ISIS Tech. Report - 2003-006

HyStar : A Matlab Toolbox for Robust Regulation of Polytopic Uncertain Piecewise Linear Hybrid Systems *

Hai Lin Panos J. Antsaklis

Department of Electrical Engineering,
University of Notre Dame, Notre Dame, IN 46556, USA
E-mail: {hlin1, antsaklis.1}@nd.edu

Abstract

This paper describes a Matlab toolbox for computational analysis and synthesis of a class of piecewise linear hybrid dynamical systems, which are affected by both time varying parametric uncertainties and persistent exterior disturbances. The robust tracking and regulation control problem for such uncertain piecewise linear systems is investigated and solved in two stages. The first stage is to analyze whether there exists an admissible control law such that the closed-loop system exhibits desired behavior. If a specified behavior can be forced on the plant by a control mechanism, then it is called attainable. The method for attainability checking is developed and implemented which employs the predecessor operator and backward reachability analysis. The second stage is to design such admissible control law for a given attainable tracking and regulation specification. The procedure for controller design proposed and implemented here is based on linear programming techniques. A numerical example is used for illustration throughout the paper.

*The partial support of the National Science Foundation (NSF ECS99-12458 & CCR01-13131) is gratefully acknowledged. The first author appreciates the support from Center of Applied Mathematics Fellowship (2003-04), University of Notre Dame.

Contents

List of Tables

List of Figures

1 Introduction

Hybrid Systems are heterogeneous dynamical systems of which the behavior is determined by interacting continuous variable and discrete event dynamics. Hybrid systems have been identified in a wide variety of applications in control of mechanical systems, process control, automotive industry, power systems, aircraft and traffic control, among many other fields. Therefore, the last decade has seen considerable research activities in modeling, analysis and synthesis of hybrid systems involving researchers from a number of traditionally distinct fields [6]. In this paper, we will develop a Matlab toolbox, HyStar, for robust analysis and synthesis for a class of uncertain hybrid systems.

There have been a variety of software toolboxes with varying methodologies and varying success in the hybrid literature. For example, HYSDEL [1] is an automated model representation tool for discrete-time piecewise linear hybrid systems, which generates MLD models to be used for control, estimation, and verification [8]. HyTech [2, 10] is an automatic verification tool for linear hybrid automata based on symbolic model checking. In [3, 7], a toolbox, `d/dt`, is developed for analyzing hybrid systems with linear differential inclusions, and the reachable states are calculated by polyhedral approximations. CheckMate [4], which is developed for automatic verification for nonlinear hybrid automata, also approximates the set of reachable states by polyhedra. In [9], a Matlab toolbox, PwLTool, for analysis of piecewise linear systems is described, which is based on piecewise quadratic Lyapunov functions and convex optimization. For further reference about existing toolboxes for hybrid systems, see for example the review paper [14].

The Matlab toolbox, HyStar (Hy*), is an implementation of our group's recent theoretic results in the field of *robust control of uncertain Hybrid Dynamical Systems* [12]. Previous work appeared in [11], which dealt with discrete-time piecewise linear hybrid systems without parametric uncertainties. The model investigated in the present paper is a class of discrete-time uncertain piecewise linear systems, which is affected by both time-variant parameter variations and persistent exterior disturbances. Section 2 describes the model representation, i.e. how an uncertain piecewise linear system is defined in this toolbox. The control objective is for the closed-loop system to exhibit certain desired behavior despite the uncertainties and disturbances. Specifically, given finite number of regions $\{\Omega_0,\ \Omega_1, \cdots,\ \Omega_M\}$ in the state space, our goal is for the closed-loop system trajectories, starting from the given initial region Ω_0, to go through the sequence of finite number of regions $\Omega_1,\ \Omega_2, \cdots,\ \Omega_M$ in the desired order and finally reach the final region Ω_M and then remain in Ω_M. This kind of specifications is analogous to the ordinary tracking and regulation problem in pure classical continuous variable dynamical control systems. In addition, it also reflects the qualitative ordering of event requirements along trajectories. In Section 3, the robust tracking and regulation control problem for such uncertain piecewise linear system is qualitatively formulated, and the commands to represent the specifications are

described. One of the main questions is to determine whether there exist admissible control laws such that the region-sequence can be followed. If such an admissible control law exists, the region-sequence specification $\{\Omega_0,\ \Omega_1, \cdots,\ \Omega_M\}$ is called *attainable*. The attainability checking is based on backward reachability analysis and symbolic model checking method, which are discussed and implemented in Section 4. In Section 5, the necessary and sufficient condition and corresponding commands for checking the attainability is given. The next question is how to design an admissible control law in order to satisfy the closed-loop specification. An optimization based method is proposed and implemented in Section 6 to design such admissible control laws. A numerical example is used for illustration throughout the paper. The command references for the toolbox is presented as an appendix to the paper. In addition, a software toolkit built for non-convex piecewise linear sets is briefly described in the appendix as well.

Notation : A polyhedron in $\mathbb{R}^n$ is a (convex) set given by the intersection of a finite number of open and/or closed half-spaces in $\mathbb{R}^n$. A polytope is a closed and bounded (i.e. compact) polyhedron. A polyhedral set $\mathcal{P}$ will be presented either by a set of linear inequalities $\mathcal{P} = \{x : F_i x \leq g_i,\ i = 1, \cdots, s\}$, synthetically $\mathcal{P} = \{x : Fx \leq g\}$, or by the dual representation in terms of its vertex set $\{x_j\}$, denoted by $vert\{\mathcal{P}\}$. A piecewise linear (Pwl) set consists of a finite union of polyhedra.

2 Model Representation

In this section, a mathematical definition of the discrete-time uncertain piecewise linear systems is given and the representation of the uncertain piecewise linear systems in HyStar (Hy*) is described.

2.1 Uncertain Piecewise Linear Systems

We consider discrete-time uncertain piecewise linear systems of the form

$$x(t+1) = A_q(w(t))x(t) + B_q(w(t))u(t) + E_q d(t), \quad t \in \mathbb{Z}^+, \text{ if } x \in \mathcal{P}_q \tag{2.1}$$

where $x(t) \in \mathcal{X} \subset \mathbb{R}^n$, $u(t) \in \mathcal{U}_q \subset \mathbb{R}^m$, $d(t) \in \mathcal{D}_q \subset \mathbb{R}^r$, $w(t) \in \mathcal{W} \subset \mathbb{R}^v$ are the system state, the control input, the disturbance input, and the uncertainty parameter respectively. Let the finite set Q stand for the collection of discrete modes q. It is assumed that $\mathcal{X}$, $\mathcal{U}_q$, $\mathcal{D}_q$ and $\mathcal{W}$ are assigned polytopes for each mode $q \in Q$, and that $\mathcal{D}_q$ contains the origin. The partition of the state space $\mathcal{X}$ is given as a finite set of polyhedra $\{\mathcal{P}_q : q \in Q\}$, where $\mathcal{P}_q \subseteq \mathcal{X}$ and $\bigcup_{q \in Q} \mathcal{P}_q = \mathcal{X}$. The continuous variable dynamics of each mode q is defined by the state matrices $A_q(w)$, $B_q(w)$ and E_q. It is assumed that the entries of $A_q(w)$ and $B_q(w)$ are continuous functions of w for every mode q.

command	description
`setupwl`	initialize uncertain piecewise linear system (UPWLsys) object
`addynamics`	define system dynamics and constraints
`addregion`	define the polyhedral regions $\mathcal{P}_q$
`getupwl`	extract UPWLsys object

Table 1: Commands for building a UPWL system.

A possible evolution of the uncertain piecewise linear systems from a given initial condition $x_0 \in \mathcal{X}$ can be described as follows. First, there exists at least one discrete mode $q_0 \in Q$ such that $x_0 \in \mathcal{P}_{q_0}$; the mode q_0 is then called feasible mode for state x_0 [1]. The next continuous variable state is given by the transition $x_1 = A_{q_0}(w)x_0 + B_{q_0}(w)u + E_{q_0}d$ for some $w \in \mathcal{W}$, $d \in \mathcal{D}_{q_0}$ and specific $u \in \mathcal{U}_{q_0}$. Then the above procedure is repeated for state x_1 to determine the next possible state x_2, and so on.

In practice uncertainties often enter linearly in the system model and they are linearly constrained. To handle this particular but interesting case, we consider the class of polyhedral sets. Such sets have been considered in previous works addressing the control of systems with input and state constraints. Their main advantage is that they are suitable for computation. Therefore, in the sequel, we turn to consider polytopic uncertainty in $A_q(w)$ and $B_q(w)$ for every mode $q \in Q$. Without loss of generality, we assume that

$$A_q(w) = \sum_{k=1}^{v_q} w_q^k A_q^k, \;\; B_q(w) = \sum_{k=1}^{v_q} w_q^k B_q^k,$$

where $w_q^k \geq 0$ and $\sum_{k=1}^{v_q} w_q^k = 1$. The pair $(A_q(w), B_q(w))$ represents the model uncertainty which belongs to the polytopic set $Conv\{(A_q^k, B_q^k),\; k = 1, \cdots, v_q\}$ for each mode $q \in Q$. This is referred to as polytopic uncertainty and provides a classical description of model uncertainty. Notice that the coefficients w_q^k are unknown and possibly time varying.

