

Copyright © 1999, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

**HYBRID SYSTEMS: MODELING,
ANALYSIS AND CONTROL
EECS 291E
LECTURE NOTES AND CLASS PROJECTS**

by

John Lygeros, Shankar Sastry and P. Arroyo, C. Cloet,
G. Gomes, B. Horowitz, J. Hu, L. M. Munoz, J. Musacchio,
M. Ozaki, A. Nilim, C. Pinello, O. Shakernia, A. Sweet,
B. Sinopoli, T. J. Koo, S. Rashid, H. J. Kim, S. Schaffert,
R. Vidal & J. Zhang

Memorandum No. UCB/ERL M99/34

Spring 1999

**HYBRID SYSTEMS: MODELING,
ANALYSIS AND CONTROL
EECS 291E
LECTURE NOTES AND CLASS PROJECTS**

by

John Lygeros, Shankar Sastry and P. Arroyo, C. Cloet,
G. Gomes, B. Horowitz, J. Hu, L. M. Munoz, J. Musacchio,
M. Ozaki, A. Nilim, C. Pinello, O. Shakernia, A. Sweet,
B. Sinopoli, T. J. Koo, S. Rashid, H. J. Kim, S. Schaffert,
R. Vidal & J. Zhang

Memorandum No. UCB/ERL M99/34

Spring 1999

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Hybrid Systems: Modeling, Analysis & Control

John Lygeros & Shankar Sastry

Course Number: 291E

Time: MW, 4:30-6:00

Place: TBA

Spring 1999

Overview

The class will concentrate on the modeling, analysis, control and simulation of hybrid systems. Hybrid systems are dynamical systems with interacting continuous dynamics (modeled, for example, by differential equations) and discrete dynamics (modeled, for example, by automata). Hybrid systems are important in applications such as CAD, real time software, robotics and automation, and transportation, and have recently attracted considerable research attention in the control and computer theory communities. A number of theoretical frameworks have been developed to model hybrid systems, analyze their behavior and synthesize controllers such that the closed loop system satisfies certain specifications. In some cases the theoretical advances have been complemented by automatic verifications tools and simulation languages.

In this class we will present recent advances in this area in a unified framework. The discussion will be motivated by examples and applications (in particular Automated Highway Systems and Air Traffic Management Systems). Even though this is a fairly new research area, there is already a substantial volume of literature dedicated to it. It will not be possible to cover all of it in detail in the class. Some topics will be covered superficially, while others will only be mentioned. Throughout the class we will try to provide references for further details.

Lecture Schedule

1. Modeling (7 lectures)
 - Modeling formalisms
 - Executions, existence and uniqueness conditions
 - Specifications, safety and liveness properties
2. Verification & Analysis (9 lectures)
 - Model checking
 - Deductive approaches
 - Lyapunov stability analysis
 - Verification tools
 - Benchmark problems
3. Control (9 lectures)

- Reachability specifications, the game theoretic approach
- Optimal control
- Other control problems and approaches
- Topics from Automated Highway Systems and Air Traffic Management

4. Simulation (3 lectures)

- Difficulties of numerical simulation
- Currently available simulation tools

Prerequisites and Requirements

An attempt will be made to keep the class as self contained as possible. Some familiarity with the basic concepts of dynamical systems and control will be necessary. Familiarity with automata theory and logic will be helpful.

The grade for the class will be based on homework and a final project. There will be no midterms or final.

We expect to hand out 6 homework assignments, two on modeling, two on verification and two on the control of hybrid systems. The frequency will be roughly one homework every two weeks. The last two weeks there will be no homework to allow more time for the final projects.

The projects can either be in the form of a review of a substantial part of the literature, or, preferably, involve the exploration of novel research ideas. Project topics will be chosen in consultation with the instructors. Projects will be documented in a short report and will be presented to the class during the last week of the semester.

Bibliography

There is no official textbook; the material will be drawn from a number of books and papers. The class will be based mostly on lecture notes. We will try to have the notes ready as a reader at the beginning of the semester.

Course: ee291E, Spring 1999
Title: Hybrid Systems: Modeling, Analysis & Control
Instructors: John Lygeros and Shankar Sastry
CCN:24472
Units:3
Time: MW 4:00-5:30
Location: 241 CORY

Course Summary

The class will concentrate on the modeling, analysis, control and simulation of hybrid systems. Hybrid systems are dynamical systems with interacting continuous dynamics (modeled, for example, by differential equations) and discrete dynamics (modeled, for example, by automata). Hybrid systems are important in applications such as CAD, real time software, robotics and transportation, and have recently been at the center of intense research activity in the control theory and computer theory communities. A number of theoretical methodologies have been proposed to model hybrid systems, analyze their behavior and synthesize controllers such that the closed loop system satisfies certain specifications. In some cases the theoretical advances have been complemented by computational tools for automatic verification, controller synthesis and simulation.

In this course we will present recent advances in this area in a unified framework. The discussion will be motivated by examples and applications (in particular Automated Highway Systems and Air Traffic Management Systems). Even though this is a fairly new research area, there is already a substantial volume of literature dedicated to it. It will not be possible to cover all of it in detail in the class. Some topics will be covered superficially, while others will only be mentioned. Throughout the class we will try to provide references for further details.

Logistics

Presentation: The class will be based mostly on lecture notes. Transparencies will occasionally be used.

Reading Material: There is no official textbook. The material will be drawn from a number of books and papers. References will be provided throughout. This is a research level graduate class, and it will not be possible to cover all the material in detail. Students are expected to follow up on any topics that they find interesting.

Office Hours: John Lygeros will be holding office hours every Monday and Wednesday, 10:00-11:00am, in 269 Cory Hall. Shankar Sastry will be holding office hours every Tuesday and Thursday, 3:00-4:00pm in 253 Cory Hall. The easiest way to contact us outside office hours is by email, at lygeros@eecs.berkeley.edu and sastry@eecs.berkeley.edu.

Web Site: All "electronic" course material will be made available on-line at the web page of John Lygeros <http://robotics.eecs.berkeley.edu/~lygeros/> under ee291E.

Prerequisites and Requirements

Prerequisites: We will try to keep the class as self contained as possible. Some familiarity with the basic concepts of dynamical systems and control will be necessary. Familiarity with automata theory and logic will be helpful.

Homework: We expect to hand out 6 homework assignments, two on modeling, two on verification and two

on the control of hybrid systems. The frequency will be roughly one homework every two weeks. The last two weeks there will be no homework to allow more time for the final projects. The grade for the homework will make up 30% of the overall grade of the class.

Midterms and Final: There will be no examination for this class.

Class Projects: The projects can either be in the form of a review of a substantial part of the literature, or, preferably, involve the exploration of original research ideas. Project topics will be chosen in consultation with the instructors. Students should provide a short (two page) project proposal by March 31. Projects will be documented in a short report and will be presented to the class during the last week of the semester. Joint project proposals will be considered. The grade for the projects will make up 60% of the overall grade of the class.

Class Participation: Attendance and active participation in the class will make up for the remaining 10% of the grade. This, among other things, involves serving as a “designated note-taker” for approximately two lectures during the semester. The duties of the designated note-taker for a particular lecture include:

- Attending the lecture in question.
- Remaining awake for the entire duration of that lecture.
- Carefully taking notes.
- Asking his colleagues and the instructor for additional input.
- Typing the notes (preferably in latex) and turning them in to the instructor within a week.

The collected notes will be distributed to the class. Please sign up for designated note-taker duty at the end of the first lecture.

Lecture Schedule

1. Modeling (7 lectures)

- Modeling formalisms
- Executions, existence and uniqueness conditions
- Specifications, safety and liveness properties

2. Verification & Analysis (9 lectures)

- Model checking
- Deductive approaches
- Lyapunov stability analysis
- Verification tools
- Benchmark problems

3. Control (9 lectures)

- Reachability specifications, the game theoretic approach
- Optimal control
- Other control problems and approaches
- Topics from Automated Highway Systems and Air Traffic Management

4. Simulation (3 lectures)

- Difficulties of numerical simulation
- Currently available simulation tools

ee291E Lecture 1: Introduction

John Lygeros

January 20, 1999

Informal Definitions

- Hybrid Systems: Dynamical systems that require more than one modeling language to characterize their dynamics.
- Here: Systems with interacting continuous and discrete components
- Continuous dynamics:
 - Movement of mechanical systems
 - Linear circuits
 - Chemical reactions
- Discrete dynamics:
 - Collisions in mechanical systems
 - Switches in circuits
 - Valves and pumps in chemical plants

Example 1: Bouncing Ball (Figure 1)

- Motion characterized by height (x_1) and vertical velocity (x_2)
- Continuous changes between bounces
- Discrete changes at bounce times
- Dynamics summarized by:
 - $q \in Q = \{0\}, x = (x_1, x_2) \in X = \mathbb{R}^2$
 - $Init = \{0\} \times \{x \in \mathbb{R}^2 \mid x_1 \geq 0\} \subseteq Q \times X$
 - $\dot{x} = f(x) = (x_2, -g) \in T_x X,$
 - $Inv(q) = (x_1 \geq 0) \subseteq X$
 - $G(q, q) = (x_1 \leq 0) \subseteq X$
 - $R(q, q, x) = (x_1, -cx_2) \subseteq X, c \in [0, 1]$
- One can show that the bouncing ball automaton is **non-blocking** (from any initial condition there exists at least one execution)
- One can show that $(x_1 \geq 0)$ is an **invariant property** of the bouncing ball automaton (by induction on the length of the executions)
- One can show that if $c < 1$ the bouncing ball automaton is **Zeno** (takes an infinite number of discrete transitions in finite time). Note that this casts aspersions on our induction proofs.

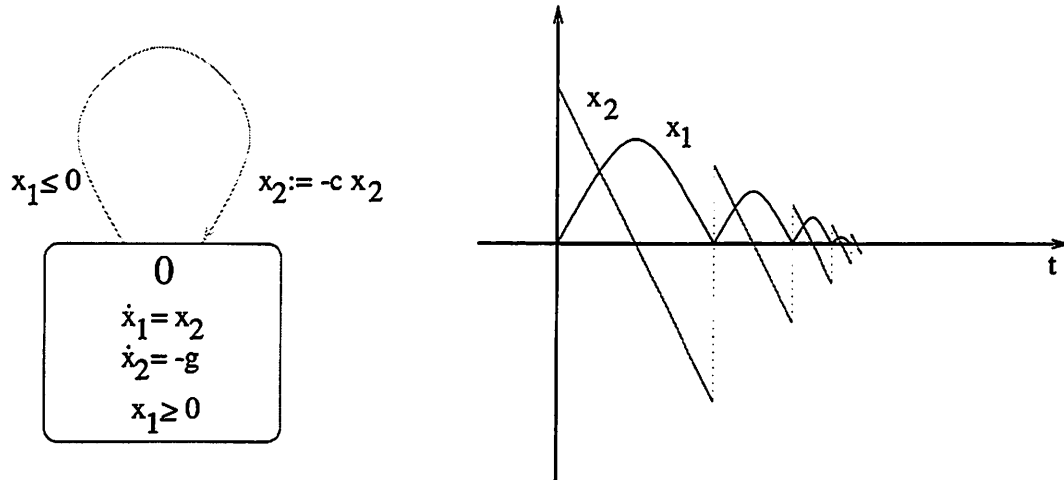


Figure 1: Bouncing ball

Example 2: Thermostat (Figure 2)

- Temperature in a room (x), controlled by switching heater on and off.
- Use a thermostat to regulate x around 75° .
- Assume when heater is off for a long time temperature stabilizes at 0° and when heater on for a long time temperature stabilizes at 100° .
- Thermostat switches heater on between 68° and 70° and off between 80° and 82°
- Dynamics summarized by:
 - $q \in Q = \{0, 1\}$, $x \in X = \mathbb{R}$
 - $Init = \{0, 1\} \times \mathbb{R} \subseteq Q \times X$
 - $\dot{x} = f(q, x) = -x + 100q \in T_x X$,
 - $Inv(0) = (x > 68) \subseteq X$, $Inv(1) = (x < 82) \subseteq X$
 - $G(0, 1) = (x \leq 70) \subseteq X$, $G(1, 0) = (x \geq 80) \subseteq X$
 - $R(0, 1, x) = x \subseteq X$, $R(1, 0, x) = x \subseteq X$
- One can show that the thermostat automaton is **non-deterministic** (for a given initial state there are multiple possible executions).

Example 3: Automated Highway Systems (AHS)

- Example of a *large scale, multi-agent, distributed* system.
- **Objective:** increase *throughput* (in number of vehicles per lane per hour) of a highway system without building new highways (because real estate is expensive) and without compromising passenger *safety* and *comfort*.
- **Problem 1:** How are throughput, safety, comfort to be interpreted in the presence of faults?
- **Temporary Solution 1:** Restrict our attention to *normal* operation. Will have to come back to design controllers for fault handling and devise a methodology for assessing their effectiveness.

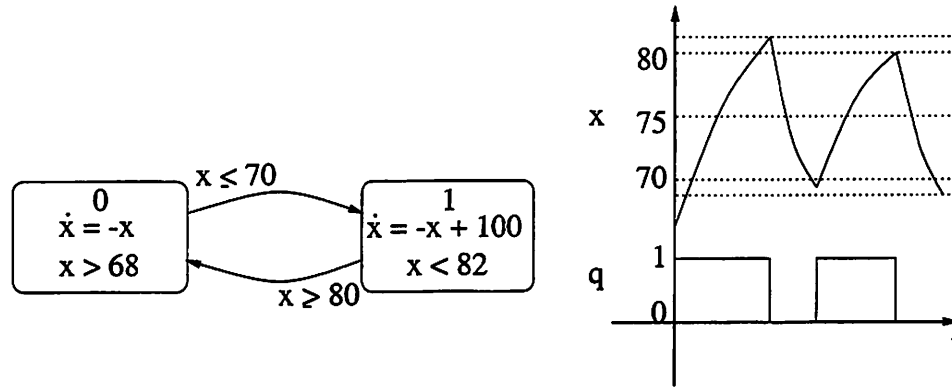


Figure 2: The thermostat system

- **Problem 2:** *Emergent behavior* for the overall system to be achieved by control of individual agents. Potential conflicts of interest: it is unclear whether greedy controller to minimize travel time of individual vehicle leads to maximum throughput of the system.
- **Temporary Solution 2:** Give up on overall optimality and go for a “heuristic” solution. Will have to come back to assess performance of overall system, and tune our heuristics accordingly.
- Requirement for throughput (high speed and close following) in conflict with requirement for safety (low speed and large spacing).
- Platooning scenario achieves a compromise. Vehicles travel in tightly spaced groups (*platoons*).
- Intra platoon spacing is small (1-5 meters), therefore:
 - Potential throughput is much higher.
 - Intra-platoon collisions (if any) are likely to be at low relative velocities (hence safe).
- Inter-platoon spacing is large, to prevent disturbances from propagating between platoons.
- Automatic control needed. Hierarchy proposed in [1] shown in Figure 3.
- Continuous “state” (for each vehicle): position and velocity (acceleration, manifold pressure, ...).
- Discrete “state” (for each vehicle): position in the platoon and lane number (origin, destination, ...).
- **Problem 3:** “State” of the AHS system “product” of states of individual vehicles. Number of vehicles very large and dynamically changing. Difficult to even simulate large scale systems efficiently.
- **Temporary Solution 3:** Focus on neighborhoods of vehicles. Achieved by appropriate coordination protocols. Problem of simulation solved by appropriate programming language [2].
- Transition between discrete states through maneuvers (Figure 4). Necessary coordination provided by communication protocols.
- For each discrete state and each transition, invoke a different controller to control the continuous states.
- **Problem 4:** Can one show that composite system is *safe* and *efficient*?
- See [1] for an introduction to AHS control architecture, [3] for a discussion of possible safety problems due to the hybrid nature of the problem and [4] for one possible solution.

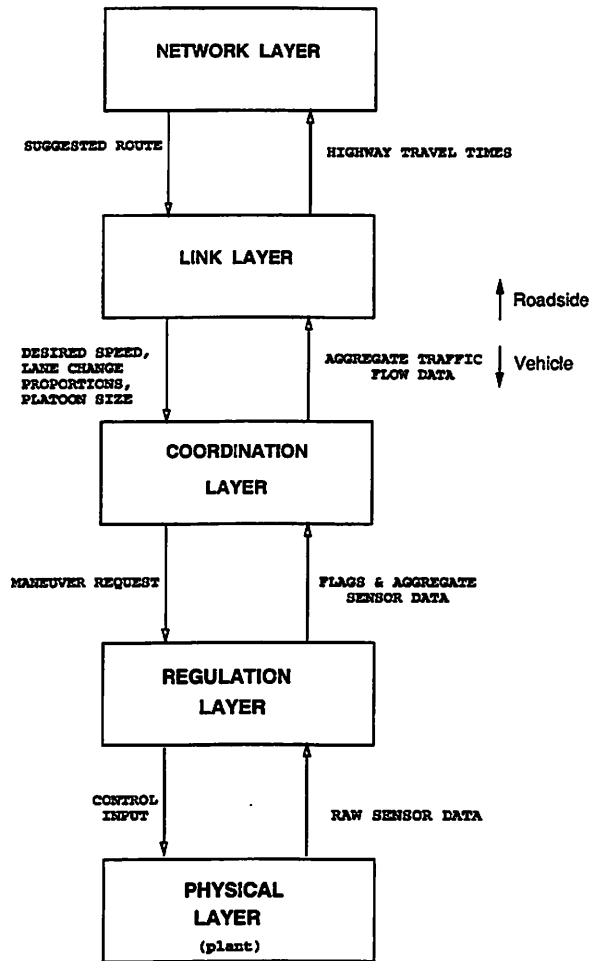


Figure 3: AHS Control Hierarchy

Classification of Hybrid Behavior

- Continuous systems with phased operation
 - Bouncing ball
 - Walking robots
 - Circuits with diodes
- Continuous systems controlled by discrete inputs
 - Thermostat
 - Circuits with switches
 - Chemical plants with valves, pumps
 - Systems controlled by digital computers
- Control Modes
 - Leader mode vs. follower mode for a vehicle on an AHS
 - Aircraft autopilot modes: maintain altitude, maintain constant airspeed, maintain angle of attack, descend at a fixed rate.

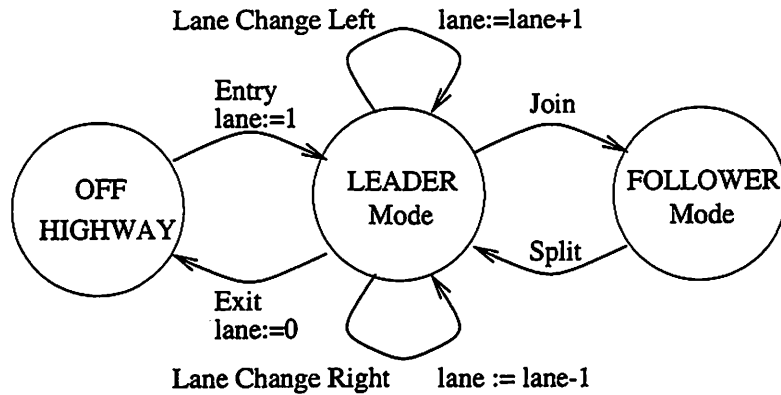


Figure 4: Discrete state of an AHS vehicle

- Normal vs. degraded modes of operation for fault handling
- Coordinating processes
 - Communication protocols for AHS maneuvers.
 - Conflict resolution for aircraft.
 - Multi-agent systems: typically continuous controllers to optimize performance of individual agents, coordination among agents to resolve conflicts.

Note: “hybrid” phenomena typically introduced for modeling convenience.

References

- [1] Pravin Varaiya, “Smart cars on smart roads: problems of control”, *IEEE Transactions on Automatic Control*, vol. AC-38, no. 2, pp. 195–207, 1993.
- [2] Akash Deshpande, Aleks Gollu, and Luigi Semenzato, “The SHIFT programming language and run-time system for dynamic networks of hybrid automata”, Tech. Rep. UCB-ITS-PRR-97-7, Institute of Transportation Studies, University of California, Berkeley, 1997.
- [3] Datta N. Godbole, John Lygeros, and Shankar Sastry, “Hierarchical hybrid control: an IVHS case study”, in *Hybrid Systems II*, P. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, Eds., number 999 in LNCS, pp. 166–190. Springer Verlag, 1995.
- [4] John Lygeros, Datta N. Godbole, and Shankar Sastry, “Verified hybrid controllers for automated vehicles”, *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 522–539, 1998.

ee291E Lecture 2: Mathematical Background, Continuous and Discrete Systems

Bruno Sinopoli & John Lygeros

January 25, 1999

Notation

Basics

- \exists “there exists”, \forall “for all”, $!$ “unique”, \ni “such that”, iff “if and only if”.
- \wedge logical “AND”, \vee logical “OR”, \neg logical “NOT”.
- Subsets of \mathbb{R} : $[a, b)$, $[a, b]$, etc.

Variables

- A set of *variables*, Y , is a set of symbols.
- The set of *valuations*, \mathbf{Y} , of a set of variables, Y , is the set of all possible values these variables can assume.
- A valuation, y , can be thought of as a map $y : Y \rightarrow \mathbf{Y}$.
- We will be concerned mainly with two types of variables:
 - *Continuous* variables, X , with $x \in \mathbf{X} = \mathbb{R}^n$, for some $n \in \mathbb{N}$.
 - *Discrete* variables, Q , with $q \in \mathbf{Q} \subseteq \mathbb{Z}$.

Topology

- The discussion in this section is terse, just enough to make the treatment of subsequent material formal. Students should be familiar with most of these concepts, at least for Euclidean spaces. See [1] for a more formal treatment.
- **Definition:** A *topological space*, (Y, \mathcal{T}) , consists of a set, Y , and a collection, \mathcal{T} , of subsets of Y , such that
 1. $\emptyset \in \mathcal{T}$.
 2. $Y \in \mathcal{T}$.
 3. \mathcal{T} is closed under arbitrary unions.
 4. \mathcal{T} is closed under finite intersections.
- \mathcal{T} is called a *topology* on Y . The elements of \mathcal{T} are called the *open sets* of Y .

