of voltages V_i. The sampling period is $\tau = 10$s.

The parameters T_{s_i}, V_i^*, $\bar{V}_i$, a_{ij}, b_i, c_i are given in [33] and have been identified with a proper identification procedure detailed in [34]. Note that here we have neglected the term $\sum_{j\in\mathcal{N}} \delta_{d_{ij}} c_{i,j} * h(T_j - T_i)$ of [33], representing the perturbation induced by the open or closed state of the doors between the rooms. Taking a "max" function into account with interval analysis is actually still a difficult task. However, this term could have been taken into account with a proper regularization (smoothing).

The main difficulty of this example is the large number of modes in the switched system, which induces a combinatorial issue.

The decomposition was obtained in 4 min with a maximum length of pattern set to $K = 2$ and a maximum bisection depth of $D = 4$. The perturbation due to human beings has been taken into account by setting the parameters δ_{s_i} equal to the whole interval $[0, 1]$ for the decomposition, and the imposed perturbation for the simulation is given Fig. 6. The temperatures T_o and T_c have been set to the interval $[27, 30]$ for the decomposition, and are set to 30 °C for the simulation. A simulation of the controller obtained with the state-space bisection procedure is given in Fig. 7, where the control objective is to stabilize the temperature in $[20, 22]^4$ while never going out of $[19, 23]^4$.

5.4 A path planning problem

This last case study is based on a model of a vehicle initially introduced in [35] and successfully controlled in [18,19] with the tools PESSOA and SCOTS. In this model, the motion of the front and rear pairs of wheels are approximated by a single front wheel and a single rear wheel. The dynamics if the vehicle is given by:

$$\dot{x} = v_0 \frac{\cos(\alpha + \theta)}{\cos(\alpha)}, \qquad \dot{y} = v_0 \frac{\sin(\alpha + \theta)}{\cos(\alpha)}, \qquad \dot{\theta} = \frac{v_0}{b} \tan(\delta), \tag{10}$$

where $\alpha = \arctan(a \tan(\delta)/b)$. The system is thus of dimension 3, (x, y) is the position of the vehicle, while θ is the orientation of the vehicle. The control inputs are v_0, an input velocity, and δ, the steering angle of the rear wheel. The parameters are: $a = 0.5$, $b = 1$. Just as in [18,19], we suppose that the control inputs are piecewise constant, which leads to a switched system of the form of Eq. (1) with no perturbation. The objective is to send the vehicle into an objective region $R_2 = [9, 9.5] \times [0, 0.5] \times]-\infty, +\infty[$ from an initial region $R_1 = [0, 0.5] \times [0, 0.5] \times [0, 0]$. The safety set is $S = [0, 10] \times [0, 10] \times]-\infty, +\infty[$. There is in fact no particular constraint on the orientation of the vehicle, but multiple obstacles are imposed for the two first dimensions, they are represented in Fig. 8. The input velocity v_0 can take the values in $\{-0.5, 0.5, 1.0\}$. The rear wheel orientation δ can take the values in $\{0.9, 0.6, 0.5, 0.3, 0.0, -0.3, -0.5, -0.6, -0.9\}$. The sampling period is $\tau = 0.3$.

Note that for this case study we used an automated pre-tiling of the state space permitting to decompose the reachability problem in a sequence of reachability problems. Using patterns of length up to $K = 10$, we managed to successfully control the system in 3619 s. In this case, the pattern is computed until almost the end without bisection as shown in Fig. 8. To obtain the last steps, the box is bissected in four ones by Algorithm 1. After that, patterns are found for the four boxes:

- $[8.43, 8.69]; [2.52, 2.78] : \{7000166\}$
- $[8.43, 8.69]; [2.78, 3.03] : \{7000256\}$
- $[8.69, 8.94]; [2.52, 2.78] : \{00055\}$
- $[8.69, 8.94]; [2.78, 3.03] : \{000265\}$

