Hai Lin, X.D. Koutsoukos and P.J. Antsaklis, "HySTAR: A Toolbox for Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems," Proceedings of the 2002 American Control Conference, pp.686-691, Anchorage, Alaska, May 8-10, 2002.

HYSTAR: A ToolBox for Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems[1]

Hai Lin[2]
Department of EE
University of Notre Dame,
Notre Dame, IN 46556 USA

Xenofon D. Koutsoukos
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA

Panos J. Antsaklis
Department of EE
University of Notre Dame,
Notre Dame, IN 46556 USA

Abstract

This paper describes a Matlab toolbox for computational analysis and hierarchical controller synthesis of piecewise linear hybrid dynamical systems. The analysis and design are based on computation of predecessor operator and backward reachability analysis. Both static specifications that do not change as time progresses and dynamic specifications that include sequencing of events and eventual execution of actions are considered. Control design is implemented using finite automata and linear programming techniques. A temperature control system is used for illustration through out the paper.

1 Introduction

This paper describes a Matlab toolbox for computational analysis and hierarchical controller synthesis of piecewise linear hybrid dynamical systems. The continuous dynamics are described by linear difference equations, the discrete dynamics by finite automata, and the interaction between the continuous and the discrete part is defined by piecewise linear maps. Piecewise linear systems arise very often as mathematical models for practical applications, see for example [14, 4] and references therein. In [3], a Matlab toolbox, PwLTool, for analysis of piecewise linear systems is described, which is based on piecewise quadratic Lyapunov functions and convex optimization. In [2], a toolbox, `d/dt`, is developed for analyzing hybrid systems with linear differential inclusions, and the reachable states are calculated by polyhedral approximations. For further reference about existing toolboxes for hybrid systems, see for example the review paper [12, 1].

The Matlab toolbox, HYSTAR (HY*), is an implementation of our group's recent theoretic results in the field of *Hierarchical Control of Hybrid Dynamical Systems* (cf. [6, 7]). The approach presented is directly related to supervisory control framework for hybrid systems (cf. [8, 15, 5]). The control objective is for the closed loop system to follow a desired output which is assumed to be generated by another dynamical system. This framework leads naturally to an input-output representation of the constituent system which is analogous to classical control design. The main questions are first the existence of appropriate control resources, and if the controller exists, then how to design it. The methods implemented in HY* answer both of these questions.

The structure of this paper is as follows. Section 2 describes the model representation, i.e. how a piecewise linear hybrid dynamical system is defined in this toolbox. Section 3 considers backward reachability analysis, which it the basic tool being used throughout the toolbox. Section 4 deals with control specifications. This is the analysis part. Section 5 is the controller synthesis part. Then we conclude with some discussion about future work.

2 Model Representation

The toolbox handles Piecewise Linear Hybrid Dynamical Systems (PLHDS) of the form:

$$x(t+1) = A_{q(t)}x(t) + B_{q(t)}u(t) + E_{q(t)}d(t) \quad (2.1)$$
$$q(t+1) = \delta(q(t), \pi(x(t)), \sigma_c(t), \sigma_u(t)) \quad (2.2)$$

where $\pi : X \to X/E_\pi$ partitions the continuous state space $X \subseteq \mathbb{R}^n$ into a number of closed polyhedral cells (possibly unbounded), and Q stands for the collection of discrete states(modes) $Q = \{q\}$; $u \in \mathcal{U} \subset \mathbb{R}^m$, $d \in \mathcal{D} \subset \mathbb{R}^p$, and $\mathcal{U}$, $\mathcal{D}$ are bounded polyhedral set; $q(t+1) \in act(\pi(x(t)))$, $act : X/E_\pi \to 2^Q$ defines the active mode set; $A_q \in \mathbb{R}^{n\times n}, B_q \in \mathbb{R}^{n\times m}$, and $E_q \in \mathbb{R}^{n\times p}$ are the system matrices for the discrete state q; $\delta : Q \times X/E_\pi \times \Sigma_c \times \Sigma_u \to Q$ is the discrete state transition function, here Σ_c and Σ_u denote the collection of controllable and uncontrollable events respectively. The *guard* $G(q,q')$ of the transition (q,q_0) is defined as the set of all states (q,x) such that $q' \in act(\pi(x(t)))$ and there exist controllable event $\sigma_c \in \Sigma_c$ such that $q' = \delta(q, \pi(x), \sigma_c, \sigma_u)$ for every uncontrollable event $\sigma_u \in \Sigma_u$. The guard of the transition describes the region of the hybrid state space where the transition can be forced to take place independently of the disturbances generated by the environment.