- Given a set Y , there are many possible topologies on Y . Given two topologies \mathcal{T} and \mathcal{T}' on Y , if $\mathcal{T} \subseteq \mathcal{T}'$, then \mathcal{T}' is *finer* than \mathcal{T} .
- A *basis*, \mathcal{B} for a topology on Y is a collection of subsets of Y such that:
 1. $\forall y \in Y$, there exists $B \in \mathcal{B}$ with $y \in B$.
 2. For each pair $B_1, B_2 \in \mathcal{B}$ and $\forall y \in B_1 \cap B_2, \exists B \in \mathcal{B}$ with $y \in B \subseteq B_1 \cap B_2$.
- The topology, \mathcal{T} , *generated* on Y by a basis, \mathcal{B} , consists of all sets $U \subseteq Y$ such that for each $y \in U$ there exists $B \in \mathcal{B}$ such that $y \in B \subseteq U$. One can show that \mathcal{T} is in fact the collection of all unions of sets in \mathcal{B} .
- A subset $Y' \subseteq Y$ of a topological space (Y, \mathcal{T}) , inherits the *subset topology*, \mathcal{T}' from Y , where $U' \in \mathcal{T}'$ if and only if there exists $U \in \mathcal{T}$ with $U' = U \cap Y'$.
- The product, $X \times Y$, of two topological spaces, (Y, \mathcal{T}) and (X, \mathcal{S}) , is given the *product topology*, generated by the basis $\mathcal{B} = \{V \times U : V \in \mathcal{T} \wedge U \in \mathcal{S}\}$.
- **Definition.** A *metric* on a set Y is a map:

$$d : Y \times Y \rightarrow \mathbb{R} \quad (1)$$

with the following properties:

1. $\forall x, y \in Y, d(x, y) \geq 0$ and $d(x, y) = 0$ only if $x = y$
 2. $\forall x, y \in Y, d(x, y) = d(y, x)$
 3. $\forall x, y, z \in Y, d(x, y) + d(y, z) \geq d(x, z)$
- \mathbb{R}^n is given the Euclidean metric topology, generated by open balls in the Euclidean metric. Recall that the Euclidean metric is given by:

$$d(x, y) = \left(\sum_{i=1}^n (x_i - y_i)^2 \right)^{\frac{1}{2}}$$

and that the open ball centered at $x \in \mathbb{R}^n$ with radius $r > 0$ is the set:

$$B(x, r) = \{y \in \mathbb{R}^n : d(x, y) < r\}$$

- Countable and finite sets are given the discrete topology, generated by the basis consisting of all the singletons. In other words, if $Q = \{q_0, q_1, \dots\}$ then:

$$\mathcal{B} = \{\{q_0\}, \{q_1\}, \dots\}$$

- **Proposition:** In the discrete topology every subset is open and closed.
- For a topological space (Y, \mathcal{T}) and a subset $K \subseteq Y$ we denote by:
 - $|K|$ the *cardinality* of K (number of elements of K)
 - K^c the *complement* of K ($K^c = Y \setminus K$)
 - K° the *interior* of K (largest open set contained in K)
 - \overline{K} the *closure* of K (smallest closed set containing K)
 - ∂K the *boundary* of K ($\partial K = \overline{K} \setminus K^\circ$)
 - 2^K the *power set* of K (set of all subsets of K)

Functions & Continuity

- Consider two topological spaces, (Y, \mathcal{T}) and (X, \mathcal{S}) , and a function $f : Y \rightarrow X$. f is called *continuous* if for all $V \in \mathcal{S}$, $f^{-1}(V) = \{y \in Y : f(y) \in V\} \in \mathcal{T}$.
- Consider two topological spaces Y and X , and a bijection $f : Y \rightarrow X$. f is called a *homeomorphism* if both the function f and its inverse $f^{-1} : X \rightarrow Y$ are continuous.
- For a topological space (Y, \mathcal{T}) , a subset $K \subseteq Y$ and an interval $U \subseteq \mathbb{R}$:
 - We denote by $C^0(U, K)$ the set of continuous functions from U to K .

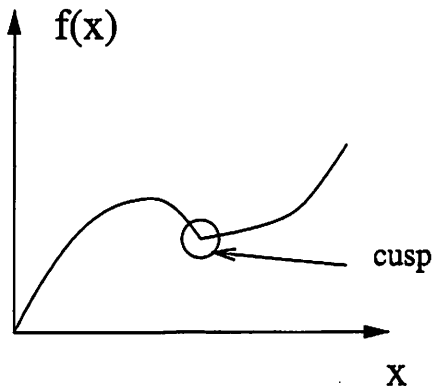


Figure 1: An example of function $f \in C^0(\mathbb{R}, \mathbb{R})$

- If $Y = \mathbb{R}^n$, we denote by $C^i(U, K)$, $i = 0, 1, \dots, \infty$ the set of i times differentiable functions from U to K and by $C^\omega(U, K)$ the set of analytic functions from U to K .
- We denote by $PC^0(U, K)$ the set of piecewise continuous functions from U to K , i.e. the set of functions which are continuous for all but a finite number of points in every bounded subset of U and for which both the left and right limits exist and are finite at all discontinuity points.

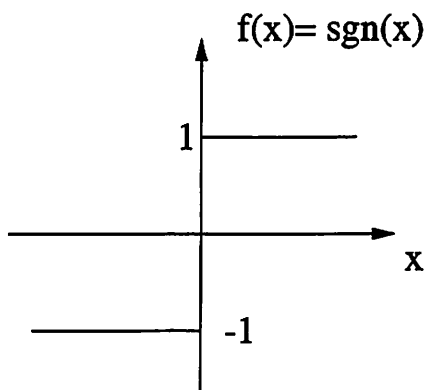


Figure 2: A piecewise continuous function

- If $Y = \mathbb{R}^n$, we denote by $PC^i(U, K)$, $i = 0, 1, \dots, \infty$ the set of $i - 1$ times differentiable functions from U to K whose i^{th} derivative is piecewise continuous and by $PC^\omega(U, K)$ the set of piecewise analytic functions from U to K .
- For a topological space (Y, \mathcal{T}) , a subset $K \subseteq Y$ and a set $U \subseteq \mathbb{Z}$ we denote by:
 - K^* the set of all functions from U to K , where U is finite.
 - K^ω the set of all functions from U to K , where U is infinite.

Continuous Dynamical Systems

- For a formal treatment see [2], chapter 3 (available at <http://robotics.eecs.berkeley.edu/~lygeros/>), [3] chapter 2, or your favorite ODE book.

- Consider $x : \mathbb{R} \rightarrow \mathbf{X} = \mathbb{R}^n$ and the ordinary differential equation:

$$\frac{dx}{dt}(t) = f(x(t), t), \quad x(0) = x_0 \quad (2)$$

where $f : \mathbb{R}^n \times \mathbb{R}_+ \rightarrow \mathbb{R}^n$ is a *vector-field*.

- An *execution* of (2) on $[0, T] \subset \mathbb{R}_+$ in the sense of Caratheodory is a piecewise differentiable function $x : [0, T] \rightarrow \mathbb{R}^n$ such that for all $t \in [0, T]$:

$$x(t) = x_0 + \int_0^t f(x(\tau), \tau) d\tau \quad (3)$$

- **Questions:**

- Do solutions in the sense of Caratheodory exist?
- Over what intervals $[0, T]$ can they be defined?
- Are they unique?
- Are they well behaved?

- **Theorem 1:** (Local Existence and Uniqueness) Assume $f(x, t)$ is piecewise continuous in t and there exist $L, r, T > 0$ such that $f(x, t)$ satisfies the Lipschitz condition:

$$\|f(t, x) - f(t, y)\| \leq L\|x - y\| \quad (4)$$

for all $x, y \in B = \{x \in \mathbb{R}^n : \|x - x_0\| \leq r\}$ and for all $t \in [0, T]$. Then there exists $\delta > 0$ such that the system (2) has a unique execution (3) on $[0, \delta]$.

- **Theorem 2:** (Global Existence and Uniqueness) Assume $f(x, t)$ is piecewise continuous in t and there exist $L, h > 0$ such that $f(x, t)$ satisfies the conditions:

$$\begin{aligned} \|f(t, x) - f(t, y)\| &\leq L\|x - y\| \\ \|f(t, x_0)\| &\leq h \end{aligned} \quad (5)$$

for all $x, y \in \mathbb{R}^n$ and for all $t \in [0, T]$. Then the system (2) has a unique execution (3) on $[0, T]$.

- **Theorem 3:** (Continuous Dependence on Initial Conditions) Assume $f(x, t)$ satisfies the conditions of Theorem 2, and let x, y be two executions of (2) starting at x_0 and y_0 respectively. Then for all $\epsilon, T > 0$ there exists $\delta > 0$ such that:

$$\|x_0 - y_0\| \leq \delta \implies \|x(t) - y(t)\| \leq \epsilon \quad (6)$$

for all $t \in [0, T]$.

- In fact $\|x(t) - y(t)\| \leq \|x_0 - y_0\|e^{Lt}$, so we can choose $\delta \leq \epsilon/e^{LT}$.

- **Remarks :**

1. The theorems provide conditions for a continuous system to be *non-blocking* (in the sense that solutions exist locally), *deterministic* (in the sense that solutions are unique) and *non-zeno* (in the sense that solutions can be extended over arbitrarily long horizons). As we shall soon see this is not necessarily the case for hybrid systems.

2. Conditions are “tight”, in the sense that there exist systems violating the conditions that fail to have unique executions.

– f discontinuous in x . Consider:

$$\dot{x} = -\text{sgn}(x)$$

The solution starting at $x_0 = 0$ is undefined for all times. In fact, all solutions end up at $x = 0$ in finite time and are undefined thereafter.

– f continuous but not Lipschitz in x . Consider:

$$\dot{x} = x^{1/3}$$

Both $x(t) = (2t/3)^{3/2}$ and $x(t) = 0$ are solutions for $x_0 = 0$.

– f Lipschitz but not globally Lipschitz in x . Consider:

$$\dot{x} = -x^2$$

The solution with $x_0 = -1$ is $x(t) = 1/(t - 1)$ and is defined only on $[0, T]$ for $T < 1$.

3. Theorem 3 (and a similar theorem for continuity of solutions with respect to parameters) allow one to simulate continuous systems numerically. This can be a problem for hybrid systems (to be discussed in the tutorial by Karl Johansson).
4. Piecewise continuity in t allows discontinuous inputs in control systems. A *control system* can be thought of as a continuous dynamical system:

$$\frac{dx}{dt}(t) = f(x(t), u(t)), \quad x(0) = x_0 \quad (7)$$

where $u : \mathbb{R} \rightarrow \mathbf{U} = \mathbb{R}^m$, $m \in \mathbb{N}$. If f is continuous in u and satisfies the Lipschitz assumptions of Theorems 1 and/or 2 with respect to x , then for each u piecewise continuous in time, the system (7) has a unique execution.

Discrete Systems

- For a formal treatment see [4] chapter 2, or [5] chapter 2.
- Restrict our attention to *finite automata*.
- Finite automata are perhaps the simplest models of computation. Other models (Turing machines, push-down automata, Petri nets) can exhibit more interesting behaviors.
- A *finite automaton*, M is a “tuple”, $(\mathbf{Q}, \Sigma, \Delta, q_0, F)$, where:
 - \mathbf{Q} is a finite *set of states*
 - Σ is a set of *input symbols*
 - $\Delta \subset \mathbf{Q} \times \Sigma \times \mathbf{Q}$ is a *transition relation*
 - $q_0 \in \mathbf{Q}$ is the *initial state*
 - $F \subseteq \mathbf{Q}$ is a set of *final states*
- Σ^* set of strings of finite length of elements of Σ . Define string of length 0 to be ϵ .
- M *accepts* a string $s \in \Sigma^*$, with $|s| = n$, if there exists a sequence of states $q \in \mathbf{Q}^*$, with $|q| = n + 1$ such that:
 - $q[0] = q_0$
 - For $i = 0, 1, \dots, n$, $(q[i], s[i], q[i + 1]) \in \Delta$

– $q[n+1] \in F$.

- The *language accepted by M* , $L(M)$ is the set of all strings accepted by M .
- Two automata, M_1 and M_2 , are *equivalent* if $L(M_1) = L(M_2)$.
- May be blocking (not “accept” certain symbols at certain states) and non-deterministic (take different transitions from the same state in response to the same symbol).
- Δ can also be given as a partial map $\delta : Q \times \Sigma \rightarrow 2^Q$ given by:

$$\delta(q, \sigma) = \{q' \in Q : (q, \sigma, q') \in \Delta\} \quad (8)$$

- A finite automaton is *deterministic* if δ is a function, i.e. $|\delta(q, \sigma)| = 1$ for all $q \in Q$ and all $\sigma \in \Sigma$. In this case, δ is called the *transition function*.
- **Theorem 4** (Equivalence of Deterministic and Non-deterministic Finite Automata) For all finite automata there exists and equivalent deterministic finite automaton.
- **Remark:** “Determinization” may increase the number of states from $|Q|$ to $2^{|Q|}$.
- **Example:** Consider the finite automaton $M = (Q, \Sigma, \Delta, q_0, F)$ for Figure 3.

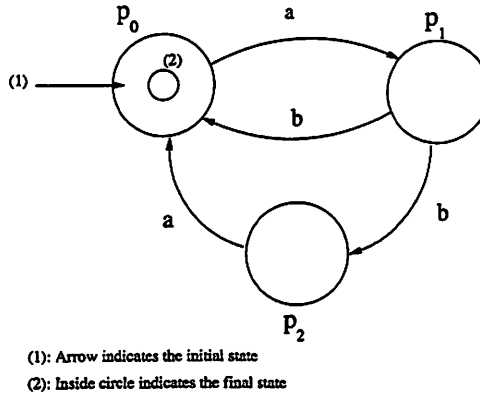


Figure 3:

- In this case:
 - * $Q = \{q_0, q_1, q_2\}$
 - * $\Sigma = \{a, b\}$
 - * $\Delta = \{(q_0, a, q_1), (q_1, b, q_0), (q_1, b, q_2), (q_2, a, q_0)\}$
 - * $q_0 = \{q_0\}$
 - * $F = \{q_0\}$
- The language accepted by this automaton is :
 $L(M) = \{\epsilon, ab, aba, abab, abaaba, \dots\} = ((ab)^*(aba)^*)^* \subseteq \Sigma^*$
- The transition relation in this example is not a function. Given the current state q_1 and the input symbol b there are two possible next states. The model itself does not determine the choice. Therefore the automaton is non deterministic. We said that two automata M, N are equivalent if and only if $L(M) = L(N)$. This definition, combined with the result provided by Theorem 4, states that there is no difference between deterministic and non deterministic automata. The fact that a nondeterministic automaton has, in general, considerably less states, may represent an advantage over the deterministic one because it allows one to express a language in more compact form.

References

- [1] J. R. Munkres, *Topology, a First Course*, Prentice Hall, 1975.
- [2] Shankar Sastry, *Nonlinear Systems: Analysis, Stability and Control*, Springer-Verlag, 1999.
- [3] Hassan B. Khalil, *Nonlinear Systems*, MacMillan, 1992.
- [4] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.
- [5] Harry R. Lewis and Christos H. Papadimitriou, *Elements of the Theory of Computation*, Prentice Hall, 1981.

ee291E Lecture 3: Modeling Autonomous Hybrid Systems

Arnab Nilim & John Lygeros

January 27, 1999

In this lecture we introduce autonomous hybrid automata, that is hybrid dynamical systems with no inputs.

Definition 1 (Hybrid Automaton) *A hybrid automaton H is a collection $H = (Q, X, \text{Init}, f, I, E, G, R)$, where*

- Q is a set of discrete variables and Q is countable;
- X is a set of continuous variables;
- $\text{Init} \subseteq Q \times X$ is a set of initial states;
- $f : Q \times X \rightarrow TX$ is a vector field;
- $\text{Inv} : Q \rightarrow 2^X$ assigns to each $q \in Q$ an invariant set;
- $E \subset Q \times Q$ is a collection of discrete transitions;
- $G : E \rightarrow 2^X$ assigns to each $e = (q, q') \in E$ a guard; and
- $R : E \times X \rightarrow 2^X$ assigns to each $e = (q, q') \in E$ and $x \in X$ a reset relation.

Remarks:

1. We refer to $(q, x) \in Q \times X$ as the *state* of H .
2. To avoid technicalities with continuous dynamics we impose the following assumption:

Assumption 1 *Assume $f(q, x)$ is globally Lipschitz continuous in its second argument.*

Example: Consider the thermostat system from Lecture 1.

- Discrete variable $Q = \{q\}$, $q \in \{\text{ON}, \text{OFF}\} = Q$
- Continuous variable, $X = \{x\}$, $x \in \mathbb{R} = X$
- Initial states, $\text{Init} = Q \times X$
- Vector field, $f(q, x) = \begin{cases} -x & \text{if } q = \text{OFF} \\ -x + 100 & \text{if } q = \text{ON} \end{cases}$
- Invariant, $\text{Inv}(q) = \begin{cases} \{x \in \mathbb{R} : x > 68\} & \text{if } q = \text{OFF} \\ \{x \in \mathbb{R} : x < 82\} & \text{if } q = \text{ON} \end{cases}$

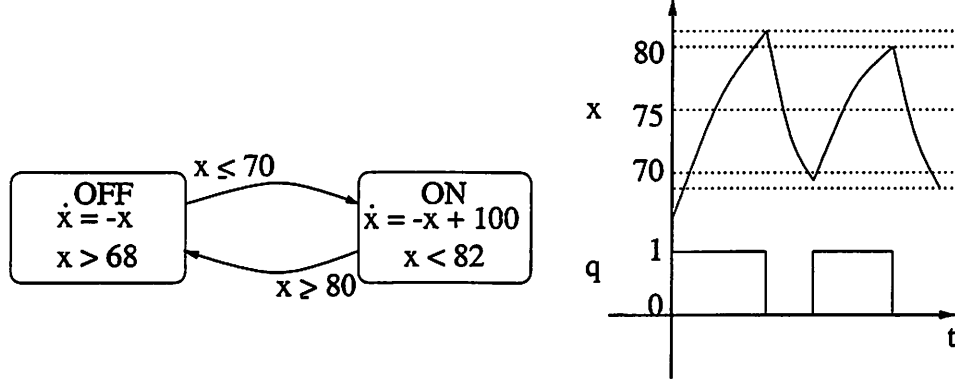


Figure 1: The thermostat system

- Discrete transitions, $E = \{(ON, OFF), (OFF, ON)\}$
- Guard, $G(e) = \begin{cases} \{x \in \mathbb{R} : x \geq 80\} & \text{if } e = (ON, OFF) \\ \{x \in \mathbb{R} : x \leq 70\} & \text{if } e = (OFF, ON) \end{cases}$
- Reset relation, $R(e, x) = \{x\}$

The state of hybrid automata evolves by interleaving pieces of continuous execution with discrete transitions. The times over which the execution of the hybrid automaton can be defined are therefore of the form:

Definition 2 (Hybrid Time Trajectory) A hybrid time trajectory τ is a finite or infinite sequence of intervals of the real line, $\tau = \{I_i\}$, $i \in \mathbb{N}$, satisfying the following conditions:

- I_i is closed, unless τ is a finite sequence and I_i is the last interval in which case it is left closed but can be right open.
- Let $I_i = [\tau_i, \tau'_i]$. Then for all i , $\tau_i \leq \tau'_i$ and for $i > 0$, $\tau_i = \tau'_{i-1}$.

Remarks:

1. Hybrid time trajectories can extend to “infinity” if τ is an infinite sequence or if it is a finite sequence ending with an interval of the form $[\tau_N, \infty)$.
2. We denote by \mathcal{T} the set of all hybrid time trajectories.
3. Each $\tau \in \mathcal{T}$ is fully ordered by the relation \prec , which, for $t \in [\tau_i, \tau'_i] \in \tau$ and $t' \in [\tau_j, \tau'_j] \in \tau$ is defined as $t \prec t'$ if $i < j$ or $i = j$ and $t < t'$.
4. For $t \in \mathbb{R}$ and $\tau \in \mathcal{T}$ we use $t \in \tau$ as a shorthand notation for “there exists a j such that $t \in [\tau_j, \tau'_j] \in \tau$ ”.
5. For a topological space K and a $\tau \in \mathcal{T}$, we use $k : \tau \rightarrow K$ as a shorthand notation for a map assigning a value from K to each $t \in \tau$.
6. We say $\tau = \{I_i\}_{i=0}^N \in \mathcal{T}$ is a *prefix* of $\tau' = \{J_i\}_{i=0}^M \in \mathcal{T}$ and write $\tau \leq \tau'$ if either they are identical or τ is finite, $M \geq N$, $I_i = J_i$ for all $i = 0, \dots, N-1$ and $I_N \subseteq J_N$.
7. We say τ is a *strict prefix* of τ' and write $\tau < \tau'$, if $\tau \leq \tau'$ and $\tau \neq \tau'$.
8. The prefix relation is a partial order on \mathcal{T} .

Definition 3 (Execution) An execution χ of a hybrid automaton H is a collection $\chi = (\tau, q, x)$ with $\tau \in \mathcal{T}$, $q : \tau \rightarrow \mathbf{Q}$, and $x : \tau \rightarrow \mathbf{X}$, satisfying

- *Initial condition:* $(q(\tau_0), x(\tau_0)) \in \text{Init}$;
- *Continuous evolution:* for all i with $\tau_i < \tau'_i$, x and q are continuous over $[\tau_i, \tau'_i]$ and for all $t \in [\tau_i, \tau'_i]$, $x(t) \in \text{Inv}(q(t))$ and $\frac{d}{dt}x(t) = f(q(t), x(t))$; and
- *Discrete Evolution:* for all i , $e = (q(\tau'_i), q(\tau_{i+1})) \in E$, $x(\tau'_i) \in G(e)$, and $x(\tau_{i+1}) \in R(e, x(\tau'_i))$.

Remarks:

1. For an execution $\chi = (\tau, q, x)$ we use $(q_0, x_0) = (q(\tau_0), x(\tau_0))$ to denote the initial state of χ .
2. We say χ is a prefix of χ' (write $\chi \leq \chi'$) if $\tau \leq \tau'$ and $(q(t), x(t))(t) = (q'(t), x'(t))$ for all $t \in \tau$.
3. We say χ is a strict prefix of χ' (write $\chi < \chi'$) if $\chi \leq \chi'$ and $\chi \neq \chi'$.
4. We say an execution is maximal if it is not a strict prefix of any other execution.
5. The prefix relation defines a partial order on the set of executions.
6. The set of executions is prefix closed.
7. As for finite automata (Lecture 2), the discrete aspect of the dynamics can again be characterized by a transition relation:

$$\begin{aligned} \Delta &\subseteq (\mathbf{Q} \times \mathbf{X}) \times (\mathbf{Q} \times \mathbf{X}) \\ \Delta &= \{((q, x), (q', x')) : (q, q') \in E, x \in G((q, q')), x' \in R((q, q'), x)\} \cup \\ &\quad \{((q, x), (q, x)) : x \in \text{Inv}(q)\} \end{aligned} \quad (1)$$

The transition relation description is more compact notationally, but may be cumbersome when modeling real systems. The two descriptions are equivalent if one is interested only in *reachability* properties. The description of Definition 0.1 is clearer if one also wants to study *liveness* properties (such as the Zeno property).

Definition 4 (Types of Execution) An execution $\chi = (\tau, q, x)$ of a hybrid automaton H is called:

- *Finite*, if τ is a finite sequence ending in a right closed interval.
- *Infinite*, if τ is an infinite sequence, or $\sum_i (\tau'_i - \tau_i) = \infty$.
- *Admissible*, if it is either finite, or $\sum_i (\tau'_i - \tau_i) = \infty$.
- *Zeno*, if it is infinite and not admissible.

Remarks:

1. A state is called *reachable* by H if it is the last state of a finite execution of H .
2. We use $\text{Reach}(H) \subseteq \mathbf{Q} \times \mathbf{X}$ to denote the reachable set of states of H .
3. For a Zeno execution, we define the *Zeno time* as $\tau_\infty = \sum_i (\tau'_i - \tau_i)$.
4. We use $\mathcal{H}_{(q_0, x_0)}$ to denote the set of executions of H with initial condition $(q_0, x_0) \in \text{Init}$.
5. We use $\mathcal{H}_{(q_0, x_0)}^\infty$ to denote the set of infinite executions of H with initial condition $(q_0, x_0) \in \text{Init}$.
6. We use \mathcal{H} to denote the set of all executions of H . Clearly:

$$\mathcal{H} = \bigcup_{(q_0, x_0) \in \text{Init}} \mathcal{H}_{(q_0, x_0)} \quad (2)$$

ee291E Lecture 4: Existence of Executions

Gabriel Gomes & John Lygeros

February 1, 1999

1 Classification of Executions

Recall that:

Definition 1 (Execution) An execution χ of a hybrid automaton H is a collection $\chi = (\tau, q, x)$ with $\tau \in \mathcal{T}$, $q : \tau \rightarrow \mathbf{Q}$, and $x : \tau \rightarrow \mathbf{X}$, satisfying

- *Initial condition:* $(q(\tau_0), x(\tau_0)) \in \text{Init}$
- *Continuous evolution:* for all i with $\tau_i < \tau'_i$, x and q are continuous over $[\tau_i, \tau'_i]$ and for all $t \in [\tau_i, \tau'_i]$, $x(t) \in \text{Inv}(q(t))$ and $\frac{d}{dt}x(t) = f(q(t), x(t))$
- *Discrete Evolution:* for all i , $e = (q(\tau'_i), q(\tau_{i+1})) \in E$, $x(\tau'_i) \in G(e)$, and $x(\tau_{i+1}) \in R(e, x(\tau'_i))$.