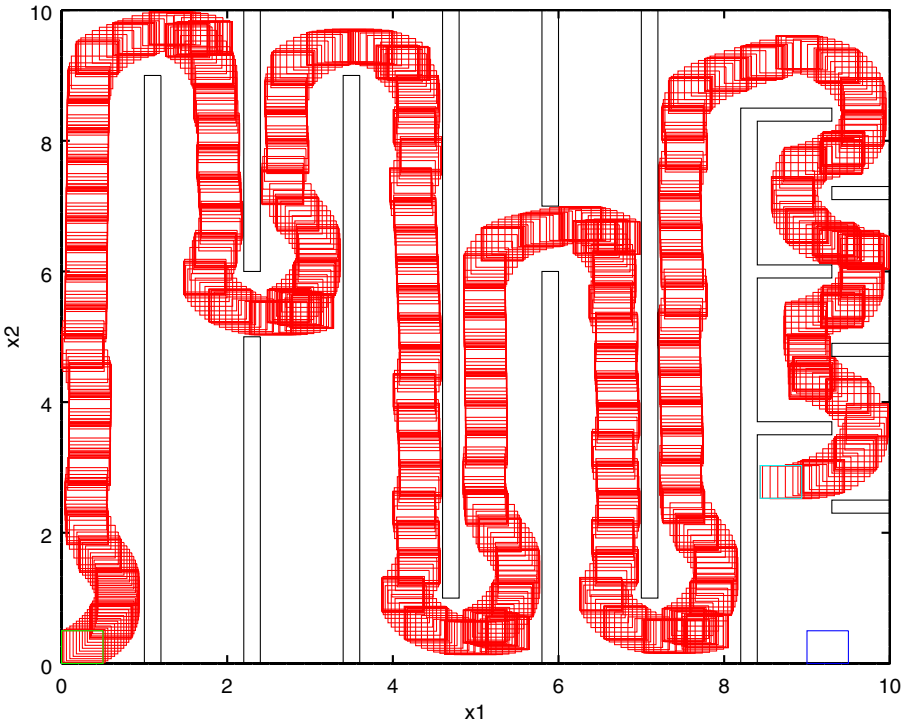

Fig. 8 Example 5.4: set simulation of the path planning example. The green box is the initial region R_1, the blue box is the target region R_2. The union of the red boxes is the reachability tube. In this case, the target region is not attained without bisection. (Color figure online)

The four set simulations obtained for the last steps are given in Fig. 9.

6 Performance tests

We present a comparison of functions $Find\_Pattern$, $Find\_Pattern2$ w.r.t. the computation times obtained, and with the state-of-the-art tools PESSOA [29] and SCOTS [30].

Table 1 shows a comparison of functions $Find\_Pattern$ and $Find\_Pattern2$, which shows that the new version highly improves computation time. We can note that the new version is all the more efficient as the length of the patterns increases, and as obstacles cut the research tree of patterns. This is why we observe significant improvements on the examples of the DC–DC converter and the polynomial example, and not on the building ventilation example, which only requires patterns of length 2, and presents no obstacle.

Table 2 shows of comparison of function $Find\_Pattern2$ with state-of-the-art tools SCOTS and PESSOA. On the example of the DC–DC converter, our algorithm manages to control the whole state space $R = [1.55, 2.15] \times [1.0, 1.4]$ in less than 1 s, while SCOTS and PESSOA only control a part of R, and with greater computation times. Note that these computation times vary with the number of discretization points used in both, but even with a very fine discretization, we never managed to control the whole box R. For the polynomial example, we manage to control the whole boxes R_1 and R_2, as with SCOTS and