The piecewise linear cells $X_i, i = 1, ..., |\pi|$ can be represented

[1]The partial support of the National Science Foundation (NSF ECS99-12458 & CCR01-13131), and of the DARPA/ITO-NEST Program (AF-F30602-01-2-0526) is gratefully acknowledged.

[2]Corresponding author. `hlin1@nd.edu`.

Hai Lin, X.D. Koutsoukos and P.J. Antsaklis, "HySTAR: A Toolbox for Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems," Proceedings of the 2002 American Control Conference, pp.686-691, Anchorage, Alaska, May 8-10, 2002.

command	description
setplhds	initialize PLHDS object
addregion	define polyhedral region
addynamics	define system dynamics
addcontrol	specify the control U
addisturbance	specify the disturbance D
addguard	specify the Guard for discrete transition
getplhds	extract PLHDS object

Table 1: Commands for building a PLHDS system.

by matrices (G_i, w_i) that satisfy $G_i x \leq w_i \Leftrightarrow x \in X_i$[1]. Here, the vector inequality $z \leq 0$ means that each entry of z is non-positive. We recognize this as the half-space representation of a polyhedron. Also we use the similar half-space representation to specify the boundary, a bounded polyhedron, of control and disturbance.

Table 1 lists the basic commands for building a PLHDS. Having partitioned the state space and used the functions for entering data into Matlab, the system is aggregated into a single record that is passed on to functions for analysis and controller synthesis. The command `setplhds` initializes the PLHDS object and should be run first. When this is done, one will typically define the entire system by repeatedly calling `addynamics` and `addregion`. The command `addynamics` is used to specify the matrix A_q, B_q and E_q corresponding to the dynamics of a certain discrete mode q of the PLHDS system. An identifier is returned for future reference to the dynamics, also we use the returned identifier to stand for the discrete states $q \in Q$, i.e. each discrete state $q \in Q$ corresponding to a three-tuple dynamic matrixes (A_q, B_q, E_q). The command `addregion` lets the user enter the region specific data (G_i, w_i) and via the references returned by `addynamics` specify the dynamics in the region. The command `addcontrol` and `addisturb` is used to specify the boundary of the continuous control u and disturbance d respectively. `Addguard` command specify the guard region of a pair discrete transition (q, q') or $q \rightarrow q'$. When all matrices are entered, the PLHDS object is extracted by `getplhds`. Please note that in addition to linking several dynamics to one region, it is also possible to link several regions to the same dynamics. To illustrate the idea, we present a temperature control system (c.f. [6, 7]) to illustrate the piecewise linear hybrid system model.

Example 2.1 (TEMPERATURE CONTROL SYSTEM (TCS)**)** The system consists of a furnace that can be switched on and off. The control objective is to control the temperature at a point of the system by applying the heat input at a different point. So, the discrete mode only contains two states, that is the furnace "off", q_0, and the furnace is "on", q_1. The continuous dynamics are described as [2]

$$x(t+1) = \begin{cases} A_0 x(t) + B_0 u(t) + E_0 d(t), & q = q_0 \\ A_1 x(t) + B_1 u(t) + E_1 d(t), & q = q_1. \end{cases}$$