Remarks:

1. For an execution $\chi = (\tau, q, x)$ we use $(q_0, x_0) = (q(\tau_0), x(\tau_0))$ to denote the initial state of χ .
2. We say χ is a *prefix* of χ' (write $\chi \leq \chi'$) if $\tau \leq \tau'$ and $(q(t), x(t))(t) = (q'(t), x'(t))$ for all $t \in \tau$.
3. We say χ is a *strict prefix* of χ' (write $\chi < \chi'$) if $\chi \leq \chi'$ and $\chi \neq \chi'$.
4. We say an execution is *maximal* if it is not a strict prefix of any other execution.
5. The prefix relation defines a partial order on the set of executions.
6. The set of executions is prefix closed (i.e. every prefix of an execution is an execution)
7. The discrete aspect of the dynamics can again be characterized by a transition relation:

$$\begin{aligned} \Delta &\subseteq (\mathbf{Q} \times \mathbf{X}) \times (\mathbf{Q} \times \mathbf{X}) \\ \Delta &= \{((q, x), (q', x')) : (q, q') \in E, x \in G((q, q')), x' \in R((q, q'), x)\} \cup \\ &\quad \{((q, x), (q, x)) : x \in \text{Inv}(q)\} \end{aligned} \quad (1)$$

The transition relation description is more compact notationally, but may be cumbersome when modeling real systems. The two descriptions are equivalent if one is interested only in *reachability* properties. The description of Definition 1 of Lecture 3 is clearer if one also wants to study *liveness* properties (such as the Zeno property).

Definition 2 (Types of Execution) An execution $\chi = (\tau, q, x)$ of a hybrid automaton H is called:

- *Finite*, if τ is a finite sequence ending in a right closed interval.

- *Infinite*, if τ is an infinite sequence, or $\sum_i(\tau'_i - \tau_i) = \infty$.
- *Admissible*, if it is either finite, or $\sum_i(\tau'_i - \tau_i) = \infty$.
- *Zeno*, if it is infinite and not admissible.

Remarks:

1. All infinite executions are maximal.
2. For a Zeno execution, $\{\tau_i\}$ is an infinite, converging sequence.
3. For a Zeno execution, we define the *Zeno time* as $\tau_\infty = \sum_i(\tau'_i - \tau_i)$.
4. We use $\mathcal{H}_{(q_0, x_0)}$ to denote the set of executions of H with initial condition $(q_0, x_0) \in \text{Init}$.
5. We use $\mathcal{H}_{(q_0, x_0)}^\infty$ to denote the set of infinite executions of H with initial condition $(q_0, x_0) \in \text{Init}$.
6. We use $\mathcal{H}_{(q_0, x_0)}^M$ to denote the set of maximal executions of H with initial condition $(q_0, x_0) \in \text{Init}$.
7. We use \mathcal{H} to denote the set of all executions of H .

Clearly:

$$\mathcal{H} = \bigcup_{(q_0, x_0) \in \text{Init}} \mathcal{H}_{(q_0, x_0)} \quad (2)$$

Unlike conventional continuous dynamical systems, the interpretation is that an automaton H *accepts* (as opposed to generates) an execution $\chi = (\tau, q, x)$. This allows one to consider hybrid automata that accept no executions for some initial states (*blocking*), accept multiple executions for the same initial state (*non-deterministic*), or do not accept executions over arbitrarily long time horizons (*Zeno*). Some of these properties (non-determinism) may facilitate modeling and analysis. Others are mostly a nuisance, and are a consequence of our desire to make the modeling formalism powerful enough to capture a wide class of hybrid phenomena.

2 Existence Conditions

We prove some facts about the existence and uniqueness of executions for a restricted class of hybrid systems where the vector field “crosses” the boundary of the invariant set. Some more examples will be given as homework. To eliminate any problems that may arise strictly as a result of continuous evolution we introduce the following assumption:

Assumption 1 *Assume $f(q, x)$ is globally Lipschitz continuous in its second argument.*

Definition 3 (Reachable State) *A state $(\hat{q}, \hat{x}) \in \mathbf{Q} \times \mathbf{X}$ is called reachable by H if there exists a finite execution $\chi = (\tau, q, x)$ with $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^N$ and $(q(\tau'_N), x(\tau'_N)) = (\hat{q}, \hat{x})$.*

We use $\text{Reach}(H) \subseteq \mathbf{Q} \times \mathbf{X}$ to denote the set of all states reachable by H .

Definition 4 (Non-Blocking and Deterministic Automaton) *A hybrid automaton, H , is called non-blocking if $\mathcal{H}_{(q_0, x_0)}^\infty$ is non-empty for all $(q_0, x_0) \in \text{Init}$. A hybrid automaton is called deterministic if $\mathcal{H}_{(q_0, x_0)}^M$ contains at most one element for all $(q_0, x_0) \in \text{Init}$.*

In other words:

$$H \text{ is non-blocking if } |\mathcal{H}_{(q_0, x_0)}^M| \geq 1 \text{ for all } (q_0, x_0) \in \text{Init}$$

(i.e. there exists at least one infinite execution for every initial condition) and

$$H \text{ is deterministic if } |\mathcal{H}_{(q_0, x_0)}^\infty| \leq 1 \text{ for all } (q_0, x_0) \in \text{Init}$$

(i.e. there exists at most one maximal execution for every initial condition)

Assume f is analytic in its second argument. For a function $\sigma : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbb{R}$, also analytic in its second argument, inductively define the *Lie derivatives* of σ along f , $L_f^m \sigma : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbb{R}$, $m = 0, 1, \dots$ by

$$L_f^0 \sigma(q, x) = \sigma(q, x) \quad \text{and} \quad L_f^m \sigma(q, x) = \left(\frac{\partial}{\partial x} L_f^{m-1} \sigma(q, x) \right) f(q, x), \quad \text{for } m > 0.$$

We define the *pointwise relative degree* of σ with respect to f , as the function $n_{(\sigma, f)} : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbb{N}$ given by

$$n_{(\sigma, f)}(q, x) := \min \{m \in \mathbb{N} : L_f^m \sigma(q, x) \neq 0\}$$

Note that $n_{(\sigma, f)}(q, x) = 0$ for all (q, x) such that $\sigma(q, x) \neq 0$.

Definition 5 (Transverse Invariants) A hybrid automaton is said to have transverse invariants if f is analytic in its second argument and there exists a function $\sigma : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbb{R}$, also analytic in its second argument, such that

- $I(q) = \{x \in \mathbf{X} : \sigma(q, x) \geq 0\}$ for all $q \in \mathbf{Q}$; and
- for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$ there exists a finite $m \in \mathbb{N}$ such that $L_f^m \sigma(q, x) \neq 0$, i.e. $n_{(\sigma, f)}(q, x) < \infty$ for all (q, x) .

The transverse invariant condition implies that the invariant sets are closed subsets of \mathbf{X} .

Example : (Transverse invariants of the bouncing ball)

Recall that for the bouncing ball automaton introduced in Lecture 1, $I(\text{FLY}) = \{(x_1, x_2) \in \mathbb{R} : x_1 \geq 0\}$. Define $\sigma : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbb{R}$: $\sigma(\text{FLY}, (x_1, x_2)) = x_1$. Then

1. f is analytic in x (as it is a polynomial function)
2. σ is analytic in x (as it is also a polynomial function)
3. $I(\text{FLY}) = \{(x_1, x_2) \in \mathbb{R} : \sigma(\text{FLY}, (x_1, x_2)) \geq 0\}$
4. We check for finiteness of the relative degree of all points:

If $x_1 \neq 0$, $L_f^0 \sigma(\text{FLY}, (x_1, x_2)) \neq 0$, therefore $n_{(\sigma, f)}(q, x) = 0$

If $x_1 = 0 \wedge x_2 \neq 0$, $L_f^1 \sigma(\text{FLY}, (0, x_2)) = \frac{\partial \sigma}{\partial x} f = x_2 \neq 0$, therefore $n_{(\sigma, f)}(q, x) = 1$

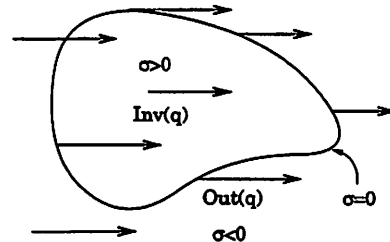
If $x_1 = 0 \wedge x_2 = 0$, $L_f^2 \sigma(\text{FLY}, (0, 0)) = -g \neq 0$, therefore $n_{(\sigma, f)}(q, x) = 2$

Overall, the automaton has transverse invariants.

For an automaton with transverse invariants and for all $q \in \mathbf{Q}$ we also define the set

$$\text{Out}(q) := \{x \in \mathbf{X} : L_f^{n_{(\sigma, f)}(q, x)} \sigma(q, x) < 0\}.$$

Note that $I(q)^c \subseteq \text{Out}(q)$.



Lemma 1 *A hybrid automaton with transverse invariants is non-blocking if for all $q \in \mathbf{Q}$ and for all $(q, x) \in \text{Reach}(H)$ with $x \in \text{Out}(q)$, there exists $(q, q') \in E$ such that*

- $x \in G(q, q')$; and
- $R(q, q', x) \neq \emptyset$.

Proof: Consider an arbitrary initial state $(q_0, x_0) \in \text{Init}$ and assume, for the sake of contradiction, that there does not exist an infinite execution starting at (q_0, x_0) . Let $\chi = (\tau, q, x)$ denote a maximal execution starting at (q_0, x_0) , and note that τ is a finite sequence.

First consider the case $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^{N-1}[\tau_N, \tau'_N]$. Let $(q_N, x_N) = \lim_{t \rightarrow \tau'_N} (q(t), x(t))$. Note that, by the definition of execution and a standard existence argument for continuous dynamical systems, the limit exists and χ can be extended to $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \{[\tau_i, \tau'_i]\}_{i=0}^N$, $\hat{q}(\tau'_N) = q_N$, and $\hat{x}(\tau'_N) = x_N$. This contradicts the maximality of χ .

Now consider the case $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^N$, and let $(q_N, x_N) = (q(\tau'_N), x(\tau'_N))$. Clearly, $(q_N, x_N) \in \text{Reach}(H)$. If $x_N \notin \text{Out}(q_N)^c = \{x \in \mathbf{X} : L_f^{n(\sigma, f)(q, x)} \sigma(q, x) > 0\}$, then, by the assumption that f and σ are analytic in their second argument, there exists $\epsilon > 0$ such that χ can be extended to $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$, $\tau_{N+1} = \tau_N$, and $\tau'_{N+1} = \tau_N + \epsilon$, by continuous evolution.

If, on the other hand $x_N \in \text{Out}(q_N)$, then there exists $(q', x') \in \mathbf{Q} \times \mathbf{X}$ such that $(q_N, q') \in E$, $x_N \in G(q_N, q')$ and $x' \in R(q_N, q', x_N)$. Therefore, χ can be extended to $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \{[\tau_i, \tau'_i]\}_{i=0}^{N+1}$, $\tau_{N+1} = \tau'_{N+1} = \tau_N$, $q(\tau_{N+1}) = q'$, $x(\tau_{N+1}) = x'$ by a discrete transition. In both cases the maximality of χ is contradicted. ■

Loosely speaking, the conditions of Lemma 1 indicate that a hybrid automaton with transverse invariants is non-blocking if transitions with non-trivial reset relations are enabled along the boundary of the invariant sets, at points where the continuous flow forces the state to exit the invariants. The conditions of Lemma 1 are tight, in the sense that blocking automata that violate the conditions exist, but are not necessary, in the sense that not all automata that violate the conditions are blocking.

Lemma 2 *A deterministic hybrid automaton with transverse invariants is non-blocking only if the conditions of Lemma 1 are satisfied.*

Proof: Consider a deterministic hybrid automaton H that violates the conditions of Lemma 1, that is there exists $(q', x') \in \text{Reach}(H)$ such that $x \in \text{Out}(q)$, but there is no $\hat{q}' \in \mathbf{Q}$ with $(q', \hat{q}') \in E$, $x' \in G(q', \hat{q}')$ and $R((q', \hat{q}'), x') \neq \emptyset$. Since $(q', x') \in \text{Reach}(H)$, there exists $(q_0, x_0) \in \text{Init}$ and a finite execution, $\chi = (\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$ such that $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^N$ and $(q', x') = (q(\tau'_N), x(\tau'_N))$.

We first show that χ is maximal. Assume first that there exists $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \{[\tau_i, \tau'_i]\}_{i=0}^{N-1}[\tau_N, \tau_N + \epsilon]$ for some $\epsilon > 0$. This requires that the execution can be extended beyond (q', x') by continuous evolution, which violates the assumption that $x \in \text{Out}(q)$. Next assume that there exists $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$ with $\tau_{N+1} = \tau'_N$. This requires that the execution can be extended beyond (q', x') by a discrete transition, that is there exists $(\hat{q}', \hat{x}') \in \mathbf{Q}$ such that $(q', \hat{q}') \in E$, $x' \in G(q', \hat{q}')$ and $\hat{x}' \in R((q', \hat{q}'), x')$. This also contradicts our original assumptions. Overall, $\chi \in \mathcal{H}_{(q_0, x_0)}^M$.

Now assume, for the sake of contradiction that H is non-blocking. Then, there exists $\chi' \in \mathcal{H}_{(q_0, x_0)}^\infty$. By definition, $\chi' \in \mathcal{H}_{(q_0, x_0)}^M$. But $\chi \neq \chi'$ (as the former is finite and the latter infinite), therefore $\mathcal{H}_{(q_0, x_0)}^M \supset \{\chi, \chi'\}$. This contradicts the assumption that H is deterministic. ■

ee291E Lecture 5: Existence of Executions: Examples

Shawn Schaffert and John Lygeros

February 3, 1999

1 Example 1: Lie Derivatives and Relative Degree

- Consider the closed unit disc, S , in \mathbb{R}^2 :

$$S = \{x \in \mathbb{R}^2 : x_1^2 + x_2^2 \leq 1\}$$

- Consider a vector field, f , in \mathbb{R}^2 :

$$f(x) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

- We want to classify the set of points in \mathbb{R}^2 such that the state can evolve for “a while” along f while staying in S ; for short, this will be referred to as whether the state *can* or *can not evolve while staying in* S .
- More formally, consider the execution $x(t)$ of the continuous dynamical system:

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \tag{1}$$

We want to determine the set of all x_0 for which there exists $\epsilon > 0$ such that $x(t) \in S$ for all $t \in [0, \epsilon]$.

- We begin analyzing this by considering three separate sub-sets of the state space, namely:
 - $x_0(t) \notin S$
 - $x_0(t) \in S^\circ$
 - x on the boundary of S
- **Notation:** S° denotes the interior of the set S . This is the largest open set contained in S .
- Clearly, when x_0 lies outside of S , we can not evolve while staying in S , since for all $\epsilon > 0$ there exists $t \in [0, \epsilon]$ (in particular $t = 0$) such that $x(t) \notin S$.
- Clearly, when x_0 lies in the interior of S (not on the boundary), the state can evolve while staying in S . If $x_0 \in S^\circ$, then there exists $\epsilon > 0$ such that $x(t) \in S$ for all $t \in [0, \epsilon]$, since S° is an open set and $x(t)$ is a continuous function of time.
- When x_0 lies on the boundary of S , it is unclear if the state can evolve and remain in S . Our intuition leads us to the following conclusions:
 - if f points into S at x_0 , the state should be able to evolve while staying in S .
 - if f points out of S at x_0 , the state should not be able to evolve while staying in S .
 - if f is tangential to S at x_0 , it is unclear if the state can or can not evolve while staying in S .

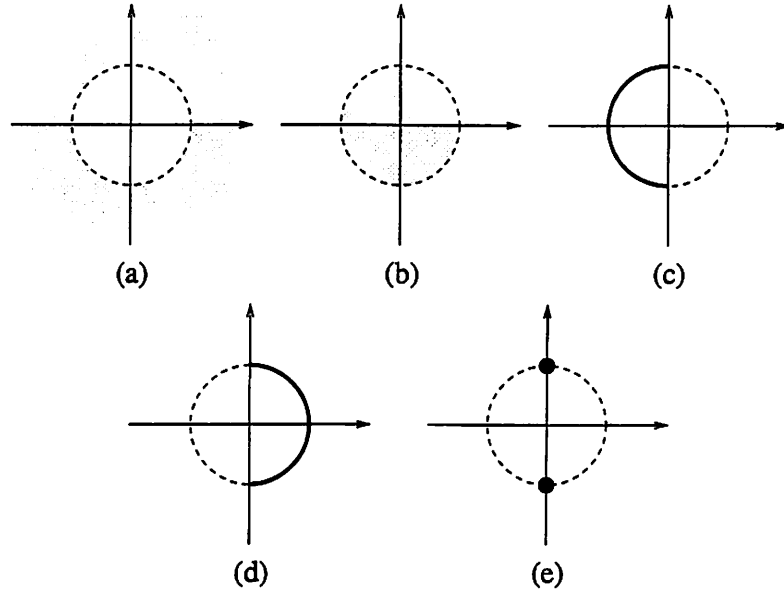


Figure 1: The unit disc S

- We would like to make this intuitive notion precise mathematically. For this reason we introduce the concepts of the Lie derivative and the relative degree.
- Embed the boundary of S in the level set of a function, $\sigma : \mathbb{R}^2 \rightarrow \mathbb{R}$

$$\sigma(x) = 1 - (x_1^2 + x_2^2)$$

- Using $\sigma(x)$, we note that:

$$\sigma(x) < 0 \iff x \notin S$$

$$\sigma(x) > 0 \iff x \in S$$

$$\sigma(x) = 0 \iff x \text{ on the boundary of } S$$

- We can reformulate our conditions on state evolution as follows:

$$x \notin S \iff \{x \in \mathbb{R}^2 : \sigma(x) < 0\} \rightarrow \text{the state can not evolve while staying in } S \text{ (Figure 1(a))}$$

$$x \in S^0 \iff \{x \in \mathbb{R}^2 : \sigma(x) > 0\} \rightarrow \text{the state can evolve while staying in } S \text{ (Figure 1(b))}$$

$$x \text{ on the boundary of } S \iff \{x \in \mathbb{R}^2 : \sigma(x) = 0\} \rightarrow ?$$

- Let us investigate the situation where x is on the boundary of S .
- Pick x_0 such that $\sigma(x_0) = 0$.
- Consider the execution $x(t)$ of (1), and in particular value of $\sigma(x(t))$. Note that $\sigma(x(0)) = \sigma(x_0) = 0$.
- Consider the derivative of $\sigma(x(t))$ with respect to time:

$$\begin{aligned} \frac{d}{dt}\sigma(x(t)) &= \frac{\partial \sigma(x(t))}{\partial x} \dot{x}(t) \\ &= [-2x_1 \quad -2x_2] \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= -2x_1 \end{aligned}$$

- Define the *Lie derivative of σ along f* by $\mathcal{L}_f \sigma(x) = -2x_1$.

- If $\sigma(x_0) = 0$ but $\mathcal{L}_f\sigma(x_0) > 0$, i.e., if $x_1^2 + x_2^2 = 1$ and $x_1 < 0$, then σ is increasing as time is progressing (i.e., we are moving inside S), therefore we can evolve while staying in S from x_0 (Figure 1(c)).
- If $\sigma(x_0) = 0$ but $\mathcal{L}_f\sigma(x_0) < 0$, i.e., if $x_1^2 + x_2^2 = 1$ and $x_1 > 0$, then σ is decreasing as time is progressing (i.e. we are moving outside S), therefore we can not evolve while staying in S from x_0 (Figure 1(d)).
- What if $\sigma(x_0) = 0$ and $\mathcal{L}_f\sigma(x_0) = 0$, i.e. if $x_1^2 + x_2^2 = 1$ and $x_1 = 0$?
- Consider the second Lie derivative, i.e.,

$$\frac{d}{dt}\mathcal{L}_f\sigma(x(t)) = \mathcal{L}_f^2\sigma(x(t)) = -2$$

- In this case, we can not evolve while staying in S since $\sigma(x_0) = 0$, $\mathcal{L}_f\sigma(x_0) = 0$, and $\mathcal{L}_f^2\sigma(x_0) < 0$; hence, the state is bound to move outside S (Figure 1(e)).
- Overall, the state can not evolve while staying in S from the following points:

$$\begin{aligned} \text{Out} &= \{x_0 \in \mathbb{R}^2 : \sigma(x_0) < 0\} \cup \\ &\quad \{x_0 \in \mathbb{R}^2 : \sigma(x_0) = 0 \wedge \mathcal{L}_f\sigma(x_0) < 0\} \cup \\ &\quad \{x_0 \in \mathbb{R}^2 : \sigma(x_0) = 0 \wedge \mathcal{L}_f\sigma(x_0) = 0 \wedge \mathcal{L}_f^2\sigma(x_0) < 0\} \cup \\ &\quad \dots \end{aligned}$$

- Define the *relative degree* as the function $n : \mathbb{R}^2 \rightarrow \mathbb{N}$ by:

$$n(x_0) = \begin{cases} 0 & \text{if } \sigma(x_0) \neq 0 \\ 1 & \text{if } \sigma(x_0) = 0 \wedge \mathcal{L}_f\sigma(x_0) \neq 0 \\ 2 & \text{if } \sigma(x_0) = 0 \wedge \mathcal{L}_f\sigma(x_0) = 0 \wedge \mathcal{L}_f^2\sigma(x_0) \neq 0 \\ \dots & \end{cases}$$

- Then Out can be written more compactly as:

$$\text{Out} = \{x_0 \in \mathbb{R}^2 : \mathcal{L}_f^{n(x_0)}\sigma(x_0) < 0\}$$

2 Analytic Functions

Definition 1 (Analytic Function) *A function is analytic if it is infinitely differentiable and the function's Taylor Series converges.*

- Considering example 1, we can see that both $\sigma(x)$ and $f(x)$ are analytic.
- Analytic vector fields are important because they produce executions that are analytic as a function of time.
- Referring to example 1, $\sigma(x(t))$ is an analytic function of t , since it is the composition of the analytic function σ with the analytic execution $x(t)$ of the analytic vector field f . The Taylor series expansion around $t = 0$:

$$\sigma(x(t)) = \sigma(x_0) + \frac{d}{dt}\sigma(x_0)t + \frac{d^2}{dt^2}\sigma(x_0)\frac{t^2}{2!} + \dots = \sigma(x_0) + \mathcal{L}_f\sigma(x_0)t + \mathcal{L}_f^2\sigma(x_0)\frac{t^2}{2!} + \dots$$

suggests why the first non-zero Lie derivative of σ dictates whether we the state can evolve while staying in S or not.