2.2 Represent Uncertain Piecewise Linear Systems

Table 1 lists the basic commands for building an uncertain piecewise linear system (UPWLsys). 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 `setupwl` initializes the UPWLsys 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 vertex state

[1] In the definition of uncertain piecewise linear systems, it is not required that the partition $\mathcal{P}_q$ has mutually empty intersections. Therefore, for the initial state x_0 there may exist more than one feasible discrete modes. In such cases, it is assumed that the current active mode, q_0, is randomly selected from these feasible modes.

matrices of the polytopic uncertain $A_q(w)$, $B_q(w)$ and E_q corresponding to the dynamics of a certain discrete mode q of the UPWL system. Command `addynamics` also specifies the boundary of the continuous control u, $\mathcal{U}_q$, and disturbance d, $\mathcal{D}_q$ for the discrete mode q. An identifier is returned for future reference to the dynamics, also we use the returned identifier to stand for the discrete states $q \in Q$. The command `addregion` lets the user enter the region specific data (F_q, g_q), where $\mathcal{P}_q = \{x : F_q x \leq g_q\}$, and via the references returned by `addynamics` specify the dynamics in the region. When all matrices are entered, the UPWLsys object is extracted by `getupwl`. Please note that in addition to linking several dynamics to one region, it is also possible to link several regions to the same dynamics. In the following, we present an numerical example to illustrate the uncertain piecewise linear system model.

Example 2.1 (Represent Model) We consider discrete-time uncertain piecewise linear systems of the form

$$x(t+1) = \begin{cases} A_1(w)x(t) + B_1(w)u(t) + E_1 d(t), & x \in \mathcal{P}_1 \\ A_2(w)x(t) + B_2(w)u(t) + E_2 d(t), & x \in \mathcal{P}_2. \end{cases}$$

where $\mathcal{P}_1 = \{x \in \mathbb{R}^2 | \|x\|_\infty \leq 100\}$ and $\mathcal{P}_2 = \{x \in \mathbb{R}^2 | \ -50 \leq x_1 \leq 100, \ -50 \leq x_2 \leq 100\}$. The vertex matrices of polytopic uncertain $A_1(w)$ and $B_1(w)$ are

$$\begin{aligned} A_1^1 &= \begin{pmatrix} 0.825 & 0.135 \\ 0.68 & 1 \end{pmatrix}, \ A_1^2 = \begin{pmatrix} 1 & 0.35 \\ 0.068 & 0.555 \end{pmatrix} \\ B_1^1 &= \begin{pmatrix} 1.7 \\ 0.06 \end{pmatrix}, \ B_1^2 = \begin{pmatrix} 1.9 \\ 0.08 \end{pmatrix}, \ E_1 = \begin{pmatrix} 0.0387 \\ 0.3772 \end{pmatrix}, \end{aligned}$$

and the vertex matrices of polytopic uncertain $A_2(w)$ and $B_2(w)$ are

$$\begin{aligned} A_2^1 &= \begin{pmatrix} -0.664 & 0.199 \\ 0.199 & 0.264 \end{pmatrix}, \ A_2^2 = \begin{pmatrix} -0.7 & 0.32 \\ 0.32 & 0.44 \end{pmatrix} \\ B_2^1 &= \begin{pmatrix} 0.8 \\ 0.1 \end{pmatrix}, \ B_2^2 = \begin{pmatrix} 0.9 \\ 0.2 \end{pmatrix}, \ E_2 = \begin{pmatrix} 0.1369 \\ 0.5363 \end{pmatrix} \end{aligned}$$

Assume the constrains of the continuous control signal is given as $\mathcal{U}_1 = \mathcal{U}_2 = [-1, 1]$, while the bound of disturbance is $d \in \mathcal{D}_1 = \mathcal{D}_2 = [-0.1, 0.1]$. The uncertain piecewise linear system model can be set up by the following codes.

```
%Initialize the UPWLSYS object
setupwl([]);
%Enter Dynamical Matrix
```

```
A0=[.825 .135;.067 .555];
A00=[1 .35;.68 1];
B0=[1.7;.06];
B00=[1.9; .08];
E0 =[.0387;.3772];

A1=[-.664 .199;.199 .264]; A11=...;
 ... ;
U.l = [1;-1]; U.r = [1;1];
D.l = [1;-1]; D.r =[.1;-.1];
% Add Dynamics
dyn1=addynamics({A0,A00},{B0,B00},E0,U,D,0);
dyn2=addynamics({A1,A11},{B1,B11},E1,U,D,0);
% Add Region
addregion([1 0;0 1;-1 0;0 -1],[100;100;100;100],[dyn1]);
addregion([1 0;0 1;-1 0;0 -1],[100;100;50;50],[dyn2]);
%Extract UPWLSYS object
upwlsys = getupwl;
```

Now, the variable `upwlsys` contains aggregated data of the uncertain piecewise linear system model, including state partition information, discrete modes and dynamics for each mode etc. In order to do analysis and synthesis for the above uncertain piecewise linear system, one simply passes the variable `upwlsys` on to functions for analysis and controller synthesis, which will be introduced later.

3 Problem Formulation

In this section, the robust tracking and regulation problem is formulated for the discrete-time polytopic uncertain piecewise linear systems and the representation of the specification in HyStar (Hy*) is described.

3.1 Robust Tracking and Regulation Problem

Tracking and regulation problem is a classical control problem in the continuous variable dynamical control systems, which can be formulated as follows. Given an initial region, a target region and a pipe connecting these two regions in the state space, design a controller such that all the trajectories starting from the initial region will be driven to the target region through the pipe. Similar specification can be formulated for the piecewise linear systems. However instead of connecting the initial region and the target region by a pipe, we use a

sequence of connected polyhedral regions Ω_i, which may be seen as inner approximations of the pipe specification (See Figure 1).

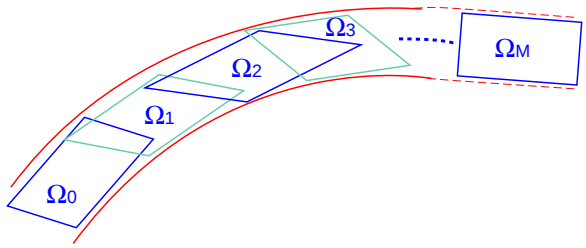

Figure 1: Tracking and regulation control specification and its polyhedral region sequence approximation.

The problem considered in this paper is to select feasible modes $q(t)$ and to design admissible control signals $u(t) \in \mathcal{U}_{q(t)}$ such that the closed-loop piecewise linear systems' trajectories, starting from the given initial region Ω_0, go through the sequence of regions $\Omega_1,\ \Omega_2, \cdots,\ \Omega_M$ in the desired order, finally reach the final region Ω_M and then remain in Ω_M, in spite of uncertainties and disturbances. Note that the region $\Omega_i \subseteq \mathcal{X}$ does not necessarily coincide with the partitions $\mathcal{P}_q$ in the definition of the uncertain piecewise linear systems (2.1).

3.2 Specification Setting

command	description
`setspec`	initialize the specification object
`addspec`	add specification
`getspec`	extract the specification object

Table 2: Commands for tracking and regulation specification setup.

The commands for setting specifications are listed in Table 2. Command `setspec` initializes a standard specification object. Then, by using the command `addspec`, we can add the regions Ω_i one by one (not necessarily in order). Command `getspec` extracts the specification data and returns a single record, which can be called for analysis and synthesis purpose later on. Let us illustrate the commands through an example.

Example 3.1 (Specification Setup) For the previous example, we consider two regions in sequence $\{\Omega_0, \Omega_1\}$, which is given as $\Omega_0 = \{x \in \mathbb{R}^2 |\ -20 \leq x_1 \leq 20,\ -40 \leq x_2 \leq 40\}$, and $\Omega_1 = \{x \in \mathbb{R}^2 | \|x\|_\infty \leq 10\}$. Our control objective is for all the initial states in the region Ω_0 will be driven by admissible control laws into region Ω_1 in finite number of steps

without existing Ω_0. When it reaches the region Ω_1, the state trajectories will stay in Ω_1 for ever. This specification can be described by the following lines.

```
%Initialize Specification object
setspec(upwlsys,[])
%Add Regions(specifications)
addspec(upwlsys,[1 0;0 1;-1 0;0 -1],[20;40;20;40],1);
addspec(upwlsys,[1 0;0 1;-1 0;0 -1],[10;10;10;10],2);
%Extract the specification object
spec = getspec;
```

3.3 Two Stage Solver

We propose to solve the robust tracking and regulation problems for the uncertain piecewise linear systems in two stages.

First stage is to check whether there exist feasible modes $q(t)$ and admissible control signals, $u(x(t)) \in \mathcal{U}_{q(t)}$, such that the region-sequence specification is satisfied despite the uncertainties and disturbances. If there exists such admissible control laws to satisfy the tracking and regulation specification, then the specification is called *attainable*. In order to check attainability, two different kinds of properties should be checked, that is the direct reachability between two successive regions, Ω_i and Ω_{i+1} for $1 \leq i < M$, and the safety (or controlled invariance) for the final region Ω_M. The analysis problems for safety and direct reachability are formulated as follows.