where

$$A_0 = \begin{pmatrix} 0.8259 & 0.1354 \\ 0.0677 & 0.5551 \end{pmatrix}, B_0 = \begin{pmatrix} 1.8179 \\ 0.0773 \end{pmatrix}, E_0 = \begin{pmatrix} 0.0387 \\ 0.3772 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} -0.6634 & 0.1997 \\ 0.1997 & 0.2641 \end{pmatrix}, B_1 = \begin{pmatrix} 0.8101 \\ 0.1369 \end{pmatrix}, E_1 = \begin{pmatrix} 0.1369 \\ 0.5363 \end{pmatrix}$$

The partition of the state space is obtained by considering the following hyperplanes:

$$\begin{aligned} h_1(x) &= x_1 - M, \ M = 20 \\ h_2(x) &= x_2 - h_t, \ h_t = 5 \\ h_3(x) &= x_2 - l_t, \ l_t = 0 \\ h_4(x) &= x_1 \end{aligned}$$

The following lines of code define the "Temperature Control System" in question.

```
%Initialize the PLHDS object
setplhds([]);
%Enter Dynamical Matrix
A0=[.8259 .1354;.0677 .5551]; B0=...; A1=... ;
dyn1=addynamics(A0,B0,E0);
dyn2=addynamics(A1,B1,E1);
% Add Region
g1=[1,0];w1=20;g2=[0,1];w2=5;
g3=[0,1];w3=0; g4=[1,0];w4=0;
addregion([g1;g2;-g3],[w1;w2;w3],[dyn1,dyn2]);
addregion(...
 ... ...
%Add Guard
allspaceG=[0 0]; allspacew = 1;
addguard(allspaceG,allspacew,dyn1,dyn1);
addguard(...
 ... ...
% Add Control
Gu=[1;-1];wu=[1;-.5]; addcontrol(Gu,wu);
%Add Disturb
Gd=[1;-1];wd=[.1;-.1]; addisturb(Gd,wd);
%Extract PLHDS object
plhds = getplhds;
```

3 Backward Reachability Analysis

The main mathematical tool to be used for backward reachability analysis is the *predecessor operator* applied recursively to subsets of the hybrid state space. The application of the predecessor operator corresponds to partition refinement into finer partitions that allow the formulation of conditions that guarantee the existence of appropriate controls for the objectives of interest.

3.1 The Predecessor Operator

A region of the state space is defined as $R \subset Q \times X$. Let's assume $R = (q, P)$ where $q \in Q$ and $P \subset \mathbb{R}^n$ is a piecewise linear set. We are interested in computing the set of all the

[1] Such a description is based on the fact that piecewise-linear algebra admits elimination of quantifiers [13].

[2] using zero-order hold sampling with $T = 1$s.

Hai Lin, X.D. Koutsoukos and P.J. Antsaklis, "HySTAR: A Toolbox for Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems," Proceedings of the 2002 American Control Conference, pp.686-691, Anchorage, Alaska, May 8-10, 2002.

states that can be driven to R by either continuous or discrete transitions.

Discrete Transitions

The predecessor operator for discrete transitions is denoted by $pre_d: 2^{Q\times X} \to 2^{Q\times X}$ and it is used to compute the set of states that can be driven to the region R by a discrete instantaneous transition $q' \to q$ that can be forced by the controller for any uncontrollable event. The predecessor operator in this case is defined as follows:

$$pre_d(R) = \{(q',x) \in Q\times X | \exists \sigma_c \in \Sigma_c, \forall \sigma_u \in \Sigma_u, q = \delta(q',x,\sigma_c,\sigma_u)\}$$

For every discrete transition that can be forced by a controllable event we have that

$$pre_d(R) = \bigcup_{q' \in act(P)} G(q',q)$$

where $G(q',q)$ is the guard of transition $q' \to q$.