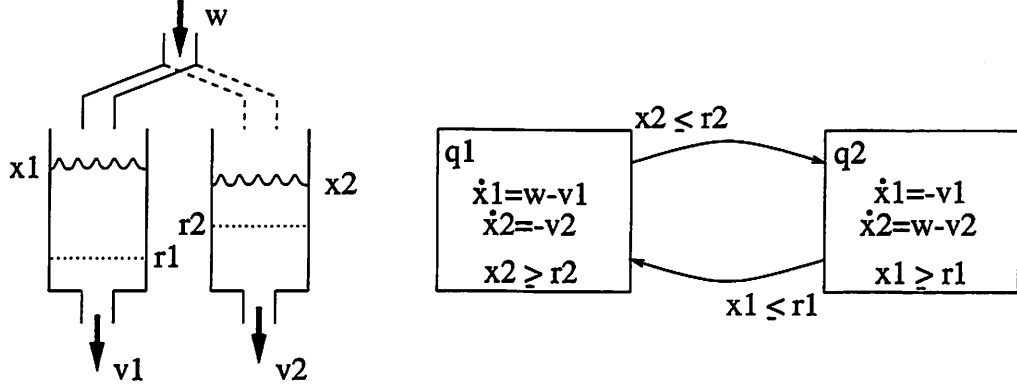


Figure 2: The water tank system

3 Example 2: The Water Tank System

Consider the water tank system of [1], shown in Figure 2. For $i = 1, 2$, let x_i denote the volume of water in Tank i , and $v_i > 0$ denote the (constant) flow of water out of Tank i . Let w denote the constant flow of water into the system, dedicated exclusively to either Tank 1 or Tank 2 at each point in time. The control task is to keep the water volumes above r_1 and r_2 , respectively (assuming that $x_1(0) > r_1$ and $x_2(0) > r_2$). This is to be achieved by a switched control strategy that switches the inflow to Tank 1 whenever $x_1 \leq r_1$ and to Tank 2 whenever $x_2 \leq r_2$. More formally:

Definition 2 (Water Tank Automaton) *The water tank automaton is a hybrid automaton with*

- $Q = \{q_1, q_2\}$ and $X = \mathbb{R}^2$;
- $\text{Init} = Q \times \{x \in X : (x_1 > r_1) \wedge (x_2 > r_2)\}$, $r_1, r_2 > 0$;
- $f(q_1, x) = (w - v_1, -v_2)^T$ and $f(q_2, x) = (-v_1, w - v_2)^T$, $v_1, v_2, w > 0$;
- $I(q_1) = \{x \in X : x_2 \geq r_2\}$ and $I(q_2) = \{x \in X : x_1 \geq r_1\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in X : x_2 \leq r_2\}$ and $G(q_2, q_1) = \{x \in X : x_1 \leq r_1\}$; and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$.

Recall that:

Definition 3 (Transverse Invariants) *A hybrid system H has transverse invariants if:*

1. f is analytic in x ,
2. $\exists \sigma : Q \times X \rightarrow \mathbb{R}$, analytic in x , such that $\text{Inv}(q) = \{x \in X : \sigma(q, x) \geq 0\}$, and
3. $n(q, x) < \infty \forall (q, x) \in Q \times X$.

Proposition 1 *The water tank automaton has transverse invariants.*

Proof:

1. f is constant for each q ; hence f is analytic in x

2. Consider $\sigma(q_1, x) = x_2 - r_2$ and $\sigma(q_2, x) = x_1 - r_1$. Then σ is polynomial (hence analytic) in x and $\sigma(q, x) \geq 0 \iff x \in \text{Inv}(q)$
3. if $\sigma(q, x) \neq 0$ then:

$$\sigma(q, x) \neq 0 \iff \begin{cases} q = q_1 \wedge x_2 \neq r_2 \\ q = q_2 \wedge x_1 \neq r_1 \end{cases} \quad \text{therefore } n(q, x) = 0$$

if $\sigma(q, x) = 0$ then:

$$\begin{aligned} \sigma(q, x) &= 0 \iff (q = q_1 \wedge x_2 = r_2) \vee (q = q_2 \wedge x_1 = r_1) \\ L_f \sigma(q, x) &= \begin{cases} -v_2 & \text{if } q = q_1 \\ -v_1 & \text{if } q = q_2 \end{cases} \quad \text{therefore } n(q, x) = 1 \end{aligned}$$

In either case, $n(q, x) \leq 1 < \infty$.

■

Recall that:

Lemma 1 (Non-Blocking Automata) *A hybrid automaton with transverse invariants is non-blocking if for all $(q, x) \in \text{Reach}(H)$ with $x \in \text{Out}(q) \exists (q, q') \in E$:*

1. $x \in G(q, q')$
2. $R((q, q'), x) \neq \emptyset$

Proposition 2 *The water tank automaton is non-blocking.*

Proof:

- By Proposition 1, the water tank automaton has transverse invariants, therefore Lemma 1 can be applied.
- Out for this example can be written as follows:

$$\text{Out}(q) = \begin{cases} \{x_2 < r_2\} \cup \{x_2 = r_2 \wedge -v_2 < 0\} = \{x_2 \leq r_2\} & \text{if } q = q_1 \\ \{x_1 \leq r_1\} & \text{if } q = q_2 \end{cases}$$

- Condition 1 of Lemma 1 is satisfied:

$$\text{Out}(q) = \begin{cases} \{x_2 \leq r_2\} = G(q_1, q_2) & \text{if } q = q_1 \\ \{x_1 \leq r_1\} = G(q_2, q_1) & \text{if } q = q_2 \end{cases}$$

- Condition 2 of Lemma 1 is satisfied:

$$R((q_1, q_2), x) = R((q_2, q_1), x) = \{x\} \neq \emptyset$$

■

References

- [1] R Alur and T A Henzinger, “Modularity for timed and hybrid systems”, in *CONCUR 97: Concurrency Theory*, A. Mazurkiewicz and J. Winkowski, Eds., Lecture Notes in Computer Science 1243, pp. 74–88. Springer-Verlag, 1997.

ee291E Lecture 6: Uniqueness of Executions

Adam Sweet & John Lygeros

February 8, 1999

1 Necessary Conditions for Existence

Lemma 1 of Lecture 4 is a tight sufficient condition: automata exist that violate the conditions of the lemma and may be either blocking or non-blocking. However, in the case that the automaton is deterministic, the lemma is sufficient. This was given as Lemma 2 of Lecture 4, but is repeated below:

Lemma 1 *A deterministic hybrid automaton with transverse invariants is non-blocking only if the conditions of Lemma 1 of Lecture 4 are satisfied.*

Proof: Consider a deterministic hybrid automaton H that violates the conditions of Lemma 1, Lecture 4: that is, there exists $(q', x') \in \text{Reach}(H)$ such that $x \in \text{Out}(q)$, but there is no $\hat{q}' \in \mathbf{Q}$ with $(q', \hat{q}') \in E$, $x' \in G(q', \hat{q}')$ and $R((q', \hat{q}'), x') \neq \emptyset$. Since $(q', x') \in \text{Reach}(H)$, there exists $(q_0, x_0) \in \text{Init}$ and a finite execution, $\chi = (\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$ such that $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^N$ and $(q', x') = (q(\tau'_N), x(\tau'_N))$.

We first show that χ is maximal, by showing that χ cannot be extended without violating our original assumptions. Assume first that there exists $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \{[\tau_i, \tau'_i]\}_{i=0}^{N-1} [\tau_N, \tau_N + \epsilon]$ for some $\epsilon > 0$. This requires that the execution can be extended beyond (q', x') by continuous evolution, which violates the assumption that $x \in \text{Out}(q)$. Next assume that there exists $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ with $\hat{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$ with $\tau_{N+1} = \tau'_N$. This requires that the execution can be extended beyond (q', x') by a discrete transition, that is there exists $(\hat{q}', \hat{x}') \in \mathbf{Q}$ such that $(q', \hat{q}') \in E$, $x' \in G(q', \hat{q}')$ and $\hat{x}' \in R((q', \hat{q}'), x')$. This also contradicts our original assumptions. Because χ cannot be extended by either continuous evolution or a discrete transition, $\chi \in \mathcal{H}_{(q_0, x_0)}^M$.

Now assume, for the sake of contradiction, that H is non-blocking. Then, there exists $\chi' \in \mathcal{H}_{(q_0, x_0)}^\infty$. By definition, $\chi' \in \mathcal{H}_{(q_0, x_0)}^M$. But $\chi \neq \chi'$ (as the former is finite and the latter infinite), therefore $\mathcal{H}_{(q_0, x_0)}^M \supset \{\chi, \chi'\}$. Because there are two maximal executions from the same initial state, this contradicts the assumption that H is deterministic. ■

2 Uniqueness Conditions

Lemma 2 *A hybrid automaton with transverse invariants is deterministic if and only if for all $q, q', q'' \in \mathbf{Q}$ and all $(q, x) \in \text{Reach}(H)$:*

- If $x \in \bigcup_{(q, q') \in E} G(q, q')$ then $x \in \text{Out}(q)$;
- if $(q, q') \in E$ and $(q, q'') \in E$ with $q' \neq q''$ then $x \notin G(q, q') \cap G(q, q'')$; and
- if $(q, q') \in E$ and $x \in G(q, q')$ then $|R(q, q', x)| \leq 1$.

Proof: For the “if” part, assume, for the sake of contradiction, that there exists an initial state $(q_0, x_0) \in \text{Init}$ and two maximal executions $\chi = (\tau, q, x)$ and $\hat{\chi} = (\hat{\tau}, \hat{q}, \hat{x})$ starting at (q_0, x_0) with $\chi \neq \hat{\chi}$. Let $\psi = (\rho, p, y) \in \mathcal{H}_{(q_0, x_0)}$ denote the maximal common prefix of χ and $\hat{\chi}$. Such a prefix exists as the executions start at the same initial state. Moreover, ψ is not infinite, as $\chi \neq \hat{\chi}$. Therefore, as in the proof of Lemma 1 in Lecture 4, ρ can be assumed to be of the form $\rho = \{[\rho_i, \rho'_i]\}_{i=0}^N$, as otherwise the maximality of ψ would be contradicted by an existence and uniqueness argument of the continuous solution along f . Let $(q_N, x_N) = (q(\rho'_N), x(\rho'_N)) = (\hat{q}(\rho'_N), \hat{x}(\rho'_N))$. Clearly, $(q_N, x_N) \in \text{Reach}(H)$. We distinguish the following cases:

Case 1: $\rho'_N \notin \{\tau'_i\}$ and $\rho'_N \notin \{\hat{\tau}'_i\}$, i.e., ρ'_N is not a time when a discrete transition takes place in either χ or $\hat{\chi}$. Then, by the definition of execution and a standard existence and uniqueness argument for continuous dynamical systems, there exists $\epsilon > 0$ such that the prefixes of χ and $\hat{\chi}$ are defined over $\hat{\rho} = \{[\rho_i, \rho'_i]\}_{i=0}^{N-1}[\rho_N, \rho'_N + \epsilon)$ and are identical. This contradicts the maximality of ψ .

Case 2: $\rho'_N \in \{\tau'_i\}$ and $\rho'_N \notin \{\hat{\tau}'_i\}$, i.e., ρ'_N is a time when a discrete transition takes place in χ but not in $\hat{\chi}$. The fact that a discrete transition takes place from (q_N, x_N) in χ indicates that there exists $q' \in Q$ such that $(q_N, q') \in E$ and $x_N \in G(q_N, q')$. The fact that no discrete transition takes place from (q_N, x_N) in $\hat{\chi}$ indicates that there exists $\epsilon > 0$ such that $\hat{\chi}$ is defined over $\hat{\rho} = \{[\rho_i, \rho'_i]\}_{i=0}^{N-1}[\rho_N, \rho'_N + \epsilon)$. A necessary condition for this is that $x_N \notin \text{Out}(q)$. This contradicts Condition 1 of the lemma.

Case 3: $\rho'_N \notin \{\tau'_i\}$ and $\rho'_N \in \{\hat{\tau}'_i\}$, symmetric to Case 2.

Case 4: $\rho'_N \in \{\tau'_i\}$ and $\rho'_N \in \{\hat{\tau}'_i\}$, i.e., ρ'_N is a time when a discrete transition takes place in both χ and $\hat{\chi}$. The fact that a discrete transition takes place from (q_N, x_N) in both χ and $\hat{\chi}$ indicates that there exist (q', x') and (\hat{q}', \hat{x}') such that $(q_N, q') \in E$, $(q_N, \hat{q}') \in E$, $x_N \in G(q_N, q')$, $x_N \in G(q_N, \hat{q}')$, $x' \in R(q_N, q', x_N)$, and $\hat{x}' \in R(q_N, \hat{q}', x_N)$. Note that by Condition 2 of the lemma, $q' = \hat{q}'$, hence, by Condition 3, $x' = \hat{x}'$. Therefore, the prefixes of χ and $\hat{\chi}$ are defined over $\hat{\rho} = \{[\rho_i, \rho'_i]\}_{i=0}^N[\rho_{N+1}, \rho'_{N+1}]$, with $\rho_{N+1} = \rho'_{N+1} = \rho'_N$, and are identical. This contradicts the maximality of ψ .

This concludes the proof of the “if” part. For the “only if” part, assume that there exists $(q', x') \in \text{Reach}(H)$ such that at least one of the conditions of the lemma is violated. Since $(q', x') \in \text{Reach}(H)$, there exists $(q_0, x_0) \in \text{Init}$ and a finite execution, $\chi = (\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$ such that $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^N$ and $(q', x') = (q(\tau'_N), x(\tau'_N))$. If condition 1 is violated, then there exists $\hat{\chi}$ and $\bar{\chi}$ with $\hat{\tau} = \{[\hat{\tau}_i, \hat{\tau}'_i]\}_{i=0}^{N-1}[\tau_N, \tau_N + \epsilon)$, $\epsilon > 0$ and $\bar{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$, $\tau_{N+1} = \tau'_N$, such that $\chi < \hat{\chi}$ and $\chi < \bar{\chi}$. If condition 2 is violated, there exist $\hat{\chi}$ and $\bar{\chi}$ with $\hat{\tau} = \bar{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$, $\tau_{N+1} = \tau'_N$ and $\hat{q}(\tau_{N+1}) \neq \bar{q}(\tau_{N+1})$, such that $\chi < \hat{\chi}$, $\chi < \bar{\chi}$. Finally, if condition 3 is violated, then there exist $\hat{\chi}$ and $\bar{\chi}$ with $\hat{\tau} = \bar{\tau} = \tau[\tau_{N+1}, \tau'_{N+1}]$, $\tau_{N+1} = \tau'_N$ and $\hat{x}(\tau_{N+1}) \neq \bar{x}(\tau_{N+1})$, such that $\chi < \hat{\chi}$, $\chi < \bar{\chi}$. In all three cases, let $\bar{\chi} \in \mathcal{H}_{(q_0, x_0)}^M$ and $\bar{\bar{\chi}} \in \mathcal{H}_{(q_0, x_0)}^M$ denote maximal executions of which $\hat{\chi}$ and $\bar{\chi}$ are prefixes of respectively. Since $\hat{\chi} \neq \bar{\chi}$, $\bar{\chi} \neq \bar{\bar{\chi}}$, therefore $|\mathcal{H}_{(q_0, x_0)}^M| \geq 2$, therefore H is non-deterministic. ■

Loosely speaking, the conditions of Lemma 2 can be summarized as follows:

- Discrete transitions must be forced by the continuous flow exiting the invariant set;
- No two discrete transitions can be enabled simultaneously; and
- No point can be mapped onto two different points by the reset map.

Note that these conditions are both necessary and sufficient.

3 Summary

Combining Lemmas 1 of Lecture 4 and Lemma 2 of Lecture 6 give the following theorem:

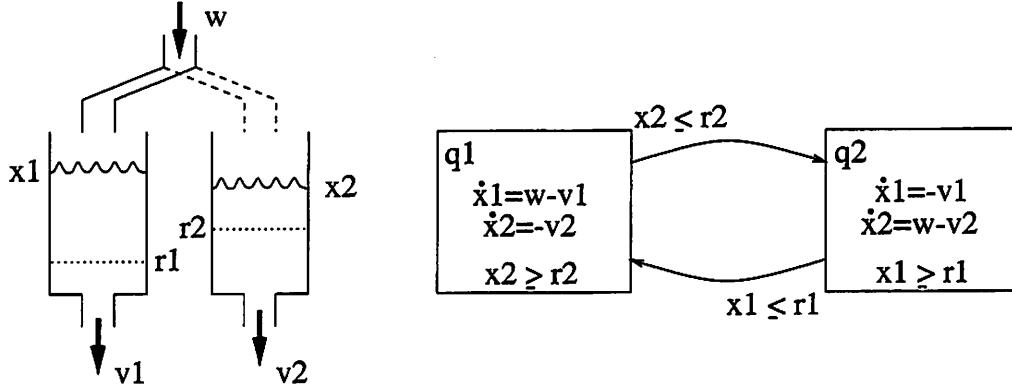


Figure 1: The water tank system

Theorem 1 (Existence and Uniqueness of Executions) *A hybrid automaton with transverse invariants accepts a unique infinite execution for all $(q_0, x_0) \in \text{Init}$ if it satisfies the conditions of Lemma 1 of Lecture 4 and Lemma 2 of Lecture 6.*

Proof: If the hybrid automaton satisfies the conditions of Lemma 1 of Lecture 4, then $|\mathcal{H}_{(q_0, x_0)}^\infty| \geq 1$, for all $(q_0, x_0) \in \text{Init}$. If it satisfies the conditions of Lemma 2 of Lecture 6, then $|\mathcal{H}_{(q_0, x_0)}^M| \leq 1$, for all $(q_0, x_0) \in \text{Init}$. But every infinite execution is also maximal, therefore $\mathcal{H}_{(q_0, x_0)}^\infty \subseteq \mathcal{H}_{(q_0, x_0)}^M$. Therefore, $1 \leq |\mathcal{H}_{(q_0, x_0)}^\infty| \leq |\mathcal{H}_{(q_0, x_0)}^M| \leq 1$, or in other words $|\mathcal{H}_{(q_0, x_0)}^\infty| = |\mathcal{H}_{(q_0, x_0)}^M| = 1$. ■

Again, the conditions of the theorem are tight. Similar conditions exist for the case where the invariant sets are open; these are given in Problem Set 1.

Notice that the conditions of the lemmas mention the set of reachable states, $\text{Reach}(H)$. However, $\text{Reach}(H)$ can be difficult to calculate explicitly for an actual system. For the purposes of the lemmas and Theorem 1 it suffices to show that the conditions of the lemmas hold in a set of states that contains $\text{Reach}(H)$ (for example $\mathbf{Q} \times \mathbf{X}$). If the conditions of the lemmas hold for that set, then they also must hold for $\text{Reach}(H)$ and the lemmas can then be applied to the system.

Definition 1 (Invariant Set) *A set of states $S \subseteq \mathbf{Q} \times \mathbf{X}$ is called invariant if $\text{Reach}(H) \subseteq S$.*

Proposition 1 *The class of invariant sets is closed under union and intersection.*

Trivially $\mathbf{Q} \times \mathbf{X}$ is an invariant set. More interesting sets are typically shown to be invariant by an induction argument on the length of the system executions.

4 Example: The Water Tank System

Consider the water tank system of [1], shown in Figure 1. For $i = 1, 2$, let x_i denote the volume of water in Tank i , and $v_i > 0$ denote the (constant) flow of water out of Tank i . Let w denote the constant flow of water into the system, dedicated exclusively to either Tank 1 or Tank 2 at each point in time. The control task is to keep the water volumes above r_1 and r_2 , respectively (assuming that $x_1(0) > r_1$ and $x_2(0) > r_2$). This is to be achieved by a switched control strategy that switches the inflow to Tank 1 whenever $x_1 \leq r_1$ and to Tank 2 whenever $x_2 \leq r_2$. More formally:

Definition 2 (Water Tank Automaton) *The water tank automaton is a hybrid automaton with*

- $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}^2$;

- $\text{Init} = \mathbf{Q} \times \{x \in \mathbf{X} : (x_1 > r_1) \wedge (x_2 > r_2)\}, r_1, r_2 > 0;$
- $f(q_1, x) = (w - v_1, -v_2)^T$ and $f(q_2, x) = (-v_1, w - v_2)^T, v_1, v_2, w > 0;$
- $I(q_1) = \{x \in \mathbf{X} : x_2 \geq r_2\}$ and $I(q_2) = \{x \in \mathbf{X} : x_1 \geq r_1\};$
- $E = \{(q_1, q_2), (q_2, q_1)\};$
- $G(q_1, q_2) = \{x \in \mathbf{X} : x_2 \leq r_2\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : x_1 \leq r_1\};$ and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = x.$

Proposition 2 (Existence and Uniqueness of Executions) *The water tank automaton accepts a unique infinite execution for each initial state.*

Proof: Let $\sigma(q_1, x) = x_2 - r_2$ and $\sigma(q_2, x) = x_1 - r_1$. Then $L_f^1 \sigma(q_1, x) = -v_2 < 0$ and $L_f^1 \sigma(q_2, x) = -v_1 < 0$. Since both f and σ are analytic functions of x and $I(q_i) = \{x \in \mathbf{X} : \sigma(q_i, x) \geq 0\}$, the water tank automaton has transverse invariants.

Note that $n(q_1, x) = 0$ if $x_2 \neq r_2$, and $n(q_1, x) = 1$ if $x_2 = r_2$, therefore $\text{Out}(q_1) = \{x \in \mathbf{X} : x_2 \leq r_2\} = G(q_1, q_2)$ (and similarly for q_2). This implies that Condition 1 of Lemma 1, Lecture 4 and Condition 1 of Lemma 2, Lecture 6 are satisfied. Moreover, $|R(q_1, q_2, x)| = |R(q_2, q_1, x)| = 1$, therefore Condition 2 of Lemma 1, Lecture 4 and Condition 3 of Lemma 2, Lecture 6 are satisfied. Condition 2 of Lemma 2, Lecture 6 is also trivially satisfied. The claim follows by Theorem 1. ■

References

- [1] R Alur and T A Henzinger, “Modularity for timed and hybrid systems”, in *CONCUR 97: Concurrency Theory*, A. Mazurkiewicz and J. Winkowski, Eds., Lecture Notes in Computer Science 1243, pp. 74–88. Springer-Verlag, 1997.

ee291E Lecture 7: Zeno Hybrid Automata

Richard Liu & John Lygeros

February 10, 1999

1 Zeno Executions

- What does “Zeno” mean? The name Zeno refers to the philosopher Zeno of Elea (500–400 B.C.), whose major work consisted of a number of famous paradoxes. They were designed to explain the view of his mentor, Parmenides, that the ideas of motion and evolving time lead to contradictions. An example is Zeno’s Second Paradox of Motion, in which Achilles is racing against a turtle.
- An execution is called Zeno, if it contains an infinite number of transitions in a finite amount of time. An automaton is called Zeno if it accepts a Zeno execution.

Definition 1 (Zeno Hybrid Automaton) *An execution $\chi = (\tau, q, x)$ of a hybrid automaton H is called Zeno if it is infinite, but $\tau_\infty = \sum_i (\tau'_i - \tau_i) < \infty$. A hybrid automaton H is called Zeno, if there exists $(q_0, x_0) \in \text{Init}$ such that $\mathcal{H}_{(q_0, x_0)}$ contains a Zeno execution.*

- τ_∞ is referred to as the *Zeno time*.
- The definition is *existential* (it is enough for one execution to be Zeno for the automaton to be called Zeno).