- Safety: Given a region $\Omega \subset \mathcal{X}$, determine whether there exist admissible control laws such that the evolution of the system starting from Ω remains inside the region for all time, despite the presence of dynamic uncertainties and disturbances.
- Direct Reachability: Given two regions $\Omega_1,\ \Omega_2 \subset \mathcal{X}$, determine whether there exist admissible control laws such that all states in Ω_1 can be driven into Ω_2 in finite steps without entering a third region.

After answering how to check whether there exists admissible control laws to satisfy a given specification, we will design the admissible control law in the second stage, if the specification is attainable. Similarly, the controller synthesis stage is also divided into two basic problems, that is safety control and direct reachability control. The two basic control problems are formulated as follows.

- Safety Controller Synthesis: Given a safe region $\Omega \subset \mathcal{X}$, determine the admissible control laws to make the region Ω safe (controlled invariant);

- Direct Reachability Controller Synthesis: Given two regions $\Omega_1,\ \Omega_2 \subset \mathcal{X}$, where Ω_2 is directly reachable from Ω_1, determine the admissible control laws such that all the states in Ω_1 can be driven into Ω_2 in finite steps without entering a third region.

In the following sections, we will solve the robust tracking and regulation problem stage by stage. First, necessary and sufficient conditions for safety and direct reachability are given in Section 5. The safety and direct reachability checking are based on backward reachability analysis. In the next section, we will briefly discuss the backward reachability analysis and its implementation, which serves as one of the basic tools for the analysis that follows.

4 Robust Backward Reachability Analysis

This section describes the results and commands for backwards reachability analysis for the uncertain piecewise linear systems, which server as the foundation for checking safety, reachability and attainability.

4.1 Robust One-Step Predecessor Set

The basic building block to be used for backward reachability analysis is the *robust one-step predecessor operator*, which is defined below.

Definition 4.1 The *robust one-step predecessor set*, $pre(\Omega)$, is the set of states in $\mathcal{X}$, for which admissible control inputs exist and drive these states into Ω in one step, despite disturbances and uncertainties, i.e.

$$\begin{aligned} pre(\Omega) = \{x(t) \in \mathcal{X} \ \ | \ \ & \exists q(t) \in Q,\ u(t) \in \mathcal{U}_{q(t)} \ : \ x(t) \in \mathcal{P}_{q(t)}, \\ & A_{q(t)}(w)x(t) + B_{q(t)}(w)u(t) + E_{q(t)}d(t) \in \Omega,\ \forall d(t) \in \mathcal{D}_{q(t)},\ w \in \mathcal{W}\} \end{aligned}$$

We can also define the one-step predecessor set under the q-th mode, $pre_q(\Omega)$, as the set of all states $x \in \mathcal{P}_q$, for which an admissible control input $u \in \mathcal{U}_q$ exists and guarantees that the system will be driven to Ω by the transformation $A_q(w)x + B_q(w)u + E_q d$ for all allowable disturbances and uncertainties.

Proposition 4.1 The robust one-step predecessor set $pre(\Omega)$ for an uncertain piecewise linear system can be computed as follows:

$$pre(\Omega) = \bigcup_{q \in Q} pre_q(\Omega)$$

Therefore, we only need to calculate the one-step predecessor set for each q-th subsystem.

4.2 Predecessor Sets for Subsystems

The difficulty in calculating $pre_q(\Omega)$ comes mainly from the fact that the region Ω is typically non-convex. Even if one starts with convex sets, the procedure deduces non-convex sets for piecewise linear systems after an one-step predecessor operation. Because of the non-convexity, some of the linearity and convexity arguments do not hold and extra care should be taken. However, under the polytopic uncertainty assumption, the calculation of the predecessor set for piecewise linear (Pwl) sets can be simplified, in view of the following proposition.

Proposition 4.2 For polytopic uncertain discrete-time piecewise linear system, the robust one-step predecessor set for an assigned piecewise linear set Ω (may be non-convex) under the q-th dynamics can be calculated as

$$pre_q(\Omega) = \bigcap_{k=1}^{v_q} pre_q^k(\Omega),$$

where $pre_q^k(\Omega)$ stands for the one-step predecessor operator of the k-th vertex state matrix (A_q^k, B_q^k) for $1 \leq k \leq v_q$, i.e.

$$pre_q^k(\Omega) = \{x \in \mathcal{P}_q \mid \exists u \in \mathcal{U}_q \ : \ A_q^k x + B_q^k u + E_q d \in \Omega, \ \forall d \in \mathcal{D}_q\}.$$

Therefore, we derived the relationship between the robust one-step predecessor operator for the polytopic uncertain systems, $pre_q(\cdot)$, and the one-step predecessor set of the vertex dynamics, $pre_q^k(\cdot)$ for $k = 1, \cdots, v_q$. From Proposition 4.2, it turns out that the robust one-step predecessor set for a piecewise linear set Ω under polytopic uncertain linear dynamics can be reduced to the finite intersection of one-step predecessor sets corresponding to the dynamic matrix polytope vertices, which have no parametric uncertainty. The predecessor set under deterministic linear dynamics, $pre_q^k(\Omega)$, has been studied extensively in the literature and can be computed by Fourier-Motzkin elimination and linear programming techniques.

Proposition 4.3 The robust one-step predecessor set for a (non-convex) piecewise linear set Ω, $pre(\Omega)$, can be written as a finite union of polyhedra.

Although the convexity is not preserved under the one-step predecessor operation, the piecewise linearity remains unchanged in view of Proposition 4.3. Therefore, one can apply the predecessor operation recursively, and this is explored in the next section.

4.3 Backward Reachability Analysis

Given Ω_1 and Ω_2, define robust one-step controllable set as all the states in Ω_1 for which there exist admissible control signals to drive such state into Ω_2 in the next step, for all

allowable uncertainties and disturbances, i.e.

$$\mathcal{K}_1(\Omega_1,\Omega_2) = \{x(t) \in \Omega_1 \quad | \quad \exists q(t) \in Q,\ u(t) \in \mathcal{U}_{q(t)},\ :\ x(t) \in \mathcal{P}_{q(t)},$$
$$A_{q(t)}(w)x(t) + B_{q(t)}(w)u(t) + E_{q(t)}d(t) \in \Omega_2,\ \forall d(t) \in \mathcal{D}_{q(t)},\ w \in \mathcal{W}\}$$

The robust one-step controllable set $\mathcal{K}_1(\Omega_1,\Omega_2)$ can be computed as follows:

$$\mathcal{K}_1(\Omega_1,\Omega_2) = pre(\Omega_2) \cap \Omega_1$$

It follows from Proposition 4.3 that the robust one-step controllable set $\mathcal{K}_1(\Omega_1,\Omega_2)$ is a piecewise linear set if Ω_1 and Ω_2 are given as piecewise linear sets. Therefore, the robust one-step controllable set operator can be used recessively to define i-step controllable set $\mathcal{K}_i(\Omega_1,\Omega_2)$ as follows.

$$\mathcal{K}_i(\Omega_1,\Omega_2) = \mathcal{K}_1(\Omega_1, \mathcal{K}_{i-1}(\Omega_1,\Omega_2)),$$

where $i \geq 1$ and $\mathcal{K}_0(\Omega_1,\Omega_2) = \Omega_2$. When the Ω_1 is set to be the whole state space $\mathcal{X}$, the i-step controllable set $\mathcal{K}_i(\mathcal{X},\Omega)$ is called the i-step backward reachable set from region Ω, which is denoted as $\mathcal{R}_i(\Omega)$.

4.4 Commands for Backward Reachability Analysis

command	description
`pre`	predecessor operator $pre(\cdot)$
`prein`	inner-approximation of the predecessor operator $pre(\cdot)$
`dreach`	finite-step controllable set $\mathcal{K}_i(\Omega_1,\Omega_2)$
`reach`	finite-step backward reachable set

Table 3: Commands for backward reachability analysis.

Table 3 lists the commands for predecessor operator and backward reachability analysis. The command `pre` exactly calculates the one step backward reachable set for uncertain piecewise linear systems. Command `prein` calculates an inner-approximation of the one step backward reachable set. Notice that `pre` and `prein` return the same result when the region in concern is convex. Their only difference is in the way dealing with non-convexity. As we discussed, the main difficulty for the calculation of $pre(\Omega)$ is the non-convexity of the piecewise linear set Ω. The command `pre` includes two times of calculation of the complement set of non-convex sets, and the complement operation become inefficient quickly as the number of polyhedra contained in the piecewise linear set and the number of faces for each polyhedron increase. This obstacle makes the calculation of $pre(\Omega)$ not cheap when Ω is non-convex, especially when one recursively uses `pre` to calculate backward reachable

set or finite-step controllable set. This motivate us to introduce the command `prein` to inner-approximate $pre(\Omega)$, which is based on the following fact:

$$\Omega = \bigcup_i \Omega_i \;\Rightarrow\; pre(\Omega) \supseteq \bigcup_i pre(\Omega_i).$$

where Ω_i are convex sets. The command `prein` does not involve complement operation when dealing with non-convex piecewise linear set Ω, which makes it more efficient than the command `pre`. However, the price paid is the accuracy, `prein` only returns an inner-approximation of $pre(\Omega)$. The command `dreach` calculates or inner-approximates the finite-step controllable set $\mathcal{K}_i(\Omega_1, \Omega_2)$ from region Ω_1 to Ω_2. The command `reach` calculates or inner-approximates the finite-step backward reachable set. Finally let's see an example for usage of the commands.