Continuous Transitions

In the case of continuous transitions, given the region $R = (q,P)$ we define the predecessor operator $pre_c: 2^{Q\times X} \to 2^{Q\times X}$ to compute the set of states for which there exists a control input so that the continuous state will be driven in the set P for every disturbance, while the system is at the discrete mode q. The action of the operator is described by

$$pre_c(R) = \{q\} \times \{x \in X | \exists u \in U, \forall d \in D, A_q x + B_q u + E_q d \in P\}$$

The set $pre_c(R)$ can be computed in closed form by elimination of quantifiers ($\exists, \forall$). This is accomplished efficiently by Fourier-Motzkin elimination and linear programming techniques [10] and has been implemented in HY*.

3.2 Algorithms for Backward Reachability Analysis

Consider a PLHDS and a region $R = (q,P)$. We denote the partition of the continuous state space X as $\{Pi\}$, $i = 1,...|\pi|$. In addition, let $pre_{c,q}: 2^X \to 2^X$ denote the predecessor operator for a continuous transition described by the discrete mode q, and let $proj(q\times X) = X$, i.e. the projection into continuous state space. The following algorithm computes all the states of the hybrid system that can be driven to R in one time-step. The algorithm is implemented by using the technical results presented in [6].

Algorithm for the computation of $pre(R)$

INPUT R=(q,P), S=$\emptyset$;
for i=1,...,$|\pi|$,
  $Q_i = P \cap P_i$
  if $Q_i \neq \emptyset$
    for $q' \in act(P_i)$
      $S_i^{q'} = proj(G(q,q')) \cap Q_i$
      if $S_i^{q'} \neq \emptyset$
        S = $S \cup (\{q'\} \times pre_{c,q'}(S_i^{q'}))$
      end if
    end
  end if
end
OUTPUT: pre(R)= S

Remark 3.1 *We have shown that the set $pre(R)$ is piecewise linear and is described using a finite set of linear inequalities. Therefore, we can apply the predecessor operator to compute the set of all states that can be driven to $pre(R)$ to get $pre(pre(R))$. Following the same procedure, we define successive applications of the predecessor operator as:*

$$pre^N(R) = \underbrace{pre(...pre(R))}_{Ntimes} \tag{3.1}$$

Remark 3.2 *For a given region R,we define the coreachable set $CR(R)$ as the set of all states that can be driven to R. The coreachable set for a region of the hybrid state space can be computed by successive application of the predecessor operator*

$$CR(R) = pre^*(R) \tag{3.2}$$

3.3 Commands for Backward Reachability Analysis

Table 2 lists the commands for predecessor operator. Command `prec` calculates the one step backward reachable set of a piecewise linear set only by continuous input no matter what the bounded disturbance is. The command `pre` exactly follows the algorithm in 3.2, which calculate the one step backward reachable set by both possible continuous control and discrete transitions. Also, the backward reachable property are reserved on the face of continuous disturbance $d \in D$ and discrete uncontrollable event (disturbance) $\sigma_u \in \Sigma_u$. Finally let's see an example for usage of the commands.

command	description
prec	predecessor for continuous transition
pre	predecessor operator

Table 2: Commands for Predecessor Operator.

Example 3.1 (TCS - PREDECESSOR OPERATOR) We still consider the temperature control example described in the previous section. Consider region $R = (\{q_0,q_1\},P)$, where $P = \{x \in \mathbb{R}^2 | (0 \le x_1 \le 20) \wedge (0 \le x_2 \le 5)\}$. First we define the region we are interested in then we call `prec` to calculate the predecessor set by the following code.

```
%define Region R
GR=[g1;g2;-g3;-g4]; wR=[w1;w2;w3;w4];
%continuous predecessor operator
preRc1 = prec(dyn1,GR,wR,plhds);
viewpart(preRc1.l,preRc1.r)
%viewpart is built to draw polyhedral sets.
preRc2 = prec(dyn2,GR,wR,plhds);
viewpart(preRc2.l,preRc2.r)
```

The predecessor set is also piecewise linear, as shown in Figure 1.