Example: Consider again the water tank system (Figure 1), and recall that it is deterministic and non-blocking. Therefore it accepts a unique infinite execution for each initial condition (q_0, x_0) . We show that if

$$\max\{v_1, v_2\} < w < v_1 + v_2$$

then this execution is Zeno. At $x_1 = r_1, x_2 = r_2$, trivially it is Zeno. Even if this is not the case, the execution is still Zeno. Assume initially that $x_1 > r_1$ and $x_2 > r_2$, all executions reach $x_1 = r_1$ and $x_2 > r_2$. Without loss of generality set $r_1 = r_2 = 0$. Note that all executions reach a state where $q = q_1, x_1 = 0$ in finite time (after at most two transitions). Therefore, consider an execution with:

$$\tau_0 = 0, \quad q(\tau_0) = q_1, \quad x_1(\tau_0) = 0, \quad x_2(\tau_0) = h \geq 0$$

The first transition takes place at:

$$\tau'_0 = \tau_1 = \frac{h}{v_2}, \quad q(\tau_1) = q_2, \quad x_1(\tau_1) = (w - v_1)\frac{h}{v_2}, \quad x_2(\tau_1) = 0$$

The second transition takes place at:

$$\tau'_1 = \tau_2 = \tau_1 + \frac{(w - v_1)h}{v_1 v_2}, \quad q(\tau_2) = q_1, \quad x_1(\tau_2) = 0, \quad x_2(\tau_2) = \frac{(w - v_1)(w - v_2)h}{v_1 v_2}$$

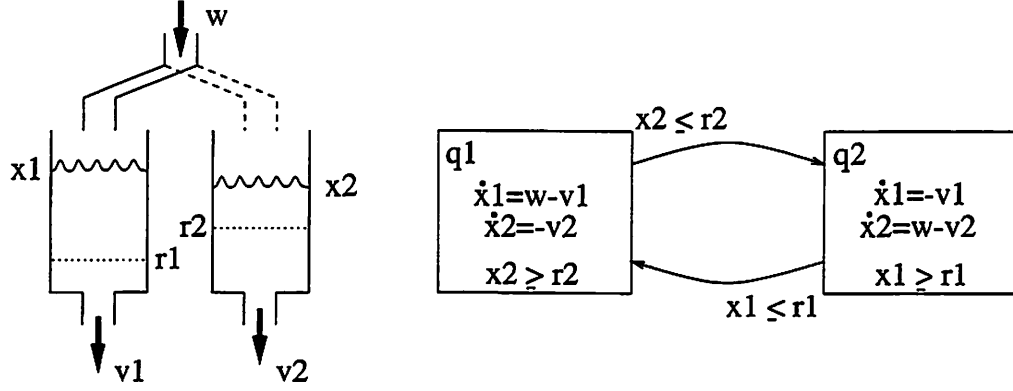


Figure 1: The water tank system

The third transition takes place at:

$$\tau'_2 = \tau_3 = \tau_2 + \frac{(w - v_1)(w - v_2)h}{v_1 v_2^2}, \quad q(\tau_3) = q_2, \quad x_1(\tau_3) = \frac{(w - v_1)^2(w - v_2)h}{v_1 v_2^2}, \quad x_2(\tau_3) = 0$$

and so on. Overall,

$$\begin{aligned} \sum_{i=0}^{\infty} (\tau'_i - \tau_i) &= \frac{h}{v_2} + \frac{(w - v_1)h}{v_1 v_2} + \frac{(w - v_1)(w - v_2)h}{v_1 v_2^2} + \dots \\ &= \frac{h}{v_2} \left[\sum_{i=1}^{\infty} \left(\frac{w - v_1}{v_1} \right)^i \left(\frac{w - v_2}{v_2} \right)^{i-1} + \sum_{i=0}^{\infty} \left(\frac{w - v_1}{v_1} \right)^i \left(\frac{w - v_2}{v_2} \right)^i \right] \\ &= \frac{h}{v_2} \left[\frac{w - v_1}{v_1} + 1 \right] \left[\sum_{i=0}^{\infty} \left(\frac{(w - v_1)(w - v_2)}{v_1 v_2} \right)^i \right] \end{aligned}$$

Since $w < v_1 + v_2$ the infinite sum converges. Therefore:

$$\sum_{i=0}^{\infty} (\tau'_i - \tau_i) = \frac{hw}{v_1 v_2} \left(\frac{1}{1 - \frac{(w - v_1)(w - v_2)}{v_1 v_2}} \right) = \frac{h}{v_1 + v_2 - w}$$

Note that this is the time one would expect the tanks to drain.

2 Types of Zeno Phenomena

- Zeno phenomena do not arise in physical systems.
- Zeno phenomena arise as a consequence of modeling over-abstraction.
- They are artifacts of our desire to make the modeling formalism powerful enough to capture a wide class of hybrid phenomena.
- They are a nuisance in more ways than one, as they lead to:
 - **Semantical problems:** How is an execution to be defined beyond the Zeno time?
 - **Analysis problems:** Induction and reachability proofs become suspect. For example, for the water tank system one can show that $x_1 \geq r_1 \wedge x_2 \geq r_2$ along all executions. Clearly, however, if $w < v_1 + v_2$ the tanks will drain in finite time.

– **Controller Synthesis problems:**

1. The controller can cheat by forcing time to converge. For example, in the case of the water tank system, the controller appears to succeed in maintaining the water levels above r_1 and r_2 .
2. Some classes of controls, such as relaxed controls and sliding mode controls, are naturally Zeno.

– **Simulation problems:** Simulation stalls at the Zeno time (refer to lecture by Karl Johansson).

• **Classification of the Zeno phenomena:**

1. **Mathematical curiosities.** Example: non-analytic invariants. Consider the hybrid automaton:

- $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}$;
- $\text{Init} = \mathbf{Q} \times \mathbf{X}$;
- $f(q, x) = 1$ for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$;
- $I(q_1) = \{x \in \mathbf{X} : e^{-1/|x|} \sin(1/x) \leq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : e^{-1/|x|} \sin(1/x) \geq 0\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbf{X} : e^{-1/|x|} \sin(1/x) \geq 0\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : e^{-1/|x|} \sin(1/x) \leq 0\}$; and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = x$.

The execution of H with initial state $(q_1, -1)$ exhibits an infinite number of discrete transitions by $\tau_\infty = 1$. The reason is that the (non-analytic) function $e^{-1/|x|} \sin(1/x)$ has an infinite number of zeros in the finite interval $(-1, 0)$.

2. **Discontinuous vector fields.** Example: sliding surfaces. Consider the hybrid automaton:

- $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}$;
- $\text{Init} = \mathbf{Q} \times \mathbf{X}$;
- $f(q_1, x) = -1, f(q_2, x) = 1$;
- $I(q_1) = \{x \in \mathbf{X} : x \geq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : x \leq 0\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbf{X} : x \leq 0\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : x \geq 0\}$; and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = x$.

All executions reach $x = 0$ in finite time and take an infinite number of transitions from then on, without any time progress.

3. **Non-zero time progress, continuous state.** Example: the water tank system.

4. **Non-zero time progress, discontinuous state.** Example: the bouncing ball.

5. **Non-zero time progress, discontinuous state, no limit.** Example: Bouncing ball with a switch. Consider the hybrid automaton:

- $\mathbf{Q} = \{q\}$ and $\mathbf{X} = \mathbb{R}^3$;
- $\text{Init} = \{q\} \times \{x \in \mathbf{X} : x_1 \geq 0 \wedge x_3 = 1\}$;
- $f(q, x) = (x_2, -g, 0)^T$ with $g > 0$;
- $I(q) = \{x \in \mathbf{X} : x_1 \geq 0\}$;
- $E = \{(q, q)\}$;
- $G(q, q) = \{x \in \mathbf{X} : [x_1 < 0] \vee [(x_1 = 0) \wedge (x_2 \leq 0)]\}$; and
- $R(q, q, x) = (x_1, -x_2/c, -x_3)^T$ with $c > 1$.

Like the bouncing ball system, this system takes an infinite number of jumps until a finite time τ_∞ . However, unlike the bouncing ball, as $t \rightarrow \tau_\infty$ the state does not converge (in particular x_3 keeps jumping from 1 to -1 faster and faster).

3 Conditions for Existence of Zeno Executions

The only known conditions to characterize the Zeno phenomenon are fairly trivial

Theorem 1 *If Q is finite and (Q, E) is a directed acyclic graph then H is not Zeno.*

Proof: If there is no loop in the graph (Q, E) there can only be a finite number of transitions in each execution. ■

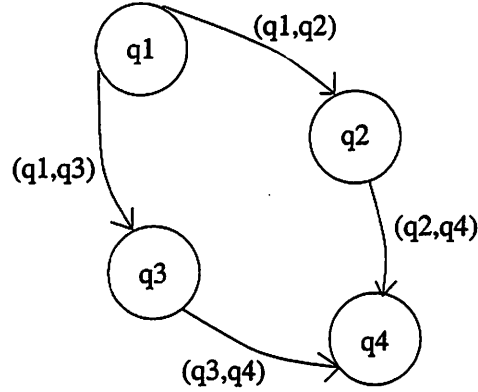


Figure 2: A directed acyclic graph

Theorem 2 *Assume there exists a finite collection of states $\{(q_i, x_i)\}_{i=1, \dots, N}$ such that:*

1. $(q_1, x_1) = (q_N, x_N)$
2. *there exists $i = 1, \dots, N$, such that $(q_i, x_i) \in \text{Reach}(H)$*
3. *for $i = 1, \dots, N - 1$, $(q_i, q_{i+1}) \in E$, $x_i \in G(q_i, q_{i+1})$ and $x_{i+1} \in R(q_i, q_{i+1}, x_i)$.*

Then H is Zeno.

Proof: Consider the execution, χ that reaches the reachable (q_i, x_i) , and then takes an infinite number of transitions around the loop without allowing time to evolve (Figure 3). ■

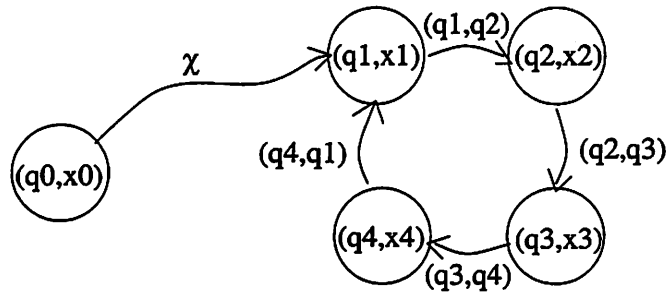


Figure 3: Cycle in transition relations

- Some better conditions may be obtained for special cases (e.g. discontinuous vector fields).

4 Resolving the Zeno Phenomenon

- In some cases it may be possible to resolve/avoid the Zeno phenomenon by appropriately redefining the automata or the executions.
 - Most mathematical curiosities can be eliminated by analyticity assumptions.
 - For discontinuous vector fields introduce new states and consider Filippov solutions.
 - Filippov solutions motivated by relaxation. Conditions exist under which relaxation leads to unique Filippov solutions.
 - This is not always the case with hybrid automata. In some cases regularization seems to lead to well defined extensions for the Zeno executions (for example in the case of the bouncing ball, Figures 4 and 5) while in others the extension seems to depend on the details of the regularization (for example, in the case of the water tank system, Figures 6 and 7).

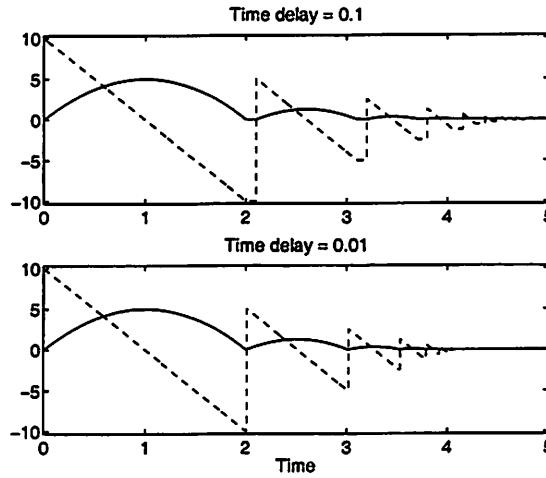


Figure 4: Simulation of bouncing ball with temporal regularization. The upper plot corresponds to a time delay during the bounce of $\epsilon = 0.1$ and the lower to $\epsilon = 0.01$.

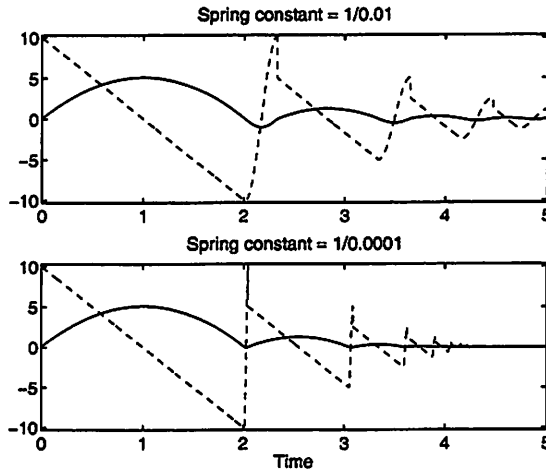


Figure 5: Simulation of bouncing ball with dynamical regularization. The upper plot corresponds to spring constant $1/\epsilon = 1/0.01$ and the lower to $1/\epsilon = 0.0001$.

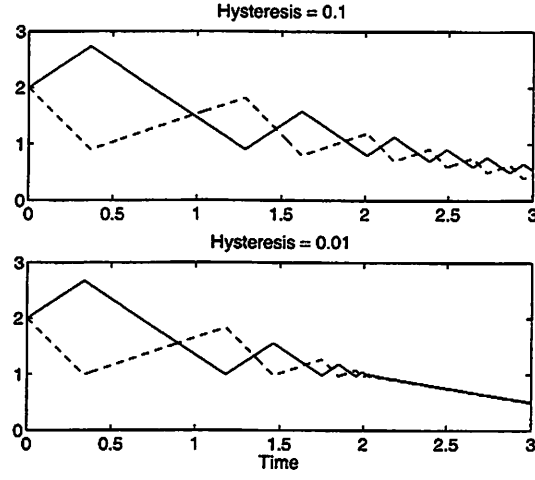


Figure 6: Simulation of spatial regularized water tanks. The upper plot corresponds to hysteresis $\epsilon = 0.1$ and the lower to $\epsilon = 0.01$. The solid line is x_1 and the dashed x_2 .

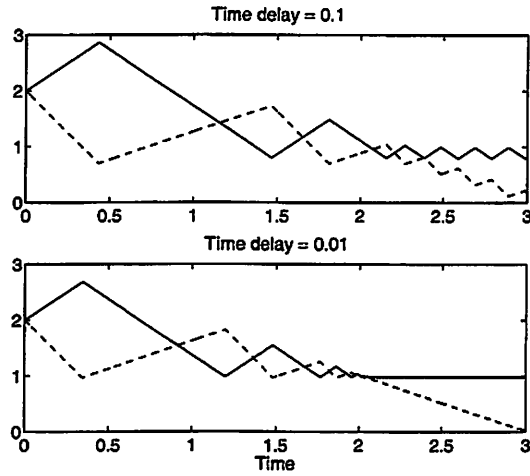


Figure 7: Simulation of temporally regularized water tank automaton. The upper plot corresponds to time delay during switching from one tank to the other of $\epsilon = 0.1$ and the lower to $\epsilon = 0.01$. The solid line is x_1 and the dashed x_2 .

ee291E Lecture 8: Open Hybrid Automata & Composition

Shahid Rashid & John Lygeros

February 17, 1999

1 Automata with Inputs and Outputs

1.1 Recap

- Up to now: *autonomous hybrid systems*, also known as *closed hybrid systems*, i.e. hybrid automata with no “inputs” or “outputs”.
- Good for:
 - Modeling and simulation of small to medium physical systems.
 - Analysis, verifying that all executions satisfy certain desirable properties.
- Limited because:
 - No sense of “control”.
 - System must be modeled as a single, monolithic block, which may be cumbersome, wasteful, or impossible. We would like to be able to build up the system description out of the “composition” of smaller pieces. For example, in automated highway model each vehicle individually.

1.2 Open Hybrid Automata

- Introduce *input* and *output* variables, and define appropriate *composition* operation.
- Big topic, we will only skim the surface here.
- Discussion based on [1, 2], notation changed over to the one used so far in the class.
- **Notation:** consider a collection of variables X , and a subset $X' \subseteq X$. For any valuation $x \in \mathbf{X}$ we define the *restriction* of x to X' , $x|_{X'}$, to be a valuation of the variables in X' that agrees with x .
- **Example:** consider $X = \{x_1, x_2, x_3\}$ with $\mathbf{X} = \mathbb{R}^3$. Then $(x_1 = 1, x_2 = 2, x_3 = 3)|_{\{x_1, x_2\}} = (x_1 = 1, x_2 = 2)$.
- Easily extends to sets of valuations.

Definition 1 (Open Hybrid Automaton) An open hybrid automaton H is a collection $H = (Q, X, V, Y, \text{Init}, f, h, \text{Inv}, E, G, R)$, where

- Q is a finite collection of discrete state variables;
- X is a finite collection of continuous state variables;

- V is a finite collection of input variables. We assume $V = V_D \cup V_C$, where V_D contains discrete and V_C contains continuous variables.
- Y is a finite collection of output variables. We assume $Y = Y_D \cup Y_C$, where Y_D contains discrete and Y_C contains continuous variables.
- $\text{Init} \subseteq \mathbf{Q} \times \mathbf{X}$ is a set of initial states;
- $f : \mathbf{Q} \times \mathbf{X} \times \mathbf{V} \rightarrow \mathbb{R}^n$ is a vector field;
- $h : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbf{Y}$ is an output map (note: \mathbf{V} is not in the domain of h);
- $\text{Inv} : \mathbf{Q} \rightarrow 2^{\mathbf{X} \times \mathbf{V}}$ assigns to each $q \in \mathbf{Q}$ an invariant set;
- $E \subset \mathbf{Q} \times \mathbf{Q}$ is a collection of discrete transitions;
- $G : E \rightarrow 2^{\mathbf{X} \times \mathbf{V}}$ assigns to each $e = (q, q') \in E$ a guard; and
- $R : E \times \mathbf{X} \times \mathbf{V} \rightarrow 2^{\mathbf{X}}$ assigns to each $e = (q, q') \in E$, $x \in \mathbf{X}$ and $v \in \mathbf{V}$ a reset relation.

Remarks:

1. The term Open Hybrid Automata is not uniformly used for non-autonomous hybrid automata throughout the literature.
2. We refer to $(q, x) \in \mathbf{Q} \times \mathbf{X}$ as the *state* of H , to $v \in \mathbf{V}$ as the input of H and to $y \in \mathbf{Y}$ as the output of H .
3. To avoid technicalities with continuous dynamics we impose the following assumption:

Assumption 1 Assume $f(q, x, v)$ and $h(q, x)$ are globally Lipschitz continuous in x and $f(q, x, v)$ is continuous in v (this is a restriction of V_C , V_D trivially satisfied).

Definition 2 (Execution) An execution χ of an open hybrid automaton $H \in \mathcal{H}$ is a collection $\chi = (\tau, q, x, v, y)$ with $\tau \in T = \{[\tau_i, \tau'_i]\}_{i=1}^N$, $q : \tau \rightarrow \mathbf{Q}$, $x : \tau \rightarrow \mathbf{X}$, $v : \tau \rightarrow \mathbf{V}$ and $y : \tau \rightarrow \mathbf{Y}$ satisfying:

- **Initial condition:** $(q(\tau_0), x(\tau_0)) \in \text{Init}$;
- **Continuous evolution:** for all i with $\tau_i < \tau'_i$, q , x , v and y are continuous over $[\tau_i, \tau'_i]$ and for all $t \in [\tau_i, \tau'_i]$, $(x(t), v(t)) \in \text{Inv}(q(t))$ and $\frac{d}{dt}x(t) = f(q(t), x(t), v(t))$;
- **Discrete Evolution:** for all i , either $(q(\tau'_i), x(\tau'_i)) = (q(\tau_{i+1}), x(\tau_{i+1}))$, or $e_i = (q(\tau'_i), q(\tau_{i+1})) \in E$, $(x(\tau'_i), v(\tau'_i)) \in G(e_i)$, and $x(\tau_{i+1}) \in R(e_i, x(\tau'_i), v(\tau'_i))$; and,
- **Output Evolution:** for all $t \in \tau$, $y(t) = h(q(t), x(t))$.

Remarks:

1. For an execution $\chi = (\tau, q, x, v, y)$ we use $(q_0, x_0) = (q(\tau_0), x(\tau_0))$ to denote the initial state of χ .
2. Two types of transition:
 - Internal transitions, where the state changes.
 - Environmental transitions, taking place “somewhere else” and affecting only the input variables of H (if anything).
3. Control can enter in a number of places. We can use v to:

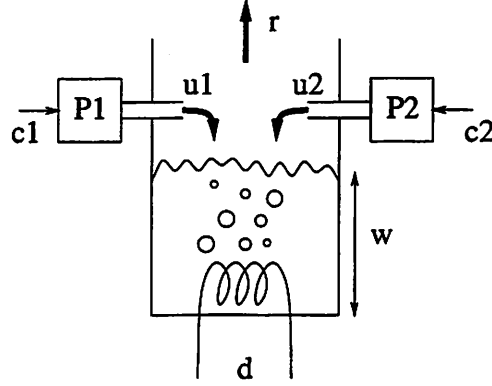


Figure 1: The Steam Boiler

- Guide the continuous evolution through f .
- Enable internal transitions through G .
- Force internal transitions through Inv .
- Determine the state after a transition through R .

4. Valuation of the input variables is unconstrained:

- Initially, and
- After every discrete transition.

A continuity constraint is imposed during continuous evolution.

5. Possible extensions:

- Input-Output dependencies: $h : Q \times X \times V \rightarrow Y$. Useful for modeling sensors, actuators, etc. See [2].
- Set valued output maps: $h : Q \times X \rightarrow 2^Y$. Useful for modeling nondeterminism.
- State dependent input constraints: $\phi : Q \times X \rightarrow 2^V$. Useful for controller synthesis, enforcing state constraints.
- Actions: associated with discrete transitions, can be thought of as discrete variables changing value. Useful for modeling discrete commands, etc. See [1].
- Synchronous vs. I/O interaction. See [3] or notes of lecture by Tunc Simsek.

Example: (Steam Boiler System, Figure 1) Consider the steam boiler system of [4] (originally introduced in [5, 6]). The steam boiler consists of a tank containing water and a heating element that causes the water to boil and escape as steam. The water level in the boiler is denoted by w , and, for the sake of simplicity, consider only $w > 0$. The water is replenished by two pumps which at time t pump water into the boiler at rates $u_1(t)$ and $u_2(t)$ respectively. The water boils off at rate r with d a variable that controls the rate of evaporation: $\dot{r} = d$. At every time t , pump i can either be on ($u_i(t) = \bar{P}_i$) or off ($u_i(t) = 0$). There is a delay \bar{T}_i between the time pump i is ordered to switch on and the time q_i switches to P_i . There is no delay when the pumps are switched off. We will use three hybrid automata to describe this system, one for the boiler, $B = (Q_B, X_B, V_B, Y_B, \text{Init}_B, f_B, h_B, \text{Inv}_B, E_B, G_B, R_B)$, and one for each of the pumps, $P_i = (Q_i, X_i, V_i, Y_i, \text{Init}_i, f_i, h_i, \text{Inv}_i, E_i, G_i, R_i)$.