Example 4.1 (Predecessor Operator) For the numerical example described in the previous section, consider the region Ω_1, which is defined as $\Omega_1 = \{x \in \mathbb{R}^2 | \|x\|_\infty \leq 10\}$. First we call `pre` to calculate the predecessor set by the following code.

```
% Calculate the one-step predecessor set
prePL = pre(upwlsys,spec(2)),
% Plot the one-step predecessor set
figure(1),clf,
viewpwl(prePL),
% Inner-approxiamte the one-step predecessor set
preinPL = prein(upwlsys,spec(2))
```

The predecessor set is also piecewise linear, as shown in Figure 2. The command `viewpwl` is used to plot a piecewise linear (maybe non-convex) set, which is based on the graphical functions provided by Geometric Bounding Toolbox (GBT 7.0) [5]. The illustration of the commands `dreach` and `reach` will be given in the next example.

5 Safety, Direct Reachability and Attainability

In this section, we first present necessary and sufficient conditions for checking the safety for a given region $\Omega \subset \mathcal{X}$ and the direct reachability between two given regions Ω_1 and Ω_2. Then a necessary and sufficient condition for checking the attainability of a given specification is presented. Finally, the commands for checking safety, direct reachability and attainability are described.

5.1 Safety

The following is an important, well-known geometric condition for a set to be safe (controlled invariant).

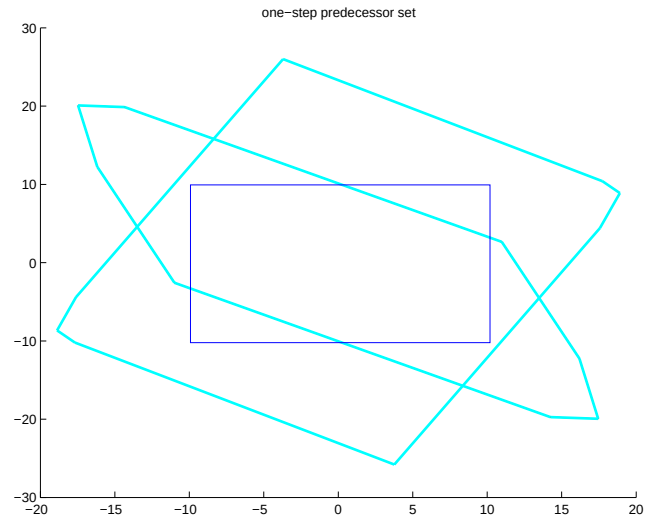

Figure 2: The one-step predecessor set for the region Ω_1.

Theorem 5.1 The set Ω is safe *if and only if* $\Omega \subseteq pre(\Omega)$.

The proof follows immediately from the definition of the predecessor set $pre(\Omega)$. Testing for safety need to: compute $pre(\Omega)$, which can be efficiently done by the predecessor operator described and implemented in the previous section; test whether $\Omega \subseteq pre(\Omega)$, this can be done by a feasibility of some linear programming problems. So this 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.

5.2 Direct Reachability

Secondly, we study the reachability problem for uncertain piecewise linear systems. It should be emphasized that we are interested only in the case when reachability between two regions Ω_1 and Ω_2 is defined so that the state is driven to Ω_2 directly from the region Ω_1 in finite steps without entering a third region. This is a problem of practical importance in hybrid systems since it is often desirable to drive the state to a target region of the state space while satisfying constraints on the state and input during the operation of the system.

The problem of deciding whether a region Ω_2 is directly reachable from Ω_1 can be solved by recursively computing all the states that can be driven to Ω_2 from Ω_1 using the predecessor operator. With the introduction of the finite step controllable set from Ω_1 to Ω_2, the geometric condition to check the direct reachability can be given as follows.

Theorem 5.2 Consider an uncertain piecewise linear systems and the regions Ω_1 and Ω_2. The region Ω_2 is directly reachable from Ω_1 in finite number of steps *if and only if* there

exist finite integer N such that $\Omega_1 \subseteq \bigcup_{i=0}^{N} \mathcal{K}_i(\Omega_1, \Omega_2)$.

5.3 Attainability Checking

Given a finite number of regions $\{\Omega_0,\ \Omega_1, \cdots,\ \Omega_M\}$, the attainability for this sequence of regions specification is equivalent to the following two different kinds of properties, that is the direct reachability from region Ω_i to Ω_{i+1} for $0 \leq i < M$ and the safety for the final region Ω_M. Therefore the attainability checking can be expressed as follows.

Theorem 5.3 The specification $\{\Omega_0,\ \Omega_1, \cdots,\ \Omega_M\}$ is attainable *if and only if* the following conditions hold: First, Ω_M is safe; and secondly the region Ω_{i+1} is directly reachable from Ω_i, for $i = 0, 1, \cdots, M-1$.

Combined with Theorem 5.1 & 5.2, it is straight forward to derive corresponding geometric conditions for attainability.

5.4 Specification Checking Commands

command	description
`issafe`	check the safety of a region
`isreach`	check finite step direct reachability between two regions
`isattain`	check the attainability of a given specification

Table 4: Commands for specification checking.

Table 4 lists the commands for safety, direct reachability and attainability checking. Command `issafe` checks the safety of a piecewise linear (maybe non-convex) region for the UPWL system. By Theorem 5.1, it calls the function `pre` to calculate the one-step predecessor set for the piecewise linear region Ω, then it calls a function `issubset` (see Appendix A) to check whether Ω is a subset of its predecessor set $pre(\Omega)$. `issafe` returns a positive scalar when the geometric condition satisfied, i.e. the region Ω is safe, and returns zero or negative scalar otherwise. Command `isreach` checks the finite steps direct reachability between two piecewise linear regions. It calls function `dreach` and `issubset` to check the geometric condition in Theorem 5.2. By Theorem 5.3, we know attainability checking can be divided into checking the safety for the terminal region and the reachability between two successive regions. So `isattain` calls the function `isreach` and the `issafe`. Finally, let's see an example for usage of the commands.

Example 5.1 (Specification Checking) Consider the uncertain piecewise linear system and the specification defined in the previous examples. First, calculating the one-step

predecessor set of the region Ω_1, $pre(\Omega_1)$, which has been illustrated in Figure 2 as a union of two polytopes. It can be checked that $pre(\Omega_1) \supset \Omega_1$, therefore the region Ω_1 is safe by Theorem 5.1.

Next, we consider direct reachability from another region Ω_0 to Ω_1. It is found that $\Omega_0 \subseteq \bigcup_{i=0}^{3} \mathcal{K}_i(\Omega_0, \Omega_1)$, therefore Ω_1 is directly reachable from Ω_0 in finite number of steps (less or equal to three steps), by Theorem 5.2. The three-step direct reachable set from Ω_0 to Ω_1 is calculated and plotted in the right part of Figure 3. The three-step backward reachable set from Ω_1, namely $\mathcal{R}_3(\Omega)$, is plotted in the left part of Figure 3 for comparation.

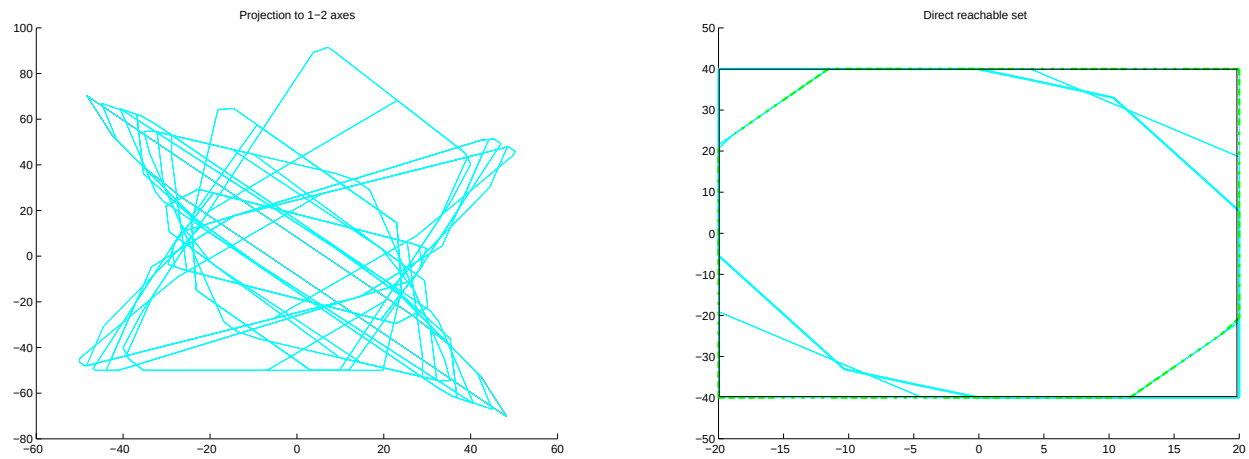

Figure 3: *Left:* The illustration of the three-step backward reachable set from Ω_1, namely $\mathcal{R}_3(\Omega)$. *Right:* Three-step direct reachable set from Ω_0 to Ω_1.