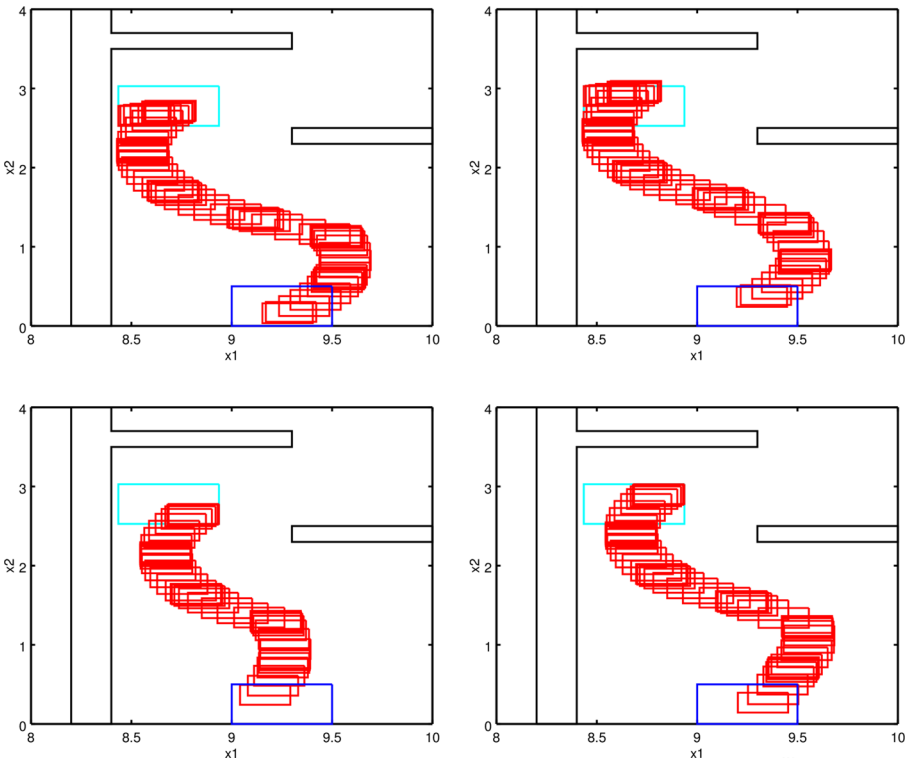

Fig. 9 Example 5.4: set simulation of the path planning example after bisection. The green boxes are the initial regions obtained by bisection, the blue box is the target region R_2. The union of the red boxes is the reachability tube. (Color figure online)

Table 1 Comparison of *Find_Pattern* and *Find_Pattern*2

Example	Computation time	
	Find_Pattern (s)	*Find_Pattern*2 (s)
DC–DC converter	1609	< 1
Polynomial example	Time out	150
Building ventilation	272	228
Path planning	Time out	3619

in a comparable amount of time. However, PESSOA does not support natively this kind of nonlinear systems. For the path planning case study, on which PESSOA and SCOTS perform well, we have not obtained computations times as good as they have. This comes from the fact that this example requires a high number of switched modes, long patterns, as well as a high number of boxes to tile the state space. This is in fact the most difficult application case of our method. This reveals that our method is more adapted when either the number of switched modes of the length of patterns is not high (though it can be handled at the cost of high computation times). Another advantage is that we do not require a homogeneous discretization of the state space. We can thus tile large parts of the state space using only few

Table 2 Comparison with state-of-the-art tools

Example	Computation time		
	FP2 (s)	SCOTS (s)	PESSOA (s)
DC–DC converter	< 1	43	760
Polynomial example	150	131	–
Path planning	3619	492	516

boxes, and this often permits to consider much fewer states than with discretization methods, especially in higher dimensions (see [36]).

7 Conclusion

We presented a method of control synthesis for nonlinear switched systems, based on a simple state-space bisection algorithm, and on guaranteed integration. The approach permits to deal with stability, reachability, safety and forbidden region constraints. Varying parameters and perturbations can be easily taken into account with interval analysis. The approach has been numerically validated on several examples taken from the literature, a linear one with constant parameters, and two nonlinear ones with varying perturbations. Our approach compares well with the state-of-the art tools SCOTS and PESSOA.