Hai Lin, X.D. Koutsoukos and P.J. Antsaklis, "HySTAR: A Toolbox for Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems," Proceedings of the 2002 American Control Conference, pp.686-691, Anchorage, Alaska, May 8-10, 2002.

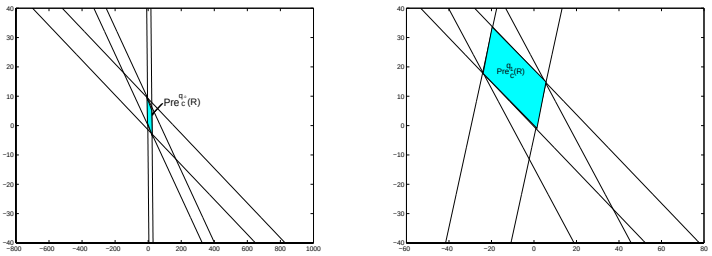

Figure 1: *The continuous predecessor operator of the temperature control system example*

command	description
issafe	check the safety
isreach1	check N-reachability between two regions
isreach2	check reachability by grid-based approximation
coreach1	N-coreach set calculation
coreach2	coreach set calculation by grid-based approx.
gridv	grid vertex generation (used in coreach2)
inplset	check whether point list is within a PwL set

Table 3: Commands for Static Specification Analysis.

4 Specifications and Analysis

Typical control specifications investigated in this paper are formulated in terms of partitions of the state space of the system. Examples include safety problems, where the controller guarantees that the plant will not enter an unsafe region. Control specification can be divided into *static specification* and *dynamic specifications. Static specifications* describe desired outputs that do not change as time progresses. For example, safety and reachability are static specifications. *Dynamic specifications* involve sequencing of events and eventual execution of actions.

4.1 Static Control Specification

At first, we focus on the safety problem and we show how the refinement of the state space partition can be used to formulated conditions for safety.

Safety

Given a set of states described by the region $R \subset Q \times X$ and an initial condition $(q_0, x_0) \in R$,we say that the system is *safe* if $(q(t), x(t)) \in R$ for every $t \geq t_0$. The conditions[6, 7] that guarantee that a given region of the hybrid state space is safe can be described as follows. A PLHDS is safe with respect to the region $R \subseteq Q \times X$ if and only if $R \subseteq pre(R)$. This safety condition can be efficiently tested by solving a finite number of linear programming problems that depends on the number of regions and discrete states of the system.

Reachability

It should be emphasized that we are interested only in the case when reachability between two regions R_1 and R_2 is defined so that the state is driven to R_2 directly from the region R_1 without entering a third region. This is a problem of practical importance in hybrid systems. The problem of deciding if a region R_2 is *directly reachable* from R_1 can be solved by recursively computing all the states that can be driven to R_2 from R_1 using the predecessor operator.That is to check whether $R_1 \subseteq CR(R_2)$ or not. [3]

In general, the proposed procedure is semi-decidable and its termination is not guaranteed. In order to formulate a constructive algorithm for reachability, we consider two approaches in the toolbox. First, we consider an upper bound on the time horizon and we examine the reachability only for the predetermined finite horizon, i.e. $CR^N(R_2)$. Second, we formulate a termination condition for the reachability algorithm based on a grid-based approximation of the piecewise linear regions of the state space.

Commands for Static Specification Analysis

Table 3 lists the Commands for Static Specification Analysis. Command `issafe` check whether there exists available control $u \in U$ and $\sigma_c \in \Sigma_c$ such that for any possible disturbance $d \in D$ and $\sigma_u \in \Sigma_u$, the piecewise linear region in question remains safety. Command `coreach1` calculate the N step backward reachable set, that is the set $CR^N(R_2)$. Command `isreach1` check the N-reachability problem between R_1 and R_2. The command `coreach2` calculate the coreachable set $CR(R_2)$ using the grid based approximation. The grid used in `coreach2` is generated by function `gridv`, and the function `inplset` is used for checking whether there exist a "substantial difference" between each iteration. Command `isreach2` check the reachability problem correspondingly.