By Theorem 5.3, the attainability of the specification, $\{\Omega_0, \Omega_1\}$, is guaranteed. The attainability checking can be implemented by the following codes.

```
attain = isattain(spec, upwlsys);
if attain>0
    display('the attainability check passed.')
else ...
```

6 Controller Synthesis

The hybrid tracking and regulation control problem considered in this section is to select feasible modes $q(t)$ and to design admissible control signals, $u(t) \in \mathcal{U}_{q(t)}$, such that the state trajectory $x(t)$ goes through the regions, namely Ω_0, Ω_1, $\Omega_2 \cdots$, in the specified order and the closed-loop system satisfies some requirements, such as sequencing of events and eventual execution of actions. In Section 5, we specified the conditions for existence of such control laws so that the closed-loop system satisfies safety, reachability and attainability

specifications. Here we design such control laws based on optimization techniques. As it was discussed in the previous section, the attainability controller synthesis problem can be divided into two basic problems, that is safety controller synthesis and direct reachability controller synthesis. In the following, we first present a systematic procedure for the controller design for these two basic cases. Then a procedure for attainability controller synthesis is given and the commands for controller synthesis and simulation is illustrated.

6.1 Safety

First, we consider the safety controller synthesis for the terminating region, Ω_M. Without loss of generality, we assume that Ω_M is a connected piecewise linear set (may be non-convex). Therefore, we can specify a polytope P_M which is contained in the interior of Ω_M. Let us assume that $P_M = \{x|G_M x \leq g_M\}$. We define the cost function $J_M : Q \times \mathcal{W} \times \mathcal{U}_q \to \mathbb{R}^+$ as

$$\begin{aligned} J_M(q, w, u) &= \|G_M(A_q(w)x + B_q(w)u)\|_\infty \\ &= \|\sum_{k=1}^{v_q} w_k[G_M(A_q^k x + B_q^k u)]\|_\infty \end{aligned}$$

where $\|\cdot\|_\infty$ stands for the infinite norm. The cost function J_M is in fact the Minkowski function induced from the the convex polytope P_M.

Because Ω_M is assumed to be safe, $\Omega_M \subseteq pre(\Omega_M) = \bigcup_q pre_q(\Omega_M)$. Therefore, for any $x \in \Omega_M$, there exists at least one mode $q \in Q$ such that $x \in pre_q(\Omega_M)$. We call such mode feasible for state x. The control signal can be selected as the solution to the following min-max optimization problem for one of such feasible modes q:

$$\begin{aligned} &\min_{u \in \mathcal{U}_q} \max_{w \in \mathcal{W}} J_M(q, w, u) \\ &s.t.\ x \in pre_q(\Omega_M) \end{aligned}$$

The constraint "$x \in pre_q(\Omega_M)$" in the above optimization problem means that the admissible control signal $u \in \mathcal{U}_q$ must keep all the possible next step states inside Ω_M despite the uncertainties and disturbances along the mode q. The existence of such control signals comes from the safety of Ω_M and the feasibility of mode q for current state x by assumption. Therefore, we can always select an admissible control signal for a feasible mode q. The optimal action of the controller is one that tries to minimize the maximum cost, to try to counteract the worst disturbance and the worst model uncertainty. In the following, we will describe step by step how to solve the above min-max optimization problem for the controller design.

First, we assume that Ω_M is convex, and it can be represented by $\Omega_M = \{x|F_M x \leq \theta_M\}$. For such case, the above min-max optimization problem can be equivalently reduced to the following linear programming problem [13], if the control constraints $\mathcal{U}_q$ is given as a

polytope.

$$\min_{u\in\mathcal{U}_q} z$$
$$s.t.\begin{cases} G_M[A_q^1x+B_q^1u]\le \bar{z} \\ G_M[A_q^2x+B_q^2u]\le \bar{z} \\ \cdots\cdots \\ G_M[A_q^{N_q}x+B_q^{N_q}u]\le \bar{z} \\ -G_M[A_q^1x+B_q^1u]\le \bar{z} \\ -G_M[A_q^2x+B_q^2u]\le \bar{z} \\ \cdots\cdots \\ -G_M[A_q^{v_q}x+B_q^{v_q}u]\le \bar{z} \\ F_M[A_q^1x+B_q^1u]\le g_M-\delta \\ F_M[A_q^2x+B_q^2u]\le g_M-\delta \\ \cdots\cdots \\ F_M[A_q^{v_q}x+B_q^{v_q}u]\le g_M-\delta \\ u\in\mathcal{U}_q \end{cases}$$

where $\bar{z}$ stands for a column vector of proper dimension and with all elements being an auxiliary variable $z\in\mathbb{R}$. Here δ is a vector whose components are given by $\delta_j=\max_{d\in\mathcal{D}_q} F_j^TE_qd$, and F_j^T is the j-th row of matrix F_M, which represents the worst case of the disturbances with respect to the safety specification. Hence the admissible control signal u can be designed for each feasible mode q of the current state x by solving a linear program of the above form. The feasibility of the linear program is guaranteed by the safety of Ω_M and the feasibility of mode q, i.e. $x\in pre_q(\Omega_M)$. Note that there may be more than one modes feasible for the current state x. For such case, a linear program is solved for each feasible mode, and the active mode and control action is selected as the pair that return the minimal value of the cost function. Notice that the values for cost functions of different mode q are comparable, because they are values of Mikowshki functions induced from the same convex set P_M. After the control action is applied and the system evolves along the active mode, the next step state x' is guaranteed to be contained inside Ω_M, namely the safety of Ω_M is guaranteed. Then the controller synthesis process repeated for the new state x' to design next step control signals.

In the following, we will deal with the case when Ω_M is a non-convex piecewise linear set. As it is shown in Proposition 4.3, $pre_q(\Omega_M)$ can be written as a finite union of polyhedra. Assume $pre_q(\Omega_M)=\bigcup_{i=1}^s\Phi_i$. For each feasible mode q, $x\in pre_q(\Omega_M)=\bigcup_{i=1}^s\Phi_i$, then there exists at least one i, for $1\le i\le s$, such that $x\in\Phi_i$. The admissible control signal $u\in\mathcal{U}_q$, which keeps the state trajectory remaining in Ω_M despite the uncertainties and disturbances along the mode q, can be designed through the following min-max optimization problems.

$$\min_{u\in\mathcal{U}_q}\ \max_{w\in\mathcal{W}} J_M(q,w,u)$$
$$s.t.\ x\in\Phi_i$$

Because of the convexity of Φ_i, for $i = 1, \cdots, s$, each min-max optimization problem can be reduced into a linear programming problem as shown for the case of convex Ω_M. It is clear that at least one of these linear programs is feasible if q is a feasible mode for x, and the solution u is an admissible control that satisfies the safety specification. It is worth pointing out that the above partition of the non-convex set $pre_q(\Omega_M)$ into finite number of convex sets Φ_i does not imply that each Φ_i is safe. In fact, we only rely on the property that if $x \in \Phi_i \subset pre_q(\Omega_M)$ then admissible control signals to keep the next state x' remaining inside Ω_M (not Φ_i) along the mode q do exist.

Therefore, at each step the admissible hybrid control law is designed by solving a finite number of linear programs. The number of linear programs to be solved depends not only on the number of feasible modes for current state but also on how many polytopic cells Φ_i of the (non-convex) region $pre_q(\Omega_M)$ contains. The number of linear constraints for each linear programming problem is determined by the number of vertex matrices, namely v_q, and the number of faces of the polytopic subregion Φ_i.

6.2 Direct Reachability

Next, we consider the reachability between two successive regions Ω_i and Ω_{i+1}. The control objective is to drive every state in Ω_i to Ω_{i+1} without entering a third region. Similarly, we specify a polytope $P_c \subseteq \Omega_{i+1}$, which can be represented as $P_c = \{x | G_C x \leq g_C\}$. We define the cost functional, $J_C : Q \times \mathcal{W} \times \mathcal{U}_q \to \mathbb{R}^+$ as

$$\begin{aligned} J_C(q, w, u) &= \|G_C(A_q(w)x + B_q(w)u)\|_\infty \\ &= \|\sum_{k=1}^{v_q} w_k[G_C(A_q^k x + B_q^k u)]\|_\infty \end{aligned}$$

Because Ω_{i+1} is direct reachable from Ω_i, so that for all states x in Ω_i, there always exist discrete mode q, and admissible control signal $u \in \mathcal{U}_q$ such that the next state remains in $\Omega_i \cup \Omega_{i+1}$, for all allowable disturbances and uncertainties. Therefore, there always exists $q \in Q$ to make the following min-max optimization problem feasible:

$$\begin{gathered} \min_{u \in \mathcal{U}_q} \max_{w \in \mathcal{W}} J_C(q, w, u) \\ s.t.\ x \in pre_q(\Omega_i \cup \Omega_{i+1}) \end{gathered}$$

This optimization problem is of the same type as the one studied in the safety controller synthesis. Based on similar arguments as in the safety controller synthesis, the above min-max optimization problem can be reduced into a finite number of linear programming problems.

6.3 Tracking Attainable Specifications

Assume that the given tracking and regulation specification $\{\Omega_0, \Omega_1, \cdots, \Omega_M\}$ is attainable. Our task in this section is to select feasible modes $q(t)$ and to design admissible control signals, $u(t) \in \mathcal{U}_{q(t)}$, such that for all the initial states x_0 contained in Ω_0 will be driven into Ω_1, without violating state and input constraints, then into Ω_2 and so on. Finally, the state trajectory will reach the final region Ω_M and stay there.