We would like to point out that the exponential complexity of the algorithms presented here, which is inherent to guaranteed methods, is not prohibitive. Two approaches have indeed been developed to overcome this exponential complexity. A first approach is the use of compositionality, which permits to split the system in two (or more) sub-systems, and to perform control synthesis on these sub-systems of lower dimensions. This approach has been successfully applied in [36] to a system of dimension 11, and we are currently working on applying this approach to the more general context of contract-based design [37]. A second approach is the use of Model Order Reduction, which allows to approximate the full-order system (1) with a reduced-order system, of lower dimension, on which it is possible to perform control synthesis. The bounding of the trajectory errors between the full-order and the reduced-order systems can be taken into account, so that the induced controller is guaranteed. This approach, described in [38], has been successfully applied on (space-discretized) partial differential equations, leading to systems of ODEs of dimension up to 100,000. The present work is a potential ground for the application of such methods to control nonlinear partial differential equations, with the use of proper nonlinear model order reduction techniques.

References

1. Alur R, Courcoubetis C, Henzinger TA, Ho P-H (1993) Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In: Proceedings of hybrid systems I, number 736 in LNCS, Springer, pp 209–229
2. Fribourg L, Kühne U, Soulat R (2014) Finite controlled invariants for sampled switched systems. Form Methods Syst Des 45(3):303–329
3. Liberzon D (2012) Switching in systems and control. Springer, London
4. Girard A (2005) Reachability of uncertain linear systems using zonotopes. In: Proceedings of hybrid systems: computation and control, volume 3414 of LNCS, Springer, pp 291–305
5. Althoff M (2013) Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In: ACM proceedings of hybrid systems: computation and control, pp 173–182