4.2 Dynamic Specification and Attainability

In general, a regulator requests certain types of outputs from the plant so that these are attained in the presence of disturbances. The desired outputs can be described as the outputs of another dynamical system, called the *exosystem*[11]. We model the control specifications using an input-output deterministic finite automaton. The dynamic behavior of the exosystem is described by the set of output sequences it can generate. For formal definition of exosystem, please turn to [7, 11].

Attainability

Our control objective is that the closed loop system consisting of the plant and the controller exhibits the same behavior as the exosystem . We consider specifications that are described with respect to regions of the hybrid state space. We may describe the specification as a sequence of regions, $\{R_1, R_2, ..., R_n\}$ where $R_i = (Q_i, P_i)$ are piecewise linear regions of the hybrid state space.

The main question is if there exists a controller so that the closed loop system follows the behavior of the exosystem, B_{sp}. This question is directly related to the existence of appropriate control resources in order for the plant to achieve the desired behavior. We formalize this notion using the *attainability* of the specified behavior. In this work, attainable

[3] We only consider regions of the form $R_1 = (Q_1, P_1)$ and $R_2 = (Q_2, P_2)$ for which P_1 and P_2 are adjacent polyhedral regions of the primary partition.

Hai Lin, X.D. Koutsoukos and P.J. Antsaklis, "HySTAR: A Toolbox for Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems," Proceedings of the 2002 American Control Conference, pp.686-691, Anchorage, Alaska, May 8-10, 2002.

command	description
setspec	initialize the specification object
addspec	add specification
getspec	extract the specification object
isattain	attainability checking

Table 4: Commands for define Specification and Attainability checking.

command	description
safereg	design regulator to guarantee safety
reachreg	design regulator to guarantee reachability
regulator	Regulator design
plhdsim	simulation for given PLHDS

Table 5: Commands for Regulator Synthesis and Simulation.

behavior refers to behavior that can be forced to the plant by a control mechanism. In the following we present the necessary and sufficient condition for attainability [7].

Theorem 4.1 *The specification behavior B_{sp} is attainable if and only if the following conditions hold:*

1. *Every terminating state y_n corresponds to a region R_n that is safe; and*
2. *For every non-terminating state y_k, there exists y_{k+1} so that, for the corresponding regions we have that R_{k+1} is reachable from R_k*

Furthermore, if B_{sp} is attainable then there exists a controller $\mathcal{C}$ so that the regulator problem has a solution.

Specification Setting

Table 4 lists the Commands for setting specifications, including static specification and dynamic specification as well. Command `setspec` initializes a standard specification object. Then, by using command `addspec`, we can add the reachable regions. Please note the regions in specification contains two parts, one is the discrete states and the other is the continuous piecewise linear set. The regions defined in specification do not necessary exactly coincide with the regions defined in PLHDS object. `getspec` extract the specification data.

Attainability Checking

`isattain` check whether there exists available control $u \in U$ and $\sigma_c \in \Sigma_c$ such that for any possible disturbance $d \in D$ and $\sigma_u \in \Sigma_u$, the specification in question remains attainable. By Theorem 4.2, we know attainability checking can be divided into checking the safety for the terminal region and the reachability between two successive regions. So `isattain` call the function `isreach1` and the `issafe` (See Table 3 for reference). Finally let's see an example for usage of the commands.