The controller design procedure is now described: For initial condition $x_0 \in \Omega_0$, determine the feasible modes for x_0 as $act(x_0) = \{q \in Q \mid x_0 \in pre_q(\Omega_0 \cup \Omega_1)\}$. Note that $act(x_0)$ is a non-empty finite set by reachability assumption. For each feasible mode, we employ the reachability controller design procedure for Ω_0 and Ω_1. This can be done by solving a finite number of linear programs. And for each feasible mode q, at least one of the linear programs is feasible and returns an admissible control signal $u \in \mathcal{U}_q$. This claim comes from the assumed direct reachability from Ω_0 to Ω_1. The active mode q and control signal $u(x_0)$ is selected among these admissible control signal pairs as the one that returns the smallest value of the cost function, which represents the best effort being taken to reach Ω_1. Then, the control signal is applied and its corresponding feasible mode q is followed, and the next state x_1 is guaranteed to be contained in Ω_0 or Ω_1, for all possible disturbances and uncertainties. If x_1 does not reach Ω_1, then the reachability controller design procedure for Ω_0 and Ω_1 is taken again to obtain the next step feasible mode and admissible control signal. If, after some steps, $x(t)$ is driven into Ω_1, then the reachability controller design procedure for Ω_1 to Ω_2 is invoked. This procedure is repeated and finally $x(t)$ reaches Ω_M, for which the safety controller synthesis procedure is followed to obtain the active modes and admissible control signals. Similar to the case of reachability controller synthesis, the non-emptiness of the feasible modes, and the feasibility of the finite number of linear programs can be guaranteed for the safety controller synthesis of Ω_M.

The following algorithm describes the controller design procedure that guarantees the attainability for an attainable specification described by $\{\Omega_0, \Omega_1, \cdots, \Omega_M\}$.

Algorithma 6.1 Attainability Control

INPUT: $\{\Omega_0,\ \Omega_1, \cdots, \Omega_M\}$;
for $i = 0,\ 1, \cdots,\ M-1$,
  while $x(t) \in \Omega_i$ and $x(t) \notin \Omega_{i+1}$
  Design reachability controller from Ω_{i} to $\Omega_{\mathrm{i+1}}$
  end
end
Design safety controller for Ω_{M}
OUTPUT: q^*, u^*

6.4 Commands for Controller Design and Simulation

command	description
`safereg`	safety controller design
`reachreg`	direct reachability controller design
`regulator`	attainability controller design
`upwlsim`	simulation for given UPWL system

Table 5: Commands for controller synthesis and simulation.

Table 5 lists the commands for controller synthesis and simulation. Command `safereg` first calls function `issafe` to check the safety of the piecewise linear region in question. If the checking passed, then `safereg` calculates and returns the feasible mode q and the appropriate control signal $u \in \mathcal{U}_q$. Similarly, Command `reachreg` first checks the reachability. If the checking passed, then `reachreg` calculates and returns the feasible mode q and the appropriate control signal $u \in \mathcal{U}_q$. The command `regulator` calls function `reachreg` to select feasible mode and design admissible control signal to satisfy the reachability between two nonterminal successive regions in the specification. And when the final (terminal) region is reached, `regulator` calls `safereg` to calculate the feasible mode q and the appropriate control signal $u \in \mathcal{U}_q$ in order to guarantee safety specification. At each step the `regulator` also calls `upwlsim` for simulation of the trajectory. Finally let's see an example for usage of the commands.

Example 6.1 (Controller Design) Consider the same example setup as in the previous section. We have defined a specification, $\{\Omega_0, \Omega_1\}$, and checked its attainability. Here we will design the controller to satisfy the tracking and regulation specification. Assume initial condition $x_0 = [16, 36]^T$, which is contained in Ω_0 but not contained in Ω_1. First, design the direct reachability control signal for x_0. Select the cost function as the induced Minkowski function of the set $P_C = \{x \in \mathbb{R}^2 | \|x\|_\infty \leq 10\}$ for each mode $q = 1, 2$. Then solve the optimization problem w.r.t. the cost function for each mode under constraints "$x \in pre_q(\Omega_0 \cup \Omega_1)$". Following the procedure developed above, the active mode and control signal can be designed by solving a finite number of linear programs. In this example, we need to solve two linear programs with 26 linear inequality constraints in u and z each (16 of these constraints are induced from the cost function, 8 constraints come from predecessor constraint, and 2 are control constraints). Here it is shown that the active mode is $q = 2$ and the control signal $u = -1.00 \in \mathcal{U}_2$ return the best control effort. This can be automatically done by calling the function `reachreg`. Therefore, if the control signal is applied and the mode $q = 2$ is followed, the next state x_1 is guaranteed to be contained in $\Omega_1 \bigcup \Omega_0$ despite uncertainties and disturbances. If we simulate the state evolution under nominal condition, i.e. setting $\mathcal{D} = \{0\}$ and choosing the epicenter of the state matrix $\frac{1}{2}(A_q^1 + A_q^2)$ for each mode, then we obtain the next step state $x_1 = [-2.42, 16.67]^T$, which is contained in $\Omega_0 \bigcup \Omega_1$ as expected.

Command `upwlsim` is called by function `regulator` to return the next step closed-loop systems state under nominal conditions. Because $x_1 \notin \Omega_1$, the reachability regulation design is repeated again, also by solving two linear programs with 26 linear inequality constraints. After one more step, we obtain $x_2 = [5.13, 5.09]^T$ which is contained in Ω_1. Then a safety regulation controller is designed. The cost function is induced from the region Ω_1, because of its convexity. Similarly, we obtain active modes and control signals by solving two linear programs at each step, which can be obtained by calling `safereg`. The following code satisfy this purpose, and the simulation results under nominal assumptions are shown in Figure 4. The sequence of selected active mode and admissible control signals of the controller is also illustrated in Figure 4.

```
x0 = [16;36];
[u,Reg,qss,xss] = regulator( upwlsys, spec, x0);
figure(1),clf
plot(xss(1,:),xss(2,:))
figure(2), clf
subplot(211),stem([1:length(qss)],qss);
subplot(212),stem([1:length(uss)],uss)
```

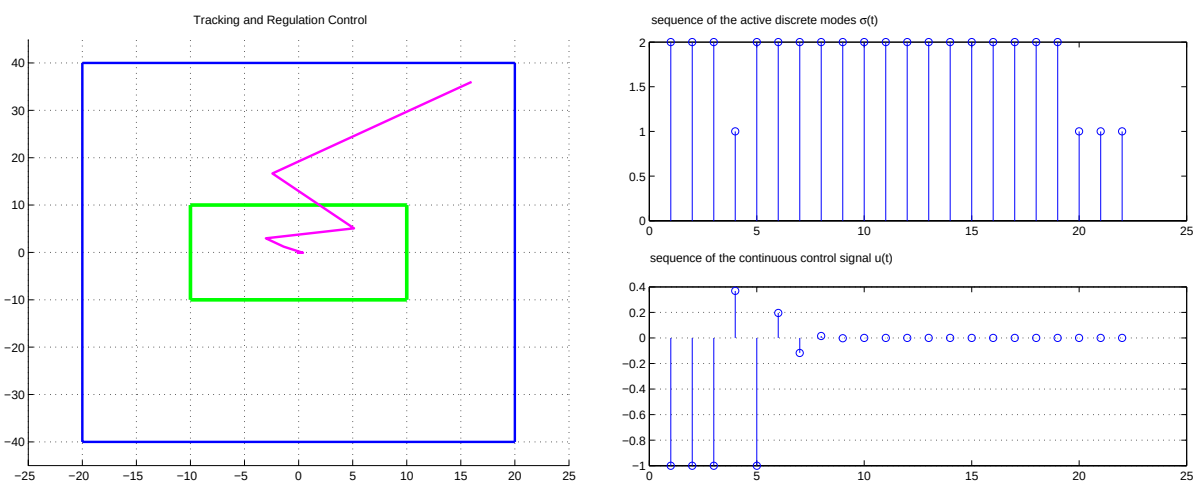

Figure 4: *Left:* Simulation for closed-loop nominal plant (assuming $d = 0$). *Right:* The active mode sequence q and control signals u of the controller.

7 Conclusions

This paper has presented a Matlab toolbox, HY*, for the tracking and regulation control problem for the polytopic uncertain piecewise linear systems. The existence of a controller

such that the closed-loop system follows desired sequence of regions under uncertainty and disturbance was studied first. The analysis is based on computation of predecessor operator and backward reachability analysis. Then, using the optimization based regulator introduced, we presented a systematic procedure for controller design by using linear programming techniques. The toolbox is available from the authors upon request.

References

[1] `http://www.aut.ee.ethz.ch/~hybrid/hysdel/`

[2] `http://www.eecs.berkeley.edu/~tah/HyTech`

[3] `http://www-verimag.imag.fr/~tdang/ddt.html`

[4] `http://www.ece.cmu.edu/~webk/checkmate/`

[5] `http://www.sysbrain.com/gbt/`

[6] P. J. Antsaklis, Ed., *Proceedings of the IEEE*, Special Issue on Hybrid Systems: Theory and Applications, vol. 88, no. 7, 2000.

[7] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise linear dynamical systems," in *Proceedings of the 3rd Hybrid Systems Computation and Control International Workshop*, 2000.

[8] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407-427, 1999.