6. Chen X, Ábrahám E, Sankaranarayanan S (2013) Flow*: an analyzer for non-linear hybrid systems. In: Proceedings of computer aided verification, volume 8044 of LNCS, Springer, pp 258–263
7. Frehse G, Guernic CL, Donzé A, Cotton S, Ray R, Lebeltel O, Ripado R, Girard A, Dang T, Maler O (2011) SpaceEx: scalable verification of hybrid systems. In: Proceedings of computer aided verification, volume 6806 of LNCS, Springer, pp 379–395
8. Moore R (1966) Interval analysis. Prentice Hall, Englewood Cliffs
9. Nedialkov NS, R KJ, Corliss GF (1999) Validated solutions of initial value problems for ordinary differential equations. Appl Math Comput 105(1):21–68
10. Lin Y, Stadtherr MA (2007) Validated solutions of initial value problems for parametric ODEs. Appl Numer Math 57(10):1145–1162
11. Dzetkulič T (2015) Rigorous integration of non-linear ordinary differential equations in Chebyshev basis. Numer Algorithms 69(1):183–205
12. Bouissou O, Martel M (2006) GRKLib: a guaranteed Runge Kutta library. In: IEEE Proceedings of scientific computing, computer arithmetic and validated numerics
13. Gajda K, Jankowska M, Marciniak A, Szyszka B (2008) A survey of interval Runge–Kutta and multistep methods for solving the initial value problem. In: Proceedings of parallel processing and applied mathematics, volume 4967 of LNCS, Springer, pp 1361–1371
14. Bouissou O, Chapoutot A, Djoudi A (2013) Enclosing temporal evolution of dynamical systems using numerical methods. In: Proceedings of NASA formal methods, number 7871 in LNCS, Springer, pp 108–123
15. Alexandre dit Sandretto J, Chapoutot A (2016) Validated explicit and implicit Runge–Kutta methods. Reliab Comput 22
16. Henzinger TA, Horowitz B, Majumdar R, Wong-Toi H (2000) Beyond HYTECH: hybrid systems analysis using interval numerical methods. In: Proceedings of hybrid systems: computational and control, volume 1790 of LNCS, Springer, pp 130–144
17. Girard A, Pola G, Tabuada P (2010) Approximately bisimilar symbolic models for incrementally stable switched systems. IEEE Trans Autom Control 55(1):116–126
18. Zamani M, Pola G, Mazo M, Tabuada P (2012) Symbolic models for nonlinear control systems without stability assumptions. IEEE Trans Autom Control 57(7):1804–1809
19. Reissig G, Weber A, Rungger M (2017) Feedback refinement relations for the synthesis of symbolic controllers. IEEE Trans Autom Control 62(4):1781–1796
20. Le Coënt A, Alexandre dit Sandretto J, Chapoutot A, Fribourg L (2016) Control of nonlinear switched systems based on validated simulation. In: IEEE Proceedings of symbolic and numerical methods for reachability analysis, pp 1–6
21. Ravanbakhsh H, Sankaranarayanan S (2016) Robust controller synthesis of switched systems using counterexample guided framework. In: ACM/IEEE proceedings of conference on embedded software, pp 1–10
22. Jaulin L, Kieffer M, Didrit O, Walter E (2001) Applied interval analysis. Springer, London
23. Hairer E, Norsett SP, Wanner G (2009) Solving ordinary differential equations I: nonstiff problems, 2nd edn. Springer, Berlin
24. de Figueiredo LH, Stolfi J (1997) Self-validated numerical methods and applications. Brazilian mathematics colloquium monographs. IMPA/CNPq, Rio de Janeiro
25. Alexandre dit Sandretto J, Chapoutot A (2016) Validated explicit and implicit Runge–Kutta methods. Reliab Comput 22:79–103
26. Lohner RJ (1987) Enclosing the solutions of ordinary initial and boundary value problems. In: Computer arithmetic, pp 255–286
27. Kapela T, Zgliczynski P (2009) A Lohner-type algorithm for control systems and ordinary differential inclusions. Discrete Contin Dyn Syst Ser B 11(2):365–385
28. Alexandre dit Sandretto J, Chapoutot A (2015) DynIBEX library. http://perso.ensta-paristech.fr/~chapoutot/dynibex/
29. Mazo M, Davitian A, Tabuada P (2010) PESSOA: a tool for embedded controller synthesis. In: Proceedings of computer aided verification, volume 6174 of LNCS, Springer, pp 566–569
30. Rungger M, Zamani M (2016) SCOTS: a tool for the synthesis of symbolic controllers. In: ACM Proceedings of hybrid systems: computation and control, pp 99–104
31. Beccuti AG, Papafotiou G, Morari M (2005) Optimal control of the boost DC–DC converter. In: IEEE Proceedings of conference on decision and control, pp 4457–4462
32. Liu J, Ozay N, Topcu U, Murray RM (2013) Synthesis of reactive switching protocols from temporal logic specifications. IEEE Trans Autom Control 58(7):1771–1785
33. Meyer P-J (2015) Invariance and symbolic control of cooperative systems for temperature regulation in intelligent buildings. Theses, Université Grenoble Alpes

34. Meyer P-J, Nazarpour H, Girard A, Witrant E (2014) Experimental implementation of UFAD regulation based on robust controlled invariance. In: IEEE Proceedings of European control conference, pp 1468–1473
35. Aström KJ, Murray RM (2010) Feedback systems: an introduction for scientists and engineers. Princeton University Press, Princeton
36. Le Coënt A, Fribourg L, Markey N, de Vuyst F, Chamoin L (2016) Distributed synthesis of state-dependent switching control. In: Proceedings of reachability problems, volume 9899 of LNCS, Springer, pp 119–133
37. Sangiovanni-Vincentelli A, Damm W, Passerone R (2012) Taming Dr. Frankenstein: contract-based design for cyber-physical systems. Eur J Control 18(3):217–238
38. Le Coënt A, de Vuyst F, Rey C, Chamoin L, Fribourg L (2016) Control of mechanical systems using set based methods. Int J Dyn Control 1–17