The boiler automaton is defined by:

- $Q_B = \{q_B\}$, $Q_B = \{\text{BOILING}\}$;
- $X_B = \{w, r\}$, $X_B = \mathbb{R}^2$;

- $V_B = \{u_1, u_2, d\}$, $\mathbf{V}_B = [0, \bar{P}_1] \times [0, \bar{P}_2] \times [-D_1, D_2]$, where $\bar{P}_1, \bar{P}_2, D_1, D_2 > 0$;
- $Y_B = \{y_1, y_2\}$, $\mathbf{Y}_B = \mathbb{R}^2$;
- $\text{Init} = \{\text{BOILING}\} \times [0, W] \times [0, R]$, where $W, R > 0$;

$$f(\text{BOILING}, w, r, u_1, u_2, d) = \begin{bmatrix} \dot{w} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} u_1 + u_2 - r \\ d \end{bmatrix}$$

$$h(\text{BOILING}, w, r) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} w \\ r \end{bmatrix}$$

- $\text{Inv}(\text{BOILING}) = \mathbf{X}_B \times \mathbf{V}_B$
- $E = \emptyset$

Notice that since $E = \emptyset$, G and R are trivial and need not be explicitly defined.

The automaton for pump i (Figure 2) is defined by:

- $Q_i = \{q_i\}$, $\mathbf{Q}_i = \{\text{OFF}, \text{GOING_ON}, \text{ON}\}$;
- $X_i = \{T_i\}$, $\mathbf{X}_i = \mathbb{R}$;
- $V_i = \{c_i\}$, $\mathbf{V}_i = \{0, 1\}$;
- $Y_i = \{u_i\}$, $\mathbf{Y}_i = [0, \bar{P}_i]$;
- $\text{Init} = \{\text{OFF}\} \times \{0\}$;

$$f(q_i, T_i, c_i) = \begin{cases} 1 & \text{if } q_i = \text{GOING_ON} \vee q_i = \text{ON} \\ 0 & \text{if } q_i = \text{OFF} \end{cases}$$

$$h(q_i, T_i) = \begin{cases} 0 & \text{if } q_i = \text{OFF} \vee q_i = \text{GOING_ON} \\ \bar{P}_i & \text{if } q_i = \text{ON} \end{cases}$$

$$\text{Inv}(q_i) = \begin{cases} \mathbf{X}_i \times \{c_i = 0\} & \text{if } q_i = \text{OFF} \\ \{T_i \leq \bar{T}_i\} \times \{c_i = 1\} & \text{if } q_i = \text{GOING_ON} \\ \mathbf{X}_i \times \{c_i = 1\} & \text{if } q_i = \text{ON} \end{cases}$$

- $E = \{(\text{OFF}, \text{GOING_ON}), (\text{GOING_ON}, \text{OFF}), (\text{GOING_ON}, \text{ON}), (\text{ON}, \text{OFF})\}$

$$G(e) = \begin{cases} \mathbf{X}_i \times \{c_i = 1\} & \text{if } e = (\text{OFF}, \text{GOING_ON}) \\ \mathbf{X}_i \times \{c_i = 0\} & \text{if } e = (\text{GOING_ON}, \text{OFF}) \\ \{T_i \geq \bar{T}_i\} \times \{c_i = 1\} & \text{if } e = (\text{GOING_ON}, \text{ON}) \\ \mathbf{X}_i \times \{c_i = 0\} & \text{if } e = (\text{ON}, \text{OFF}) \end{cases}$$

$$R(e, T_i, c_i) = \begin{cases} \{0\} & \text{if } e = (\text{OFF}, \text{GOING_ON}) \\ \{0\} & \text{if } e = (\text{GOING_ON}, \text{OFF}) \\ \{T_i\} & \text{if } e = (\text{GOING_ON}, \text{ON}) \\ \{0\} & \text{if } e = (\text{ON}, \text{OFF}) \end{cases}$$

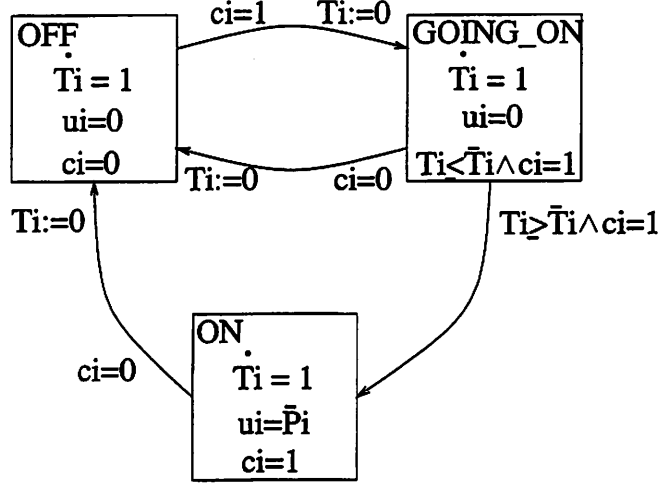


Figure 2: The pump hybrid automaton

1.3 Traces

- Consider an execution $\chi = (\tau, q, x, v, y) \in \mathcal{H}$ of an open hybrid automaton H . To the “outside world” the only visible part of this execution is (τ, v, y) .
- A transition is called *inert* if $(v(\tau'_i), y(\tau'_i)) = (v(\tau_{i+1}), y(\tau_{i+1}))$.
- Inert transitions can be eliminated, by concatenating the trajectories either side of the transition.
- More formally, the *concatenation* of two functions:

$$\begin{aligned} (v_1, y_1) &: [\tau_i, \tau'_i] \rightarrow \mathbf{V} \times \mathbf{Y} \\ (v_2, y_2) &: [\tau_{i+1}, \tau'_{i+1}] \rightarrow \mathbf{V} \times \mathbf{Y} \end{aligned}$$

with $\tau'_i = \tau_{i+1}$ and $(v(\tau'_i), y(\tau'_i)) = (v(\tau_{i+1}), y(\tau_{i+1}))$, can be defined as the function

$$(v, y) : [\tau_i, \tau'_{i+1}] \rightarrow \mathbf{V} \times \mathbf{Y}$$

such that $(v(t), y(t)) = (v_1(t), y_1(t))$ for all $t \in [\tau_i, \tau'_i]$ and $(v(t), y(t)) = (v_2(t), y_2(t))$ for all $t \in [\tau_{i+1}, \tau'_{i+1}]$.

- A *trace* of H is a finite or infinite execution of H , restricted to the input and output variables, with all inert transitions removed and the corresponding trajectories concatenated.
- We use $\text{Trace}(H)$ to denote the set of all traces by H .
- Two open hybrid automata H_1 and H_2 are *comparable* if $V_1 = V_2$ and $Y_1 = Y_2$.
- We say an open hybrid automaton H_1 implements an open hybrid automaton H_2 if they are comparable and $\text{Trace}(H_1) \subseteq \text{Trace}(H_2)$.
- Think of H_2 as a specification and of H_1 as a proposed “implementation” of that specification (in software or in hardware).

2 Composition

Definition 3 (Compatible Hybrid Automata) *Two open hybrid automata H_1 and H_2 are called compatible if:*

$$\begin{aligned}(Q_1 \cup X_1) \cap (Q_2 \cup X_2 \cup V_2 \cup Y_2) &= \emptyset \\ (Q_2 \cup X_2) \cap (Q_1 \cup X_1 \cup V_1 \cup Y_1) &= \emptyset \\ Y_1 \cap Y_2 &= \emptyset\end{aligned}$$

For two compatible hybrid automata, H_1 and H_2 define:

$$\begin{aligned}V_1 &= V_{11} \cup V_{12} \text{ with } V_{11} = V_1 \setminus Y_2 \text{ and } V_{12} = V_1 \cap Y_2 \\ V_2 &= V_{21} \cup V_{22} \text{ with } V_{22} = V_2 \setminus Y_1 \text{ and } V_{21} = V_2 \cap Y_1\end{aligned}$$

For this partition define the maps:

$$\begin{aligned}h_{12} : Q_1 \times X_1 &\longrightarrow V_{21} \\ (q_1, x_1) &\longmapsto h_1(q_1, x_1)|_{V_{21}} \\ &\text{and} \\ h_{21} : Q_2 \times X_2 &\longrightarrow V_{12} \\ (q_2, x_2) &\longmapsto h_2(q_2, x_2)|_{V_{12}}\end{aligned}$$

Definition 4 (Composition) *The composition of two compatible hybrid automata H_1 and H_2 is a hybrid automaton $H = H_1 \| H_2 = (Q, X, V, Y, \text{Init}, f, h, \text{Inv}, E, G, R)$, with*

- $Q = Q_1 \cup Q_2$;
- $X = X_1 \cup X_2$;
- $V = (V_1 \cup V_2) \setminus (Y_1 \cup Y_2) = (V_1 \cup V_2) \cap (Y_1 \cup Y_2)^C$;
- $Y = Y_1 \cup Y_2$;
- $\text{Init} = \{(q, x) \in Q \times X : (q|_{Q_i}, x|_{X_i}) \in \text{Init}_i, \text{ for } i = 1, 2\}$;
- $f : Q \times X \times V \rightarrow \mathbb{R}^{n_1+n_2}$ given by:

$$f(q, x, v) = \begin{bmatrix} f_1 \left(q|_{Q_1}, x|_{X_1}, v|_{V_{11}}, h_{21} \left(q|_{Q_2}, x|_{X_2} \right) \right) \\ f_2 \left(q|_{Q_2}, x|_{X_2}, v|_{V_{22}}, h_{12} \left(q|_{Q_1}, x|_{X_1} \right) \right) \end{bmatrix}$$

- $h : Q \times X \rightarrow Y$ is a vector field given by:

$$h(q, x) = \left(h_1 \left(q|_{Q_1}, x|_{X_1} \right), h_2 \left(q|_{Q_2}, x|_{X_2} \right) \right)$$

- $\text{Inv} : Q \rightarrow 2^{X \times V}$ given by:

$$\text{Inv}(q) = \left\{ (x, v) \in X \times V : \left(x|_{X_i}, v|_{V_{ii}}, h_{ji} \left(q|_{Q_j}, x|_{X_j} \right) \right) \in \text{Inv}_i(q|_{Q_i}), \text{ for } i, j = 1, 2, i \neq j \right\}$$

- $E \subset Q \times Q$ given by:

$$\begin{aligned}E &= \{(q, q') \in Q \times Q : (q|_{Q_1}, q'|_{Q_1}) \in E_1 \wedge q|_{Q_2} = q'|_{Q_2}\} \\ &\cup \{(q, q') \in Q \times Q : q|_{Q_1} = q'|_{Q_1} \wedge (q|_{Q_2}, q'|_{Q_2}) \in E_2\} \\ &\cup \{(q, q') \in Q \times Q : (q|_{Q_1}, q'|_{Q_1}) \in E_1 \wedge (q|_{Q_2}, q'|_{Q_2}) \in E_2\}\end{aligned}$$

- $G : E \rightarrow 2^{X \times V}$ given by:

$$G(q, q') = \{(x, v) \in X \times V : e_i = (q|_{Q_i}, q'|_{Q_i}) \in E_i \Rightarrow (x|_{X_i}, v|_{V_i}, h_{ji}(q|_{Q_j}, x|_{X_j})) \in G_i(e_i) \text{ for } i, j = 1, 2, i \neq j\}$$

- $R : E \times X \times V \rightarrow 2^X$ defined by:

$$R(q, q', x, v) = \{x' \in X : e_i = (q|_{Q_i}, q'|_{Q_i}) \in E_i \Rightarrow \begin{aligned} & x'|_{X_i} \in R_i(e_i, x|_{X_i}, v|_{V_i}, h_{ji}(q|_{Q_j}, x|_{X_j})) \\ & \vee (q|_{Q_i}, x|_{X_i}) = (q'|_{Q_i}, x'|_{X_i}) \\ & (q|_{Q_i}, q'|_{Q_i}) \notin E_i \Rightarrow x'|_{X_i} = x|_{X_i} \end{aligned} \text{ for } i, j = 1, 2, i \neq j \}$$

Remarks:

1. Both interleaving and synchronous transitions are allowed.
2. A transition is forced for the composition if a transition is forced in at least one of the constituents.
3. A transition is enabled for the composition if a transition is enabled in at least one of the constituents.
4. For an execution $\chi = (\tau, q, x, v, y)$ of the composition $H = H_1 \| H_2$, let $\chi|_{H_i}$ denote the restriction of the execution to the variables of H_i , i.e. the collection $\chi_i = (\tau_i, q_i, x_i, v_i, y_i)$ where $\tau_i \in T$, $q_i : \tau_i \rightarrow Q_i$, $x_i : \tau_i \rightarrow X_i$, $v_i : \tau_i \rightarrow V_i$ and $y_i : \tau_i \rightarrow Y_i$ defined by $\tau_i = \tau$, and for all $t \in \tau$, $q_i(t) = q(t)|_{Q_i}$, $x_i(t) = x(t)|_{X_i}$, $v_i(t) = v(t)|_{V_i}$, $y_i(t) = y(t)|_{Y_i}$.
5. The domain of H was earlier restricted to $Q \times X$ to disallow self-loops in the composition of systems.

Proposition 1 Let $H = H_1 \| H_2$ be the composition of two compatible hybrid automata H_1 and H_2 . Then $\mathcal{H} = \{\chi : \chi|_{H_i} \in \mathcal{H}_i, \text{ for } i = 1, 2\}$.

Example: In the steam boiler example, since the only shared variables are u_1 and u_2 :

Proposition 2 B , P_1 , and P_2 are compatible.

The composition $H = B \| P_1 \| P_2$ can therefore be defined. It is an open hybrid automaton with:

- $Q = \{q_B, q_1, q_2\}$, $|Q| = 9$;
- $X = \{w, r, T_1, T_2\}$, $X = \mathbb{R}^4$;
- $V = \{c_1, c_2, d\}$, with $V = V_C \cup V_D$, $V_C = \{d\}$ and $V_D = \{c_1, c_2\}$;
- $Y_B = \{y_1, y_2, u_1, u_2\}$;
- ...

3 Renaming and Hiding

- Somewhat cumbersome to define each variable individually. Renaming allows us to define a single automaton, and then use it in different compositions by changing the name of its variables. This operation does not change the dynamics of the system, only the way it interacts with other systems.
- It may sometimes be desirable to eliminate output variables, especially after it has been “composed” with an input variable of another automaton. This operation does not change the dynamics of the automaton, it just affects its external behavior.
- See [2, 7] for a formal discussion.

4 Receptivity

- Zenoness more difficult to define for open hybrid automata.
- *Receptivity*: An open automaton is receptive if it accepts an admissible (time divergent) execution for all input trajectories with a finite number of transitions.
- See [8, 1, 2] for a formal discussion.

References

- [1] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H.B. Weinberg, “Hybrid I/O automata”, in *Hybrid Systems III*, number 1066 in LNCS, pp. 496–510. Springer Verlag, 1996.
- [2] R. Alur and T.A. Henzinger, “Modularity for timed and hybrid systems”, in *Proceedings of the Eighth International Conference on Concurrency Theory (CONCUR 1997)*. 1997, number 1243 in LNCS, pp. 74–88, Springer Verlag.
- [3] Akash Deshpande, Aleks Gollu, and Luigi Semenzato, “The SHIFT programming language and run-time system for dynamic networks of hybrid automata”, Tech. Rep. UCB-ITS-PRR-97-7, Institute of Transportation Studies, University of California, Berkeley, 1997.
- [4] Tomas A. Henzinger and Howard Wong-Toi, “Using HYTECH to synthesize control parameters for a steam boiler”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS, pp. 265–282. Springer Verlag, 1996.
- [5] J.-R. Abrial, E. Börger, and H. Langmaack, “The steam-boiler case study project, an introduction”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS. Springer Verlag, 1996.
- [6] J.-R. Abrial, “The steam-boiler control specification problem”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS. Springer Verlag, 1996.
- [7] R. Alur and T.A. Henzinger, “Reactive modules”, in *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. 1996, pp. 207–218, IEEE Computer Society Press.
- [8] Rainer Gawlick, Roberto Segala, Jorgen Sogaard-Andersen, and Nancy Lynch, “Liveness in timed and untimed systems”, in *Automata, Languages and Programming*, number 820 in LNCS, pp. 166–177. Springer Verlag, 1994.

ee291E Lecture 9: Analysis, Synthesis & Sequence Properties

John Musacchio and John Lygeros

February 22, 1999

1 Analysis, Synthesis and Validation

- Up to now:
 - Modeling of hybrid systems
 - Autonomous systems and systems with inputs and outputs
 - Properties of models: non-blocking, deterministic, Zeno
- Coming up:
 1. **Analysis:** (Verification) Prove that a hybrid automaton satisfies certain properties
 2. **Synthesis:** (Controller design) Choose control inputs so that closed loop hybrid automaton satisfy certain properties
 3. **Validation:** Apply these methods to real systems and test them in experiment or simulation.
- Formalize these notions, starting with “property” and “satisfy”

2 Notation

- Consider a collection of variables W
- Let $\text{Hyb}(W)$ denote the set of *hybrid sequences* of W :

$$\text{Hyb}(W) = \{(\tau, w) : \tau \in \mathcal{T}, w : \tau \rightarrow \mathbf{W}\}$$

- Recall that:

Definition 1 (Hybrid Time Trajectory) A hybrid time trajectory τ is a finite or infinite sequence of intervals of the real line, $\tau = \{I_i\}$, $i \in \mathbb{N}$, satisfying the following conditions:

- I_i is closed, unless τ is a finite sequence and I_i is the last interval in which case it is left closed but can be right open.
- Let $I_i = [\tau_i, \tau'_i]$. Then for all i , $\tau_i \leq \tau'_i$ and for $i > 0$, $\tau_i = \tau'_{i-1}$.

- Note that τ may also be an empty sequence.
- **Example:** Consider an open hybrid automaton, $H = (Q, X, V, Y, \text{Init}, f, h, \text{Inv}, E, G, R)$. Then:

$$\mathcal{H} \subseteq \text{Hyb}(Q \cup X \cup V \cup Y), \text{Trace}(H) \subseteq \text{Hyb}(V \cup Y)$$

- Given $H = (Q, X, V, Y, \text{Init}, f, h, \text{Inv}, E, G, R)$, we denote by $\text{Var}(H) = Q \cup X \cup V \cup Y$.
- To simplify the notation we use the term “Hybrid Automaton” to refer to both autonomous and open hybrid automata. Note that one can interpret an autonomous hybrid automaton as an open hybrid automaton with $V = Y = \emptyset$.
- For $W \subseteq \text{Var}(H)$ and for a $\chi = (\tau, q, x, v, y) \in \mathcal{H}$ we use $\chi|_W = (\tau, w) \in \text{Hyb}(W)$ to denote the hybrid sequence of W such that for all $t \in \tau$, $w(t) = (q(t), x(t), v(t), y(t))|_W$.
- We denote by

$$\mathcal{H}|_W = \{(\tau, w) \in \text{Hyb}(W) : \exists \chi \in \mathcal{H} \text{ with } (\tau, w) = \chi|_W\}$$

The set of executions of H restricted to the variables in W . Note that $\text{Trace}(H) = \mathcal{H}|_{(V \cup Y)}$.

3 Sequence Properties

Definition 2 (Sequence Property) A sequence property is a pair (W, P) of a collection of variables, W , and a map:

$$P : \text{Hyb}(W) \rightarrow \{\text{True}, \text{False}\}$$

- We say a sequence χ **satisfies** property (W, P) if:

$$\chi \in \text{Hyb}(W) \text{ and } P(\chi) = \text{True}$$

- We say a hybrid automaton, H , satisfies a property (W, P) (and write $H \models (W, P)$) if:
 1. $W \subseteq \text{Var}(H)$
 2. $P(\chi|_W) = \text{True}$, for all $\chi \in \mathcal{H}$.
- We will sometimes use Temporal Logic formulas to specify properties.
- **Example:** Consider a hybrid automaton $H = (Q, X, V, Y, \text{Init}, f, h, \text{Inv}, E, G, R)$, and a subset $F \subseteq Q \times X$. We define “always F ” as the property $(Q \cup X, \Box F)$, where:

$$\Box F(\chi) = \text{True} \text{ iff } \forall t \in \tau, (q(t), x(t)) \in F$$

We define “eventually F ” as the property $(Q \cup X, \Diamond F)$,

$$\Diamond F(\chi) = \text{True} \text{ iff } \exists t \in \tau \text{ such that } (q(t), x(t)) \in F$$

- H satisfies “always F ” if along all executions of H the state remains in F .
- H satisfies “eventually F ” if along all executions of H the state reaches F at some point.
- More complex combinations lead to different types of properties: $\Box \Diamond F$ (“Responsiveness”: always, eventually in F), $\Diamond \Box F$ (“Persistence”: eventually, always in F), etc.
- For a formal treatment see [1, 2]

4 Analysis and Synthesis for Sequence Properties

Problem 1 (Sequence Analysis) Given a hybrid automaton H and a sequence property (W, P) with $W \subseteq \text{Var}(H)$ show that $H \models (W, P)$.

Remarks:

- If this is not the case, find a witness $\chi \in \mathcal{H}$ such that $P(\chi|_W) = \text{False}$.
- Works for autonomous hybrid automata.
- Can have input variables acting as uncontrollable disturbances. The sequence analysis problem then requires showing for all possible disturbance sequences, the resulting executions satisfy the desirable properties.

Problem 2 (Sequence Synthesis) *Given a hybrid automaton H_P and a sequence property (W, P) with $W \subseteq \text{Var}(H)$ find a compatible hybrid automaton H_C such that $H_P || H_C \models (W, P)$.*

Remarks:

- H_P models the “plant”
- H_C models the “controller”
- Requires controllable input variables (else there is nothing to control and the synthesis problem reduces to an analysis problem).
- May have both controllable inputs (controls) and uncontrollable inputs (disturbances). The synthesis problem then becomes a game between the control and the disturbance; we are looking for a controller to determine the control sequence such that for all disturbance sequences the resulting executions satisfy the desired properties.

Example: Analysis of the bouncing ball. Consider the bouncing ball automaton (Figure 1):

- $Q = \{\text{FLY}\}$ and $X = \mathbb{R}^2$;
- $\text{Init} = \{\text{FLY}\} \times \{x \in X : x_1 \geq 0\}$;
- $f(\text{FLY}, x) = (x_2, -g)^T$ with $g > 0$;
- $I(\text{FLY}) = \{x \in X : x_1 \geq 0\}$;
- $E = \{(\text{FLY}, \text{FLY})\}$;
- $G(\text{FLY}, \text{FLY}) = \{x \in X : [x_1 < 0] \vee [(x_1 = 0) \wedge (x_2 \leq 0)]\}$; and
- $R(\text{FLY}, \text{FLY}, x) = (x_1, -x_2/c)^T$ with $c > 1$.

Proposition 1 *The bouncing ball automaton satisfies $(X, \Box\{x_1 \geq -1\})$.*

Proof: We show that in fact H satisfies $(X, \Box\{x_1 \geq 0\})$. (Note that $\Box\{x_1 \geq 0\}(\chi) = \text{True} \Rightarrow \Box\{x_1 \geq -1\}(\chi) = \text{True}$). Let:

$$\text{Inv} = \{\text{FLY}\} \times \{x \in X : x_1 \geq 0\}, \quad \text{Inv}_{\text{FLY}} = \{x \in X : x_1 \geq 0\}$$

We want to show that for all executions χ , $(q(t), x(t)) \in \text{Inv}$ for all $t \in \tau$.