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

[10] T. A. Henzinger, P. -H. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," *Software Tools for Technology Transfer*, vol. 1, pp. 110-122, 1997.

[11] H. Lin, X. D. Koutsoukos, and P. J. Antsaklis, "HySTAR: A toolbox for hierarchical control of piecewise linear hybrid dynamical systems," in *Proceedings of 2002 American Control Conference*, 2002.

[12] H. Lin and P. J. Antsaklis, " Robust regulation of polytopic uncertain linear hybrid systems with networked control system applications", Chapter in *Contemporary Issues in Systems Stability and Control with Applications*, D. Liu and P. J. Antsaklis Eds., Chp. 4, pp. 83-108, Birkhäuser, 2003.

[13] H. Lin and P. J. Antsaklis, "Robust tracking and regulation control of uncertain piecewise linear systems," ISIS Technical Report, ISIS-2003-005, October 2003. Available on-line at `http://www.nd.edu/~isis/techreports/isis-2003-005.pdf`

[14] B. Silva, O. Stursberg, B. Krogh, and S. Engell, "An assessment of the current status of algorithmic approaches to the verification of hybrid systems," in *Proceedings of the 40th Conference on Decision and Control*, 2001.

Appendix A: Toolkit for non-convex piecewise linear sets

In the implementation of the backward reachability analysis and regulator synthesis for the polytypic uncertain piecewise linear systems, there involves a lot of operations on the piecewise linear (Pwl) sets (maybe non-convex), for example intersection, union, linear transform and so on. For convenience, we build a separate toolkit in order to deal with non-convex Pwl sets. Table 6 lists the commands for operations on Pwl sets.

command	description
`cellreduce`	reduces the redundant cells of a (non-convex) Pwl set.
`fme`	Fourier-Motzkin elimination method
`iscontain`	check whether the collection of points is contained in a Pwl set
`isnull`	check whether a polytope has empty interior
`issubset`	check whether a Pwl set is a subset of another Pwl set
`lintrans`	computes the linear transform of a Pwl set
`mfease`	feasibility check of a Pwl set
`mreduce`	reduction of a constraints of a polytope
`msum`	calculates the Minkowski sum of two polytopes
`pdiff`	calculates the Pontryagin difference between two Pwl sets
`plcomp`	complement set of a Pwl set
`plinters`	intersection set of two Pwl sets
`plunion`	union set of two Pwl sets
`viewpwl`	plots the Pwl set

Table 6: Commands for dealing with non-convex piecewise linear sets.

Appendix B: HyStar Command Reference

command	description
`addynamics`	define system dynamics and constraints
`addregion`	define the polyhedral regions $\mathcal{P}_q$
`addspec`	add specification
`dreach`	finite-step controllable set $\mathcal{K}_i(\Omega_1, \Omega_2)$
`getspec`	extract the specification object
`getupwl`	extract UPWLsys object
`isattain`	check the attainability of a given specification
`isreach`	check finite step direct reachability between two regions
`issafe`	check the safety of a region
`pre`	predecessor operator $pre(\cdot)$
`prein`	inner-approximation of the predecessor operator $pre(\cdot)$
`reach`	finite-step backward reachable set
`reachreg`	direct reachability controller design
`regulator`	attainability controller design
`safereg`	safety controller design
`setspec`	initialize the specification object
`setupwl`	initialize uncertain piecewise linear system (UPWLsys) object
`upwlsim`	simulation for given UPWL system

Table 7: Command list for HyStar.

ADDREGION **ADDREGION**

Purpose

Adds a new polyhedral region partition to the uncertain piecewise linear systems currently described.

Synopsis

```
reg = addregion(G, w, UsedDynamics)
```

Description

Adds a new polyhedral region partition $\mathcal{P}_q$ to the uncertain piecewise linear systems currently described. A label can be optionally attached to this specification to facilitate future reference to it.

Parameters

Input :

`G, w`
Constraint matrices that specify the polyhedral region $\mathcal{P}_i = \{x \mid Gx \leq w\}$.
`UsedDynamics`
A reference label for which dynamics to use in this region. Several dynamics specifications can be linked to one region by entering a vector here. That is the discrete state q.
Output :
`reg`
A label for future reference to the region.

See Also

`SETUPWL, ADDYNAMICS, GETUPWL`

ADDSPEC **ADDSPEC**

Purpose

Adds a new region specification to the region-sequence specification currently described.

Synopsis

```
reg = addspec( upwlsys, G, w, seq )
```

Description

Adds a new region specification to the specification currently described. A label can be optionally attached to this specification to facilitate future reference to it

Parameters

Input :
`upwlsys`
The uncertain piecewise linear system (UPwlsys) in question.
`G, w`
Constraint matrices (G, w) for region identification of the specification: $\Omega_i = \{x \in \mathbb{R}^n | Gx \leq w\}$.
`seq`
Indicate the order of the region in the tracking and regulation specification, e.g. the first region `(seq=1)` or the `k`th region `(seq=k)`, and so on.
Output :
`reg`
A label for future reference to the region specification.

See Also

`SETSPEC, GETSPEC`

ADDYNAMICS **ADDYNAMICS**

Purpose

Adds a new continuous variable dynamics to the currently active uncertain piecewise linear systems.

Synopsis

```
dyn = addynamics(A, B, E, U, D, comb)
```

Description

Adds a new continuous variable dynamics to the uncertain piecewise linear system currently described. A label can be optionally attached to this dynamics to facilitate future reference to it.

Parameters

Input :

`A,B,E`

Data describing the polytopic uncertain dynamics

$$x(t+1) = \sum_i w_i A_i x(t) + \sum_i w_i B_i u + Ed. \tag{7.1}$$

`A` (or `B`) is a cell with elements describing the vertex matrices A_i (or B_i respectively), namely `A`=$\{A_1, \cdots, A_v\}$ (or `B`=$\{B_1, \cdots, B_v\}$ respectively).

`U,D`

Data describing the bound of continuous control and disturbance. `U` and `D` are both in `struct` form with field `.l` and field `.r`. The control constraint is described as $\mathcal{U} = \{u \in \mathbb{R}^m | \texttt{U.l}u \leq \texttt{U.r}\}$. Similarly, $\mathcal{D} = \{d \in \mathbb{R}^r | \texttt{D.l}u \leq \texttt{D.r}\}$.

`comb`

Flag indicating whether the combination between the vertex matrices of cell `A` and `B` is necessary, which means that polytopic uncertainty $(A(w), B(w)) \in Conv\{(A_i, B_j)\}$ for all vertex matrices of cell `A` and `B`, namely A_i and B_j.

Output :

`dyn`

A label for future reference to the dynamics.

See Also

`SETUPWL, ADDREGION, GETUPWL`

DREACH **DREACH**

Purpose

Calculating the finite-step controllable set between two (non-convex) piecewise linear sets, $\mathcal{K}_N(\Omega_1, \Omega_2)$.

Synopsis

```
DR = dreach(upwlsys,PL1,PL2,nstep)
```

Description

Calculating the finite-step controllable set between two (non-convex) piecewise linear sets, $\mathcal{K}_N\Omega_1(\Omega_2)$. Note that the finite-step controllable set $\mathcal{K}_N(\Omega_1, \Omega_2)$ is recursively defined as

$$\mathcal{K}_i(\Omega_1, \Omega_2) = \mathcal{K}_1(\Omega_1, \mathcal{K}_{i-1}(\Omega_1, \Omega_2)),$$

where $i \geq 1$ and $\mathcal{K}_0(\Omega_1, \Omega_2) = \Omega_2$.

Parameters

Input :

`upwlsys`

specify the UPwlsys object in concern.

`PL1,PL2`

Define the two (non-convex) piecewise linear regions for reachability analysis. Please note that the non-convex piecewise linear sets `PL1` and `PL2` is express in a cell format, whose elements are convex polyhedra (The union of these convex polyhedra gives the non-convex piecewise linear set.). Each polyhedron is in `struct` format with two fields, namely `.l` and `.r`.

`nstep`

Specify the step need to be calculated. The default value of `nstep` is 3.

Output :

`DR`

The N-step controllable set between two (non-convex) piecewise linear sets, $\mathcal{K}_N(\Omega_1, \Omega_2)$, where $N =$ `nstep`.

See Also

`REACH, PRE, PREIN, ISREACH`

GETSPEC **GETSPEC**

Purpose

Extract the specification object.

Synopsis

```
spec = getspec
```

Parameters

Output :

`spec`

The variable `spec` contains aggregated data of the ordered region-sequence tracking and regulation specification for the uncertain piecewise linear system in concern. The variable

`spec` is called to refer to the tracking and regulation specification defined by `setspec` and `addspec`.

See Also

`SETSPEC, ADDSPEC`

GETUPWL **GETUPWL**

Purpose

Extract the UPwlsys object.

Synopsis

```
upwlsys = getupwl
```

Description

Returns the internal representation UPwlsys of an uncertain piecewise linear system once this system has been fully described with `ADDREGION` and `ADDYNAMICS`. The internal representation UPwlsys can be passed directly to any uncertain piecewise linear system's analysis and design function.

Parameters

Output :

`upwlsys`

The variable `upwlsys` contains aggregated data of the uncertain piecewise linear system model, including state partition information, discrete modes and dynamics for each mode etc.