Example 4.1 (TCS - SPECIFICATION) Consider region $R_1 = (\{q_0, q_1\}, P_1)$ and $R_2 = (\{q_0, q_1\}, P_2)$, where $P_1 = \{x \in \mathbb{R}^2 | (0 \le x_1 \le 20) \wedge (-20 \le x_2 \le 0)\}$, and $P_2 = \{x \in \mathbb{R}^2 | (0 \le x_1 \le 20) \wedge (0 \le x_2 \le 5)\}$. It is desirable that every sate from region R_1 can be driven to R_2 without entering a third region, then the state will stay inside R_2. The following code set the above simple specification first, then checking its attainability.

```
%Initialize Specification object
setspec(plhds,[])
%Add Regions(specifications)
addspec(plhds,[dyn1],[-g4],[-w4],1);
addspec(plhds,...);
  ...
%Extract the specification object
spec=getspec;
%Attainability Checking
attain =isattain(spec, plhds);
if attain>0
    display('the attainability check passed.')
else ...
```

5 Hierarchical Controller Synthesis

In this section, we illustrate the control design functionalities of our toolbox. We have implemented a systematic procedure for controller synthesis based on a formulation of the control regulator problem and the proposed notion of *attainability*. It is assumed that the desired behavior is attainable and therefore there exists a control policy so that the plant will follow the output of the exosystem. Our objective is to build a convenient representation of the controller. The design of the controller is based on the regions $\{R_1, ..., R_n\}$ that are used to define the control specifications.

The controller consists of three agents [7]. The *event generator* receives the discrete-time measurement signal of the hybrid plant, and issues appropriate events when the state $(q(t), x(t))$ enters a new region R_i of the hybrid state space. The *control automaton* is a finite automaton whose states correspond to the regions R_i and its main purpose is to select an appropriate cost functional based on the control objective. Finally, *the actuator* determines the control input which is applied to the hybrid plant. The control input consists of a continuous component $u \in U$ and a discrete component $\sigma_c \in \Sigma_c$ which triggers feasible discrete transitions.

Commands for Regulator Design and Simulation

Table 5 lists the Commands for Regulator Synthesis and Simulation. Command `safereg` first call function `issafe` to check whether there exists available control $u \in U$ and $\sigma_c \in \Sigma_c$ such that for any possible disturbance $d \in D$ and $\sigma_u \in \Sigma_u$, the piecewise linear region in question remains safety. If the checking passed, then `safereg` calculate and return the appropriate control $u \in U$ and $\sigma_c \in \Sigma_c$. Similarly, Command `reachreg` first check the reachability. If the checking passed, then `reachreg` calculate and return the appropriate control $u \in U$ and $\sigma_c \in \Sigma_c$.

`regulator` calls function `reachreg` to calculate the ap-

Hai Lin, X.D. Koutsoukos and P.J. Antsaklis, "HySTAR: A Toolbox for Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems," Proceedings of the 2002 American Control Conference, pp.686-691, Anchorage, Alaska, May 8-10, 2002.

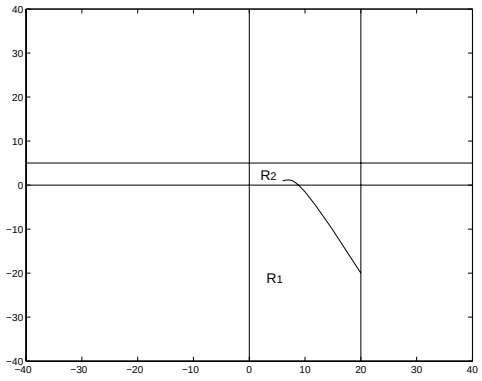

Figure 2: *The simulation for regulator of the PLHDS*

propriate control $u \in U$ and $\sigma_c \in \Sigma_c$ to satisfy the reachability between two nonterminal successive regions in the specification. And when the final region (terminal) reaches, `regulator` call `safereg` to calculate the appropriate control $u \in U$ and $\sigma_c \in \Sigma_c$ in order to guarantee safety specification. At each step the `regulator` also call `plhdsim` for simulation of the trajectory.
Finally let's see an example for usage of the commands.