- Claim is true at all initial states, $\text{Init} \subseteq \text{Inv}$.
- If a discrete transition takes place from a state in Inv , the state after the transition will also be in Inv , since there is only one transition, $e = (\text{FLY}, \text{FLY})$, with $G(e) \cap \text{Inv}_{\text{FLY}} = \{x \in X : (x_1 = 0) \wedge (x_2 \leq 0)\}$, and $R(e, x) = (x_1, -x_2/c)^T$, therefore:

$$x \in G(e) \cap \text{Inv}_{\text{FLY}} \Rightarrow R(e, x) \subseteq \text{Inv}_{\text{FLY}}$$

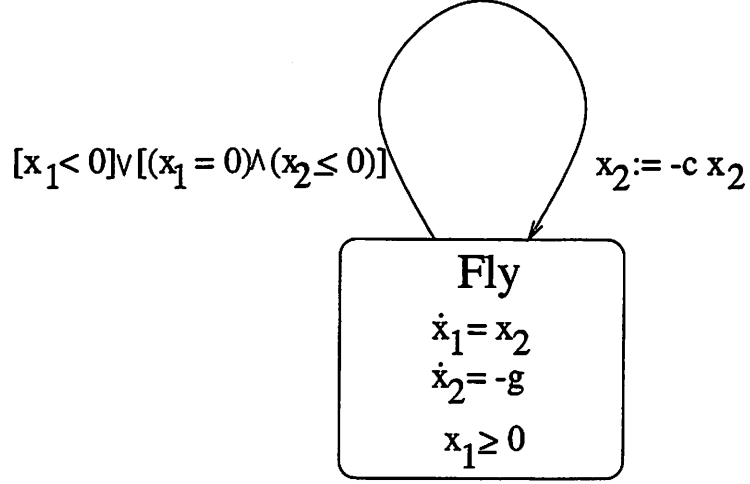


Figure 1: Bouncing ball

- Along continuous evolution one can have $x_1 < 0$ only if $x_1 = 0 \wedge x_2 \leq 0$ first. If this is the case system is forced to take a transition, and can therefore not exit Inv along continuous evolution. ■

Proposition 2 *The bouncing ball automaton satisfies $(X, \Diamond\{x_1 = 0\})$.*

Proof: After at most one discrete transition, continuous evolution starts. Along continuous evolution, $x_1(t) = x_1(0) + x_2(0)t - gt^2/2$, therefore, sooner or later $x_1(t) = 0$. ■

Remarks:

- The bouncing ball automaton also satisfies $(X, \Box\Diamond\{x_1 = 0\})$. The proof is the same as above. Intuitively, $(X, \Box\Diamond\{x_1 = 0\})$ requires $(X, \Diamond\{x_1 = 0\})$ to hold as if all reachable states were initial conditions.
- The bouncing ball automaton does not satisfy $(X, \Diamond\Box\{x_1 = 0\})$.

5 Safety and Liveness Properties

- For a formal treatment see [1, 2, 3].
- The above are instances of deductive proof techniques:
 - Invariant assertions $((q, x) \in \text{Inv})$
 - Progress functions $(x_1(t))$.
- These proof techniques are typically used to prove certain classes of properties:
 - Invariant assertions are used to prove *safety properties*
 - Progress functions are used to prove *liveness properties*

Definition 3 (Safety Property) *A sequence property (W, P) is called a safety property if it is:*

1. *Non-empty:* $\{\chi \in \text{Hyb}(W) : P(\chi) = \text{True}\} \neq \emptyset$

2. *Prefix closed*: if $P(\chi) = \text{True}$ then $P(\hat{\chi}) = \text{True}$ for all $\hat{\chi} \leq \chi$.
3. *Limit closed*: if $\chi_1 \leq \chi_2 \leq \dots$ is an infinite sequence with $P(\chi_i) = \text{True}$ and $\chi = \lim_{i \rightarrow \infty} \chi_i$ then $P(\chi) = \text{True}$.

Remarks

- The limit is taken in the successive extension ordering, and is unique.
- Loosely interpreted as meaning “something bad does not happen”.
- In this context, the conditions of the definition can be thought of as:
 1. Non-emptiness: Nothing bad can happen if nothing happens at all, therefore $P((\emptyset, w)) = \text{True}$.
 2. Prefix closure: If nothing bad happens in a sequence, nothing bad could have happened in all prefixes of that sequence.
 3. Limit closure: if something bad happens in a sequence, it has to happen after a finite “time”.
- Example: $(X, \Box\{x_1 \geq 0\})$ is a safety property for the bouncing ball.
- In fact:

Proposition 3 $(W, \Box F)$ for $F \subseteq \mathbf{W}$ with $F \neq \emptyset$ is a safety property.

- For discrete systems all safety properties are of the form $(W, \Box F)$. It is unclear whether this is the case for hybrid systems.

Definition 4 (Liveness Property) A sequence property (W, P) is called a liveness property if for all finite sequences $w \in \text{Hyb}(W)$ there exists $\hat{w} \in \text{Hyb}(W)$ such that:

1. $w \leq \hat{w}$
2. $P(\hat{w}) = \text{True}$

Remarks

- Loosely interpreted as meaning that “something good eventually happens”.
- In this context, the conditions of the definition can be thought of as requiring that whatever has happened up to now, it should still be possible for something good to happen eventually.
- Example: $(X, \Diamond\{x_1 = 0\})$ is a liveness property for the bouncing ball.
- In fact:

Proposition 4 $(W, \Diamond F)$ for $F \subseteq \mathbf{W}$ with $F \neq \emptyset$ is a liveness property.

- More classes of liveness properties are possible, $\Box\Diamond$, $\Diamond\Box$, etc.

Theorem 1 (W, P) is both a safety and a liveness property if and only if $P(\chi) = \text{True}$ for all $\chi \in \text{Hyb}(W)$.

Theorem 2 Let (W, P) be a sequence property such that $\{\chi \in \text{Hyb}(W) : P(\chi) = \text{True}\} \neq \emptyset$. Then there exist a safety property (W, P_1) and a liveness property (W, P_2) such that $P(\chi) = \text{True}$ if and only if $P_1(\chi) = \text{True}$ and $P_2(\chi) = \text{True}$.

References

- [1] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: specification*, Springer-Verlag, Berlin, 1992.
- [2] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Safety*, Springer-Verlag, New York, 1995.
- [3] Nancy Lynch, *Distributed Algorithms*, Morgan Kaufmann, 1996.

ee291E Lecture 10: Stability

René Vidal & John Lygeros

February 24, 1999

1 Stability Definitions

Consider an autonomous hybrid automaton H .

Definition 1 (Equilibrium) $x = 0 \in X$ is an equilibrium point of H if:

1. $f(q, 0) = 0$ for all $q \in Q$, and
2. $((q, q') \in E) \wedge (0 \in G(q, q')) \Rightarrow R(q, q', 0) = \{0\}$.

Proposition 1 If $(q_0, 0) \in \text{Init}$ and $(\tau, q, x) \in \mathcal{H}_{(q_0, 0)}$ then $x(t) = 0$ for all $t \in \tau$.

- If the continuous part of the state starts on the equilibrium point, it stays there forever.
- One would like to characterize the notion that if the continuous state starts close to the equilibrium point it stays close, or even converges to it.
- Use the definitions of Lyapunov for this purpose.

Definition 2 (Stable Equilibrium) Let $x = 0 \in X$ be an equilibrium point of H . $x = 0$ is stable if for all $\epsilon > 0$ there exists $\delta > 0$ such that for all $(\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$ with $\|x_0\| < \delta$, $\|x(t)\| < \epsilon$ for all $t \in \tau$.

- Stability definition captures the notion of “start close, stay close”.
- $x = 0$ is called unstable if it is not stable.
- Stability is NOT a sequence property. The part “ $\|x_0\| < \delta$, $\|x(t)\| < \epsilon$ for all $t \in \tau$ ” by itself is a sequence property, but not the quantification over ϵ and δ .
- Stability does not imply convergence to the equilibrium.
- To analyze convergence, first consider an infinite execution, $\chi = (\tau, q, x)$, and let $\tau_\infty = \sum_i (\tau'_i - \tau_i)$. Notice that $\tau_\infty < \infty$ if χ is Zeno and $\tau_\infty = \infty$ otherwise.

Definition 3 (Asymptotically Stable Equilibrium) Let $x = 0 \in X$ be an equilibrium point of H . $x = 0$ is asymptotically stable if it is stable and there exists $\delta > 0$ such that for all $(\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}^\infty$ with $\|x_0\| < \delta$, $\lim_{t \rightarrow \tau_\infty} x(t) = 0$.

- Recall that each τ is fully ordered, so the limit is well defined.

- $\lim_{t \rightarrow \infty} x(t) = 0$ does not necessarily imply stability. As a counterexample consider the continuous system $\dot{x}_1 = x_1^2 - x_2^2$, $\dot{x}_2 = 2x_1x_2$.
- $\lim_{t \rightarrow \infty} x(t) = 0$ is a liveness property. $\|x_0\| < \delta$ and $\lim_{t \rightarrow \infty} x(t) = 0$ is the intersection of a safety property and a liveness property. Asymptotic stability is NOT a sequence property, since the stability part is not.
- There are variants of the stability definitions such as: global asymptotic stability, exponential stability, uniform stability, permutations thereof.
- See [1, 2] for details.

2 Continuous Systems

- A continuous dynamical system (Lecture 2) can be thought of as a hybrid automaton with $|\mathbf{Q}| = 1$ and $E = \emptyset$.
- For continuous systems we typically use energy arguments to prove stability properties.
- Roughly speaking, if the system “dissipates energy” along its executions it will be stable.
- Statement is clear for mechanical systems because energy has a physical interpretation.
- More generally, we use energy-like functions, known as Lyapunov functions.

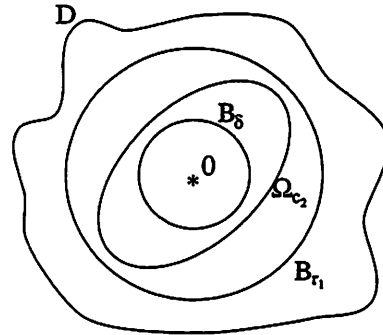
Theorem 1 *Let $x = 0$ be an equilibrium point of a continuous system H . Assume that there exists an open set $D \subseteq \mathbb{R}^n$ with $0 \in D$ and a continuously differentiable function $V : D \rightarrow \mathbb{R}$ such that:*

1. $V(0) = 0$,
2. $V(x) > 0$ for all $x \in D \setminus \{0\}$, and
3. $\frac{\partial V}{\partial x}(x)f(x) \leq 0$ for all $x \in D$.

Then $x = 0$ is a stable equilibrium of H .

Proof: For $r > 0$ let $B_r = \{x \in \mathbb{R}^n : \|x\| < r\}$, $S_r = \{x \in \mathbb{R}^n : \|x\| = r\}$ and $\Omega_r = \{x \in \mathbb{R}^n : V(x) \leq r\}$. Given ϵ :

- Choose $r_1 \in (0, \epsilon)$ such that $B_{r_1} \subseteq D$.
- Let $c_1 = \min_{x \in S_{r_1}} V(x)$.
- Choose $c_2 \in (0, c_1)$. Then $\Omega_{c_2} \subseteq B_{r_1} \subseteq B_\epsilon$.
- Choose $\delta > 0$ such that $B_\delta \subseteq \Omega_{c_2}$.



If $x_0 \in B_\delta$, then $V(x_0) < c_2$. Since V is not increasing along the system executions, executions that start inside B_δ can not leave Ω_{c_2} . Therefore $\forall t$ we have $x(t) \in \Omega_{c_2} \subset B_{r_1} \subset B_\epsilon$. Thus $\|x(t)\| \leq \epsilon$. ■

Theorem 2 If in addition, $\frac{\partial V}{\partial x}(x)f(x) < 0$ for all $x \in D \setminus \{0\}$, then $x = 0$ is an asymptotically stable equilibrium point.

Example 1 Consider the pendulum equations $\dot{x}_1 = x_2$, $\dot{x}_2 = -(g/l)\sin(x_1) - (k/m)x_2$. Consider the energy of the system as a Lyapunov candidate function, i.e., $V(x) = (g/l)(1 - \cos(x_1)) + (1/2)x_2^2$. Clearly $V(0) = 0$, $V(x) \neq 0 \forall x_1 \in (-2\pi, 2\pi) \setminus \{0\}, \forall x_2 \neq 0$, and $\dot{V}(x) = -(k/m)x_2^2 \leq 0$. Thus $x = 0$ is stable.

3 Hybrid Systems

One would expect that a hybrid system for which all individual continuous systems are stable would be stable, at least if $R(q, q', x) = \{x\}$ for all $(q, q') \in E$ and $x \in G(q, q')$. However, this is NOT necessarily the case:

Example 2 Consider the hybrid automaton, H , with:

- $Q = \{q_1, q_2\}$ and $X = \mathbb{R}^2$;
- $\text{Init} = Q \times \{x \in X : \|x\| > 0\}$;
- $f(q_1, x) = A_1x$ and $f(q_2, x) = A_2x$, with:

$$A_1 = \begin{bmatrix} -1 & 10 \\ -100 & -1 \end{bmatrix}, A_2 = \begin{bmatrix} -1 & 100 \\ -10 & -1 \end{bmatrix}$$

- $I(q_1) = \{x \in X : x_1x_2 \leq 0\}$ and $I(q_2) = \{x \in X : x_1x_2 \geq 0\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in X : x_1x_2 \geq 0\}$ and $G(q_2, q_1) = \{x \in X : x_1x_2 \leq 0\}$; and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$.

Proposition 2 $x = 0$ is an equilibrium of H .

Proof: $f(q_1, 0) = f(q_2, 0) = 0$ and $R(q_1, q_2, 0) = R(q_2, q_1, 0) = \{0\}$. ■

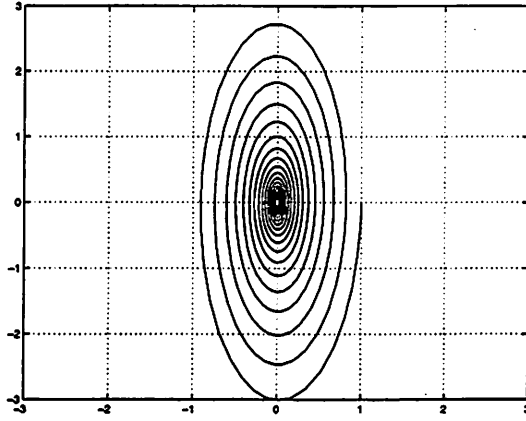
Proposition 3 The continuous systems $\dot{x} = A_i x$ for $i = 1, 2$ are asymptotically stable.

Proof: The eigenvalues of both systems are $-1 \pm j\sqrt{1000}$. ■

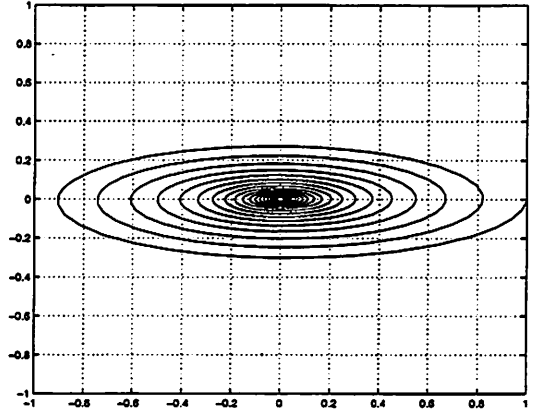
However, $x = 0$ is unstable for H !! (see in Figure 1(c)). Notice that if the switching were done the other way around the system would be stable (see Figure 1(d)). Therefore, in general we can not expect to analyze the stability of a hybrid system just by studying the individual continuous systems. We really have to study the switching.

Theorem 3 Consider a hybrid automaton H with $x = 0$ an equilibrium point, $|Q| < \infty$, and $R(q, q', x) = \{x\}$. Consider an open set $D \subseteq X$ with $0 \in D$ and a function $V : Q \times D \rightarrow \mathbb{R}$ continuously differentiable in x such that for all $q \in Q$:

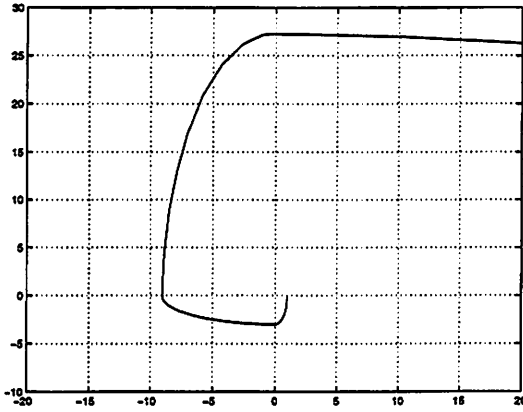
1. $V(q, 0) = 0$,
2. $V(q, x) > 0$ for all $x \in D \setminus \{0\}$, and
3. $\frac{\partial V}{\partial x}(q, x)f(q, x) \leq 0$ for all $x \in D$.



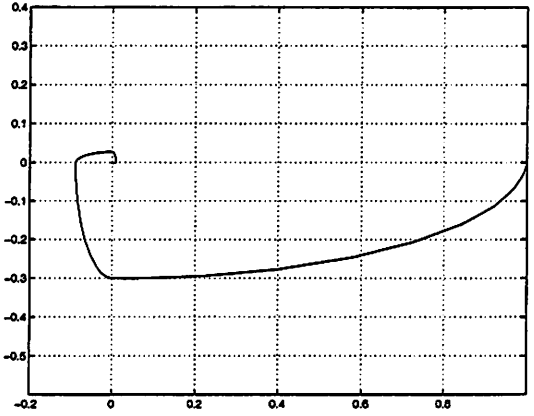
(a) Trajectory of $\dot{x} = A_1 x, x(0) = (1, 0)^T$



(b) Trajectory of $\dot{x} = A_2 x, x(0) = (1, 0)^T$



(c) Trajectory of $H, x(0) = (1, 0)^T$



(d) Trajectory of $H, x(0) = (1, 0)^T$ with the switching the other way around

Figure 1: Different trajectories for the system of Example 2

If for all $(\tau, q, x) \in \mathcal{H}$ and all $\hat{q} \in \mathbf{Q}$ the sequence $\{V(q(\tau_i), x(\tau_i)) : q(\tau_i) = \hat{q}\}$ is non increasing, then $x = 0$ is a stable equilibrium of H

- Can think of Theorem 3 as allowing one Lyapunov function for each q . (See Figure 2)
- When the discrete state q is in \hat{q} , corresponding Lyapunov function can not be increasing. Lyapunov functions corresponding to other states could however increase. In any case, every time we switch to a new discrete state, the value of the corresponding Lyapunov function can not be larger than what it was last time we switched there.
- More thorough treatment in [3, 4].

Proof: Consider the case $\mathbf{Q} = \{q_1, q_2\}$, the proof is similar if the system has more discrete states. $\Omega_r^q = \{x \in \mathbb{R}^n : V(q, x) \leq r\}$. The trick is that since every time we switch to a discrete state we are below where we were the previous time we switched to it the only thing that matters is the first time we switch to every discrete state.

Given $\epsilon > 0$:

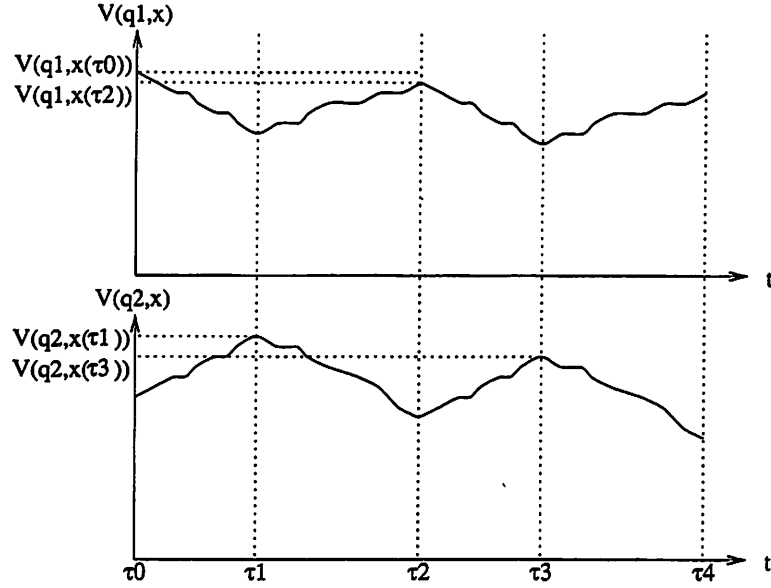


Figure 2: Evolution of Lyapunov functions

- Choose $r_1 \in (0, \epsilon)$ such that $B_{r_1} \subseteq D$.
- Set $c_1(q) = \min_{x \in S_{r_1}} V(q, x)$, $q \in \mathbf{Q}$.
- Choose $c_2(q) \in (0, c_1(q))$. Then $\Omega_{c_2(q)}^q \subseteq B_{r_1}$.
- Choose $r_2(q) > 0$ such that $B_{r_2(q)} \subseteq \Omega_{c_2(q)}^q$.
- Up to now same as individual Lyapunov proofs.
- Set $r = \min_{q \in \mathbf{Q}} r_2(q)$.
- If either individual system starts within r of the equilibrium, it stays within r .
- Set $c_3(q) = \min_{x \in S_r} V(q, x)$, $q \in \mathbf{Q}$.
- Choose $c_4(q) \in (0, c_3(q))$. Then $\Omega_{c_4(q)}^q \subseteq B_r$.
- Choose $r_4(q) > 0$ such that $B_{r_4(q)} \subseteq \Omega_{c_4(q)}^q$.
- Set $\delta = \min_{q \in \mathbf{Q}} r_4(q)$.

Take $(\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$, with $\|x_0\| < \delta$ and assume without loss of generality that $q_0 = q_1$. By a continuous Lyapunov argument, $x(t) \in \Omega_{c_4(q_1)}^{q_1} \subseteq B_r$ for $t \in [\tau_0, \tau'_0]$. Therefore, $x(\tau_1) = x(\tau'_0) \in \Omega_{c_2(q_2)}^{q_2}$. By a continuous Lyapunov argument, $x(t) \in \Omega_{c_2(q_2)}^{q_2} \subseteq B_\epsilon$ for $t \in [\tau_1, \tau'_1]$. By assumption, $x(\tau'_1) = x(\tau_2) \in \Omega_{c_4(q_1)}^{q_1}$. By a continuous Lyapunov argument, $x(t) \in \Omega_{c_4(q_1)}^{q_1} \subseteq B_r$ for $t \in [\tau_2, \tau'_2]$. The claim follows by induction. ■

Note: For the notation used in class please see the notes for Lecture 11.

References

- [1] Hassan B. Khalil, *Nonlinear Systems*, MacMillan, 1992.
- [2] Shankar Sastry, *Nonlinear Systems: Analysis, Stability and Control*, Springer-Verlag, 1999.

- [3] Michael S. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems", *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 475–482, 1998.
- [4] Hui Ye, Anthony Michel, and Ling Hou, "Stability theory for hybrid dynamical systems", *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 461–474, 1998.

ee291E Lecture 11: Lyapunov Stability Theorems for Hybrid Systems

John Koo

John Lygeros

February 24, 1999

1 Proof of Theorem 3, Lecture 10

The following theorem comes from [1]

Theorem 1 Consider a hybrid automaton H with $x = 0$ an equilibrium point, $|\mathbf{Q}| < \infty$, and $R(q, q', x) = \{x\}$. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and a function $V : \mathbf{Q} \times D \rightarrow \mathbb{R}$ continuously differentiable in x such that for all $q \in \mathbf{Q}$:

1. $V(q, 0) = 0$,
2. $V(q, x) > 0$ for all $x \in D \setminus \{0\}$,
3. $\frac{\partial V}{\partial x}(q, x)f(q, x) \leq 0$ for all $x \in D$.