See Also

`SETUPWL, ADDYNAMICS, ADDREGION`

ISATTAIN **ISATTAIN**

Purpose

Check attainability of the tracking and regulation specification.

Synopsis

```
[flag] = isattain(spec, upwlsys, nmax)
```

Description

Check attainability of the tracking and regulation specification, which is defined by calling commands `setspec`, `addspec`, and `getspec`. The function `isattain` checks the attainability based on Theorem 5.3, and calls the function `isreach` and `issafe`.

Parameters

Input :

`upwlsys`

Specify the UPwlsys object in concern.

`spec`

Specify the tracking and regulation specification in concern, which is defined by calling commands `setspec`, `addspec`, and `getspec`.

`nmax`

Specify the maximal step length allowed.

Output :

`flag`

`flag > 0`, if specification `spec` is attainable.

`flag < 0`, otherwise.

See Also

`ISREACH, ISSAFE`

ISREACH **ISREACH**

Purpose

Check direct reachability between two regions in finite steps.

Synopsis

```
[flag] = isreach (upwlsys, PL1, PL2, nstep)
```

Description

Check direct reachability between two piecewise linear (may be non-convex) regions, described by `PL1` and `PL2`, in a finite number of steps. It is based on the geometric condition in Theorem 5.2 and calls the function `dreach` to calculate the finite step controllable sets.

Parameters

Input :

`upwlsys`

Specify the UPwlsys object in concern.

`PL1, PL2`

Specify the successive region pair Ω_1 and Ω_2 in concern. `PL1` and `PL2` may represent a non-convex piecewise linear set.

`nstep`

Specify the maximal step length allowed.

Output :

`flag`

`flag > 0`, if region `PL1` can direct reach region `PL2` in `nstep` steps or less.

`flag < 0`, otherwise.

See Also

`DREACH, PRE, PREIN`

ISSAFE **ISSAFE**

Purpose

Check safety of a (non-convex) piecewise linear region.

Synopsis

```
[flag] = issafe (upwlsys, PL)
```

Description

Check the safety of a piecewise linear (may be non-convex) region, described by `PL`. It is based on the geometric condition in Theorem 5.1 and calls the function `pre` to calculate the one-step predecessor set of `PL`.

Parameters

Input :

`upwlsys`

Specify the UPwlsys object in concern.

`PL`

Specify the region Ω in concern. `PL` may represent a non-convex piecewise linear set.

Output :

`flag`

`flag > 0`, if the region in concern is safe.

`flag < 0`, otherwise.

See Also

`PRE`

PRE **PRE**

Purpose

Calculate the one-step predecessor set.

Synopsis

```
prePL = pre( upwlsys, PL )
```

Description

`pre` calculates the one-step predecessor set for the (non-convex) piecewise linear region described by `PL` based on the methods developed in Section 4.

Parameters

Input :

`upwlsys`

Specify the UPwlsys object in concern.

`PL`

Specify the piecewise linear region in concern. Note that `PL` may represent a non-convex piecewise linear set.

Output :

`prePL`

Returns the exact one-step predecessor set for region `PL`.

See Also

`PREIN`

PREIN **PREIN**

Purpose

Calculate an inner approximation of the one-step predecessor set.

Synopsis

```
preinPL = prein( upwlsys, PL )
```

Description

`prein` calculates an inner approximation of the one-step predecessor set for the (non-convex) piecewise linear region described by `PL` based on the following observation:

$$\Omega = \bigcup_i \Omega_i \;\Rightarrow\; pre(\Omega) \supseteq \bigcup_i pre(\Omega_i).$$

The command `prein` does not involve complement operation when dealing with non-convex piecewise linear set Ω, which makes it more efficient than the command `pre`. Note that `pre` and `prein` returns the same result for convex set `PL`.

Parameters

Input :

`upwlsys`

Specify the UPwlsys object in concern.

`PL`

Specify the piecewise linear region in concern. Note that `PL` may represent a non-convex piecewise linear set.

Output :

`preinPL`

Returns the inner approximation of the one-step predecessor set for region `PL`.

See Also

PRE

REACH REACH

Purpose

Calculating the finite-step backward reachable set for a (non-convex) piecewise linear set, $\mathcal{R}_N(\Omega)$.

Synopsis

```
R = reach(upwlsys, PL, nstep)
```

Description

Calculating (or inner-approximating) the finite-step backward reachable set from region Ω, which equals to finite-step controllable set from $\mathcal{X}$ (whole state region) to Ω, namely, $\mathcal{R}_i(\Omega) = \mathcal{K}_i(\mathcal{X}, \Omega)$.

Parameters

`upwlsys`

Specify the UPwlsys object in concern.

`PL`

Specify the piecewise linear region in concern. Note that `PL` may represent a non-convex piecewise linear set.

`nstep`

Specify the step need to be calculated. The default value of `nstep` is 3.

Output :

`R`

The N-step backward reachable set from region Ω, where $N =$ `nstep`.

See Also

`REACH, PRE, PREIN`

REACHREG REACHREG

Purpose

Controller design for the direct reachability specification.

Synopsis

```
[u, q, flag, mincost] = reachreg( upwlsys, PL1, PL2, x0)
```

Parameters

Input :

`upwlsys`

Specify the UPwlsys object in concern.

`PL1, PL2`

Specify the successive region pair Ω_1 and Ω_2 in concern. `PL1` and `PL2` may represent a non-convex piecewise linear set.

`x0`

Specify the initial state which is contained inside `PL1`.

Output :

`u,q`

Returns the feasible continuous control signal `u` and active mode `q` that satisfy the reachability specification.

`flag`

Feasibility of the controller synthesis problem. If `flag` is positive, then feasible control laws exist and being calculated. If `flag` is negative, then the specification is not feasible.

`mincost`

Returns the cost function value for the best control effort.

See Also

`SAFEREG, REGULATOR`

REGULATOR **REGULATOR**

Purpose

Controller design for the tracking and regulation specification.

Synopsis

```
[u, Reg, qss, xss] = regulator( upwlsys, spec, x0)
```

Parameters

Input :

`upwlsys`

Specify the UPwlsys object in concern.

`spec`

Specify the tracking and regulation specification in concern, which is defined by calling commands `SETSPEC, ADDSPECT`, and `GETSPEC`.

`x0`

Specify the initial state which is contained inside the initial region in `spec`.

Output :

`u, qss`

Returns the sequence feasible continuous control signal `u` and active mode `qss` that satisfy the specification.

`xss`

The trajectory of the closed-loop system under nominal condition.

`Reg`

The sequence of region index that the state trajectory visited.

See Also

`REACHREG, SAFEREG, UPWLSIM`

SAFEREG **SAFEREG**

Purpose

Controller design for the safety specification.

Synopsis

```
[u, q, flag, mincost] = safereg( upwlsys, PL, x0)
```

Parameters

Input :

`upwlsys`

Specify the UPwlsys object in concern.

`PL`

Specify the region Ω_M in concern. `PL` may represent a non-convex piecewise linear set.

`x0`

Specify the initial state which is contained inside `PL`.

Output :

`u,q`

Returns the feasible continuous control signal `u` and active mode `q` that satisfy the specification.

`flag`

Feasibility of the controller synthesis problem. If `flag` is positive, then feasible control laws exist and being calculated. If `flag` is negative, then the specification is not feasible.

`mincost`

Returns the cost function value for the best control effort.

See Also

`REACHREG, REGULATOR`

SETSPEC **SETSPEC**

Purpose

Initializes the description of a new specification for the uncertain piecewise linear system

under concern.

Synopsis

`setspec(upwlsys,spec0)`

Description

Initializes the description of a new specification for the uncertain piecewise linear system under concern.

1. To start from scratch, type
   `setspec(upwlsys,[])`
2. To add on to an existing specification spec0, type
   `setspec(upwlsys,spec0)`

See Also

`ADDSPEC, GETSPEC`

SETUPWL **SETUPWL**

Purpose

Initializes the description of a new uncertain piecewise linear system.

Synopsis

`setupwl(upwlsys0)`

Description

Initializes the description of a new uncertain piecewise linear system.

1. To start from scratch, type
   `setupwl([])`
2. To add on to an existing piecewise linear hybrid dynamical system `upwlsys0`, type
   `setupwl(upwlsys0)`

See Also

`ADDYNAMICS, ADDREGION, GETUPWL`

UPWLSIM **UPWLSIM**

Purpose

Simulation for a given UPWL system under nominal assumption.

Synopsis

```
[xc,xv] = upwlsim(upwlsys, q0, u, x0)
```

Description

Simulates an uncertain piecewise linear system. The system should be specified using `SETUPWL, ADDREGION, ADDYNAMICS`, and `GETUPWL`. The state evolution is simulated under nominal condition, i.e. setting $\mathcal{D} = \{0\}$,

Parameters

Input :

`upwlsys`

The uncertain piecewise linear system to be simulated.

`q0`

The active mode to be followed.

`u`

The continuous control signal, which maybe returned from the synthesis commands like `safereg`, `reachreg` or `regulator`.

`x0`

Specify the initial state of the UPwlsys in concern.

Output :

`xc`

The epi-center of the next continuous sate under nominal assumption.

`xv`

The cell `xv` contains a collection of vectors, whose convex combination, $conv\{xv\}$, gives all possible next step continuous sate under nominal assumption.

See Also

`REGULATOR`