Example 5.1 (TCS-REGULATOR DESIGN) We also consider the temperature control systems. In the previous section, we have defined a specification and checked its attainability. Here we will define the regulator to satisfy the specification. At last we call `plhdsim` to simulation the closed loop PLHDS. The following code satisfy this purpose, and the simulation is shown in Figure 2. It should be noted that this behavior can be attained for any initial condition in the region R_1 and for any disturbance provided the behavior is attainable.

```
x0=[20;-20];
[u,Reg,qss,xss] = regulator( plhds, spec, x0);
figure(1),clc
viewpart(spec(2).l,spec(2).r), hold on
plot(xss(1,:),xss(2,:)),hold off
```

6 Conclusion

This paper has presented a Matlab toolbox, HY*, for analysis and hierarchical control synthesis of piecewise linear hybrid dynamical systems. The analysis is based on computation of predecessor operator and backward reachability analysis. In the future, we will extend the class of piecewise linear hybrid dynamical systems to include parameter uncertainty, for theoretical results please see [9]. The toolbox is available from the authors upon request.

References

[1] P. Antsaklis and X. Koutsoukos. *Hybrid Dynamical Systems: Review and Recent Progress*. Chapter in Software-Enabled Control: Information Technologies for Dynamical Systems, T. Samad and G. Balas, Eds., IEEE Press. To appear in 2002.

[2] E. Asarin, O. Bournez, T. Dang, and O. Maler *Approximate Reachability Analysis of Piecewise Linear Dynamical Systems*. In Hybrid Systems: Computation and Control, March 2000, Pittsburgh, USA.

[3] S. Hedlund and M. Johansson. *A toolbox for computational analysis of piecewise linear systems*. In Proceedings of European Control Conference, Karlsruhe, 1999.

[4] M. Johansson, *Piecewise Linear Control Systems*, Ph.D. Thesis, Lund Institute of Technology, Sweden, 1999.

[5] X. Koutsoukos, P. Antsaklis, J. Stiver, and M. Lemmon. *Supervisory control of hybrid systems*. Proceedings of IEEE, 88(7):10261049, July 2000.

[6] X.D. Koutsoukos and P.J. Antsaklis,*Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems Based on Discrete Abstractions*, ISIS Technical Report ISIS-2001-001, February 2001 (`http://www.nd.edu/ isis/tech.html`).

[7] X.D. Koutsoukos and P.J. Antsaklis, *Hierarchical Design of Piecewise Linear Hybrid Dynamical Systems Using a Control Regulator Approach*, ISIS Technical Report ISIS-2001-004, June 2001. (`http://www.nd.edu/ isis/tech.html`).

[8] M. Lemmon, K. He, and I. Markovsky. *Supervisory hybrid systems*. Control Systems Magazine, 19(4):42-55, August 1999.

[9] H. Lin X.D. Koutsoukos and P.J. Antsaklis, *Hierarchical Control of a class Uncertain Piecewise Linear Hybrid Dynamical Systems*, to appear at IFAC'2002, Barcelona, Spain, July, 2002.

[10] T. Motzkin *The theory of linear inequalities*. Rand Corp., Santa Monica, CA, 1952.

[11] M. Sain. *Introduction to Algebraic System Theory*. Academic Press, 1981.

[12] B. Silva, O. Stursberg, B. Krogh, S. Engell. *An assessment of the current status of algorithmic approaches to the verification of hybrid systems* the 40th Conference on Decision and Control, Vol. 3, pp.2867 -2874, 2001.

[13] E. Sontag. *Remarks on piecewise-linear algebra*. Pacific Journal of Mathematics, 92(1):183210, 1982.

[14] E. Sontag. *Interconnected automata and linear systems: A theoretical framework in discrete-time*. Hybrid Systems III, volume 1066 of Lecture Notes in Computer Science, pp 436448. Springer, 1996.

[15] J. Stiver. *Analysis and design of hybrid control systems*. PhD thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 1995.