If for all $(\tau, q, x) \in \mathcal{H}$ and all $\hat{q} \in \mathbf{Q}$ the sequence $\{V(q(\tau_i), x(\tau_i)) : q(\tau_i) = \hat{q}\}$ is non increasing, then $x = 0$ is a stable equilibrium of H

- Can think of Theorem 1 as allowing one Lyapunov function for each q .
- When discrete state in \hat{q} corresponding Lyapunov function can not be increasing.
- Every time we switch to a new discrete state, the value of the corresponding Lyapunov function can not be larger than what it was last time we switched there.

Proof: Consider the case $\mathbf{Q} = \{q_1, q_2\}$, the proof is similar if the system has more discrete states. Let $\Omega_r(q) = \{x \in \mathbb{R}^n : V(q, x) \leq r\}$. Since every time we switch to a discrete state we are bound to be below where we were the previous time we switched to it the only thing that matters is the first time we switch to every discrete state. The following construction takes advantage of that fact.

Consider an arbitrary $\epsilon > 0$. We will try to find $\delta > 0$ such that for all executions (τ, q, x) , $x(\tau_0) \in B_\delta$ implies $x(t) \in B_\epsilon$ for all $t \in \tau$.

- Choose $r \in (0, \epsilon)$ such that $B_r \subseteq D$.
- Set $a(q) = \min_{x \in S_r} V(q, x)$, for all $q \in \mathbf{Q}$.
- Choose $b(q) \in (0, a(q))$. Then $\Omega_{b(q)}(q) \subseteq B_r$.
- Choose $p(q) > 0$ such that $B_{p(q)} \subseteq \Omega_{b(q)}(q)$.
- Up to now same as individual Lyapunov proofs. If we knew that the system will always stay in one of the two discrete states, say q_1 , (i.e. $\tau < [\tau_0, \infty)$ and $q(\tau_0) = q_1$) and we started with $x(\tau_0) \in B_{p(q_1)}$ we would always stay in $\Omega_{b(q)}(q)$, therefore in B_r , therefore in B_ϵ .

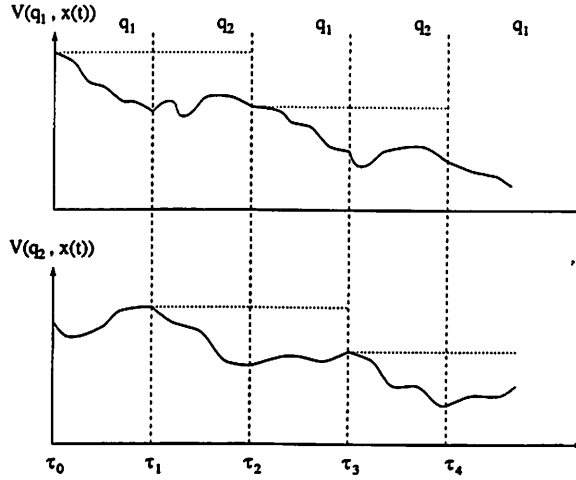


Figure 1: Lyapunov functions of two linear systems

- Set $s = \min_{q \in \mathbf{Q}} p(q)$.
- If we knew that the system will always stay in one discrete state whichever that may be, i.e. $\tau < [\tau_0, \infty)$ and we started with $x(\tau_0) \in B_s$, then we should always stay in B_r , therefore B_ϵ .
- Still have not taken care of switching. To do this we need to repeat the process.
- Set $c(q) = \min_{x \in S} V(q, x)$, for all $q \in \mathbf{Q}$.
- Choose $d(q) \in (0, c(q))$. Then $\Omega_{d(q)}(q) \subseteq B_s$.
- Choose $w(q) > 0$ such that $B_{w(q)} \subseteq \Omega_{d(q)}(q)$.
- Set $\delta = \min_{q \in \mathbf{Q}} w(q)$.

Take $(\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$, with $\|x_0\| < \delta$ and assume without loss of generality that $q_0 = q_1$.

- By a continuous Lyapunov argument, $x(t) \in \Omega_{d(q_1)}(q_1) \subseteq B_s \subseteq B_\epsilon$ for $t \in [\tau_0, \tau'_0]$.
- If $\tau < [\tau_0, \infty)$ we are done.
- Otherwise, note that by the assumption on R , $x(\tau_1) = x(\tau'_0) \in \Omega_{d(q_1)}(q_1) \subseteq B_s \subseteq \Omega_{b(q_2)}(q_2)$.
- By a continuous Lyapunov argument, $x(t) \in \Omega_{b(q_2)}(q_2) \subseteq B_r \subseteq B_\epsilon$ for all $t \in [\tau_1, \tau'_1]$.
- If $\tau < [\tau_0, \tau'_0][\tau_1, \infty)$ we are done.
- Otherwise, note that $q(\tau_2) = q(\tau_0) = q_1$.
- By the non-increasing sequence condition $V(q(\tau_2), x(\tau_2)) \leq V(q(\tau_0), x(\tau_0)) \leq d(q)$.
- Therefore, $x(\tau_2) \in \Omega_{d(q_1)}(q_1) \subseteq B_\epsilon$.
- And so on ... The claim follows by induction.

■

Remarks

- For $|\mathbf{Q}| > 2$ we need to repeat the nested construction $|\mathbf{Q}|$ times.

- Continuous argument needed to:
 - Bound the value of the Lyapunov function at the first switching time.
 - Guarantee stability if there are a finite number of transitions.
- As a consequence of the last remark, Theorem 1 of Lecture 10 for continuous dynamical systems follows from Theorem 1 of Lecture 11 as a corollary.
- Some other immediate Corollaries are the following ...

Corollary 1 Consider a hybrid automaton H with $x = 0$ an equilibrium point, $|\mathbf{Q}| < \infty$, and $R(q, q', x) = \{x\}$. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and assume there exists a function $V : D \rightarrow \mathbb{R}$ continuously differentiable in x such that:

1. $V(0) = 0$,
2. $V(x) > 0$ for all $x \in D \setminus \{0\}$,
3. $\frac{\partial V}{\partial x}(x)f(q, x) \leq 0$ for all $q \in \mathbf{Q}$ and all $x \in D$.

Then $x = 0$ is a stable equilibrium of H

Proof: Define $\hat{V} : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbb{R}$ by $\hat{V}(q, x) = V(x)$ for all $q \in \mathbf{Q}$, $x \in \mathbf{X}$ and apply Theorem 1. ■

Corollary 2 Consider a hybrid automaton H with $x = 0$ an equilibrium point, $|\mathbf{Q}| < \infty$, and assume $R(q, q', x)$ is non-expanding. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and a function $V : \mathbf{Q} \times D \rightarrow \mathbb{R}$ continuously differentiable in x such that for all $q \in \mathbf{Q}$:

1. $V(q, 0) = 0$,
2. $V(q, x) > 0$ for all $x \in D \setminus \{0\}$,
3. $\frac{\partial V}{\partial x}(q, x)f(q, x) \leq 0$ for all $x \in D$.

If for all $(\tau, q, x) \in \mathcal{H}$ and all $\hat{q} \in \mathbf{Q}$ the sequence $\{V(q(\tau_i), x(\tau_i)) : q(\tau_i) = \hat{q}\}$ is non increasing, then $x = 0$ is a stable equilibrium of H

R is non-expanding if for all $(q, q') \in E$, all $x, y \in \mathbf{X}$, all $x' \in R(e, x)$ and all $y' \in R(e, y)$:

$$\|x' - y'\| \leq \|x - y\|$$

In particular, setting $y = 0$ (and recalling that 0 being an equilibrium requires $R(e, 0) = \{0\}$) non-expanding implies that for all $e \in E$, for all $x \in \mathbf{X}$, and all $x' \in R(e, x)$

$$\|x'\| \leq \|x\|$$

Notice that unless the non-expanding assumption is made the proof of Theorem 1 breaks down at the time of the first jump. I believe that R being a continuous function will also work, but I did not bother to prove so (it requires some more argument).

2 Other Lyapunov-like Theorems

More general reset relations are also covered by the following theorem (which can be found in [2]).

Theorem 2 *Consider a hybrid automaton H with $|\mathbf{Q}| < \infty$. Consider an open set $D \subseteq \mathbf{X}$ with $0 \in D$ and a function $V : \mathbf{Q} \times D \rightarrow \mathbb{R}$ continuous in x with $V(q, 0) = 0$ and $V(q, x) > 0$ for all $x \in D \setminus \{0\}$. Assume that for all $(\tau, q, x) \in \mathcal{H}$ the sequence $\{V(q(\tau_i), x(\tau_i))\}$ is non increasing and that there exists a continuous function $g : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ with $g(0) = 0$, such that for all $t \in [\tau_i, \tau'_i]$, $V(q(t), x(t)) \leq g(V(q(\tau_i), x(\tau_i)))$. Then $x = 0$ is a stable equilibrium of H*

Remarks:

- The conditions of Theorem 2 are weaker than those of Theorem 1 since:
 - R is not constrained.
 - V is not required to be decreasing along continuous evolution as long as it remains bounded by g . If the Lyapunov function happens to be decreasing, $g(x) = x$ will work.
- The conditions of Theorem 2 are stronger than those of Theorem 1 since they require the sequence over all i to be non-increasing (not just the individual subsequences for which $q(\tau_i) = \hat{q}$).

Theorems 1 and 2 can in fact be applied to more general invariant sets.

- A set $S \subseteq \mathbf{Q} \times \mathbf{X}$ is called invariant if for all $(\tau, q, x) \in \mathcal{H}_{(q_0, x_0)}$, $(q_0, x_0) \in S$ implies that $(q(t), x(t)) \in S$ for all $t \in \tau$. Watch out for the overloading of the terminology. The term invariant has already been used twice: in the definition of hybrid automata to describe sets of continuous states for which continuous evolution at a particular discrete state is allowed, and also to describe sets of states that contain all the reachable states.
- An equilibrium point $x = 0$ is a special case of an invariant set, of the form $S = \mathbf{Q} \times \{0\}$.
- For continuous systems, other types of invariant sets include limit cycles, level sets of Lyapunov functions, etc.
- The above theorems extend to more general invariant sets. The proofs are obtained by replacing the Euclidean norm $\|\cdot\|$ by the “distance to the set S ” defined as:

$$d(q, x) = \min_{(\hat{q}, \hat{x}) \in S} (d_D(q, \hat{q}) + \|x - \hat{x}\|)$$

where $d_D(q, \hat{q})$ is the discrete metric defined by $d_D(q, \hat{q}) = 0$ if $q = \hat{q}$ and $d_D(q, \hat{q}) = 1$ otherwise.

This and extensions to asymptotic stability, exponential stability, boundedness and converse theorems can be found in [2].

3 Example

Consider the hybrid automaton, H , with:

- $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}^2$;
- $\text{Init} = \mathbf{Q} \times \{x \in \mathbf{X} : \|x\| > 0\}$;

- $f(q_1, x) = A_1 x$ and $f(q_2, x) = A_2 x$, with:

$$A_1 = \begin{bmatrix} -1 & 10 \\ -100 & -1 \end{bmatrix}, A_2 = \begin{bmatrix} -1 & 100 \\ -10 & -1 \end{bmatrix}$$

- $I(q_1) = \{x \in \mathbf{X} : Cx \geq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : Cx \leq 0\}$ for some $C^T \in \mathbb{R}^2$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbf{X} : Cx \leq 0\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : Cx \geq 0\}$; and
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$.

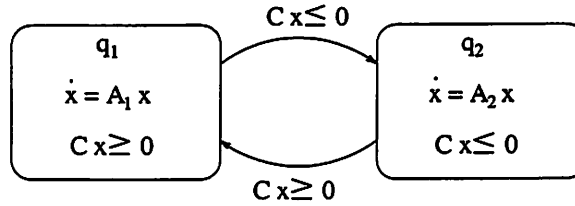


Figure 2: The switching system

We show that $x = 0$ is a stable equilibrium point of H for all $C^T \in \mathbb{R}^2$.

Proposition 1 $x = 0$ is an equilibrium of H .

Proof: $f(q_1, 0) = f(q_2, 0) = 0$ and $R(q, q', 0) = \{0\}$. ■

Proposition 2 The continuous systems $\dot{x} = A_i x$ for $i = 1, 2$ are asymptotically stable.

Proposition 3 If $\dot{x} = A_i x$ is stable, then there exists $P_i = P_i^T > 0$ such that $A_i^T P_i + P_i A_i = -I$.

Proof: See [3]. ■

Recall that $P_i > 0$ (positive definite) if and only if $x^T P_i x > 0$ for all $x \neq 0$, and I is the identity matrix.

Consider the candidate Lyapunov function:

$$V(q, x) = \begin{cases} x^T P_1 x & \text{if } q = q_1 \\ x^T P_2 x & \text{if } q = q_2 \end{cases}$$

Check that the conditions of the Theorem hold. For all $q \in \mathbf{Q}$:

1. $V(q, 0) = 0$
2. $V(q, x) > 0$ for all $x \neq 0$ (since the P_i are positive definite)
3. $\frac{\partial V}{\partial x}(q, x) f(q, x) \leq 0$ for all x since:

$$\begin{aligned} \frac{\partial V}{\partial x}(q, x) f(q, x) &= \frac{d}{dt} V(q, x(t)) \\ &= \dot{x}^T P_i x + x^T P_i \dot{x} \\ &= x^T A_i^T P_i x + x^T P_i A_i x \\ &= x^T (A_i^T P_i + P_i A_i) x \\ &= -x^T I x \\ &= -\|x\|^2 \leq 0 \end{aligned}$$

It remains to test the non-increasing sequence condition. Notice that the level sets of $x^T P_i x$ are ellipses centered at the origin. Therefore each level set intersects the switching line $Cx = 0$ at exactly two points, \hat{x} and $-\hat{x}$. Assume $x(\tau_i) = \hat{x}$ and $q(\tau_i) = q_1$. The fact that $V(q_1, x(t))$ decreases for $t \in [\tau_i, \tau'_i]$ (where $q(t) = q_1$) implies that the next time the switching line is reached, $x(\tau'_i) = -\hat{x}\alpha$ for some $\alpha \in (0, 1)$. Therefore, $\|x(\tau_{i+1})\| = \|x(\tau'_i)\| < \|x(\tau_i)\|$. By a similar argument, $\|x(\tau_{i+2})\| = \|x(\tau'_{i+1})\| < \|x(\tau_{i+1})\|$ and $Cx(\tau_{i+2}) = 0$. Therefore, $q(\tau_i) = q(\tau_{i+2}) = q_1$ and $V(q(\tau_i), x(\tau_i)) \leq V(q(\tau_{i+2}), x(\tau_{i+2}))$.

Remark: For hybrid systems like this, where the dynamics in each one of the discrete states is linear (or affine), it is possible to obtain piecewise quadratic Lyapunov functions computationally, using convex optimization techniques (in particular, Linear Matrix Inequalities) [4]. A (strict) linear matrix inequality(LMI) has the form

$$F(x) = F_0 + \sum_{i=1}^m x_i F_i > 0, \quad (1)$$

where $x \in \mathbb{R}^m$ is the variable and the symmetric matrices $F_i = F_i^T \in \mathbb{R}^{n \times n}$, $i = 0, \dots, m$, are given. In (1), $F(x)$ is positive definite. There is also non-strict LMIs, which has the form

$$F(x) \geq 0$$

For solving the Lyapunov inequality of system $\dot{x} = Ax$

$$A^T P + P A < 0,$$

$A \in \mathbb{R}^{n \times n}$ is assumed to be given, and $P = P^T$ is treated as variable. Let P_1, \dots, P_m be a basis for symmetric $n \times n$ matrices ($m = n(n+1)/2$), and take $F_0 = 0$, $F_i = -A^T P_i - P_i A$.

To find a single Lyapunov function satisfies the conditions as shown in Corollary 1, consider the “simultaneous Lyapunov stability problem”. We are given $A_j \in \mathbb{R}^{n \times n}$, $j = 1, \dots, L$, and need to find P satisfying the LMI

$$P > 0, \quad A_j^T P + P A_j < 0, \quad j = 1, \dots, L,$$

or determine that no such P exists. In literature, it is called polytopic LDIs, Linear Differential Inclusions, problem where the vector field is a convex combinations of $A_j x$, $j = 1, \dots, L$. For details about LMIs, please refer to [5].

References

- [1] Michael S. Branicky, “Multiple Lyapunov functions and other analysis tools for switched and hybrid systems”, *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 475–482, 1998.
- [2] Hui Ye, Anthony Michel, and Ling Hou, “Stability theory for hybrid dynamical systems”, *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 461–474, 1998.
- [3] Frank M. Callier and Charles A. Desoer, *Linear System Theory*, Springer-Verlag, 1991.
- [4] M. Johansson and A. Rantzer, “Computation of piecewise quadratic lyapunov functions for hybrid systems”, *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 555–559, 1998.
- [5] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM, 1994.

ee291E Lecture 12: Reachability

H. Jin Kim and John Lygeros

March 3, 1999

1 Reachability and Sequence Properties

- After a short aside on stability, we return to sequence properties, to study reachability.
- The problem we will address is, given a hybrid automaton H to compute $\text{Reach}(H) \subseteq \mathbf{Q} \times \mathbf{X}$.
- If we can solve this problem we can also answer questions about safety properties.

Proposition 1 H satisfies $(Q \cup X, \Box G)$ for $G \subseteq \mathbf{Q} \times \mathbf{X}$ if and only if $\text{Reach}(H) \subseteq G$.

- The interpretation is that G is a “good” set of state you would always like to stay in. Equivalently $B = G^c$ is a “bad” set of states you would like to keep away from.
- Different methods have been proposed for solving the reachability problem:
 1. **Optimal Control:** See [1, 2]. The role of the “control” is often played by the non-determinism of the system.
 2. **Deductive Techniques:** See [3, 4, 5]. Establish invariant sets to bound $\text{Reach}(H)$.
 3. **Model Checking Techniques:** See [6, 7, 8, 9]. Automatically compute $\text{Reach}(H)$. Require one to be able to “compute” with sets of states. Class of systems to which it applies is inherently limited.
 4. **Approximation:** Works with all of the above.
 - For optimal control, approximate by sets and dynamics for which optimal control problems are easier to solve. (e.g. ellipsoidal sets and linear dynamics)
 - Deductive techniques are inherently based on over-approximation.
 - For model checking, approximate by a system you can compute with.
 - Also possible to do “brute force” over-approximation, based, for example, on gridding [10, 11].
- In all cases you stop once your question has been answered. For example, if your question was “does H satisfy $(Q \cup X, \Box G)$?”, you stop once you find a reachable state outside G .
- For approximation, you typically “over-approximate”.
- All methods are supported by computational tools:
 1. Optimal Control techniques typically use optimal control and convex optimization tools.
 2. Deductive techniques typically use theorem provers.
 3. Model checking techniques typically use model checkers.
 4. Approximation has been done using “gridding” or polynomial manipulation packages.

- **Simulation** can also be used for reachability investigations. It is not a formal method however since:
 - It is a shot in the dark: We simulate. If B is reached then fine, else we have to choose another initial condition and try again.
 - There is no termination guarantee.

Still it is the best we can do for many classes of hybrid systems.

2 General Transition Systems

Definition 1 (Transition System) *A transition system is a collection $T = (S, \Sigma, \rightarrow, S_0, S_F)$, where*

- S is a set of states;
- Σ is an alphabet of events;
- $\rightarrow \subseteq S \times \Sigma \times S$ is a transition relation;
- $S_0 \subseteq S$ is a set of initial states; and,
- $S_F \subseteq S$ is a set of final states.

Example: A finite automaton $M = (Q, \Sigma, \Delta, q_0, F)$, is a transition system with:

- $S = Q$
- Σ the same
- $\rightarrow = \Delta$
- $S_0 = \{q_0\}$
- $S_F = F$

Example: A hybrid automaton $H = (Q, X, \text{Init}, f, I, E, G, R)$ and a safety property $(Q \cup X, \Box G)$ form a transition system with:

- $S = Q \times X$
- $\Sigma = E \cup \{\tau\}$
- $\rightarrow = \{ \text{discrete transitions} \} \cup \{ \text{continuous evolution} \}$, all of which are characterized by G, I and R
- $S_0 = \text{Init}$
- $S_F = G^c$

Problem 1 (Reachability) *Given a transition system T , is any state $s_f \in S_F$ reachable from a state $s_0 \in S_0$ by a sequence of transitions?*

Remark: For finite automata we can always “decide” reachability problems by brute force.

Example: Consider for example the finite automaton of Figure 1. This is a transition system with $S = \{q_0, q_1, \dots, q_6\}$, $\Sigma = \{a, b, c\}$, $\rightarrow = \{(q_0, a, q_1), (q_0, a, q_2), (q_0, b, q_0), \dots\}$, $S_0 = \{q_0\}$, $S_F = \{q_3, q_6\}$. We can start “exploring” from q_0 and keep going until we either visit a state in S_F or have nowhere else to go (i.e., have visited all reachable states).

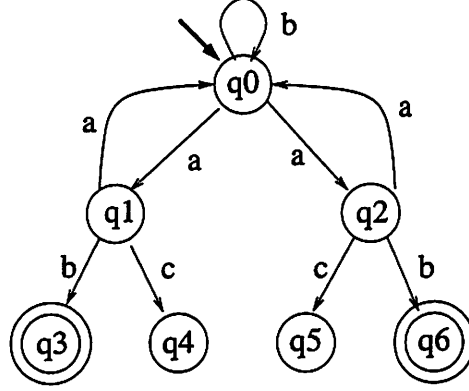


Figure 1: Reachability example for finite automata

- More formally, the set of reachable states for a transition system can be computed using the following “algorithm”:

Algorithm 1 (Reachability)

Initialization:

$\text{Reach}_0 = S_0$

$\text{Reach}_{-1} = \emptyset$

$i = 0$

while $\text{Reach}_i \neq \text{Reach}_{i-1}$ **do**

begin

$\text{Reach}_{i+1} = \text{Reach}_i \cup \{s' \in S : \exists s \in \text{Reach}_i, \sigma \in \Sigma \text{ with } (s, \sigma, s') \in \rightarrow\}$

$i = i + 1$

end

- If the algorithm can be implemented, it terminates and upon termination $\text{Reach}_i \cap S_F = \emptyset$, then the answer to the reachability problem is ‘no’.
- For the example of Figure 1, $\text{Reach}_{-1} = \emptyset$, $\text{Reach}_0 = \{q_0\}$, $\text{Reach}_1 = \{q_0, q_1, q_2\}$ and $\text{Reach}_2 = Q$.
- For finite automata the algorithm *can be implemented* and *always terminates*. What properties of finite automata allow us to do this?
 1. We can represent states in a finite way (by enumeration).
 2. We can represent transitions among states in a finite way (by enumeration)
 3. We are guaranteed that if we start exploring a particular execution, after a finite number of steps we will either reach a state we have visited already, or a state from which we have nowhere else to go, or a final state.
- **Remark:** Enumeration is a fairly naive way of going about reachability analysis. In practice, sets of states are not enumerated, but are represented more compactly, e.g. by BDD’s.

3 Bisimulation

- In the example of Figure 1, q_1 and q_2 have very similar properties, since they can both be reached from q_0 by a and all subsequent executions look similar. This suggests that these states are in some sense “equivalent”. In this section we will try to make this statement more precise by introducing the notion of bisimulation.

- Consider the set of states S . A *relation* on S is a subset of $S \times S$.

Definition 2 (Equivalence Relation) A relation $\sim \subseteq S \times S$ is called an *equivalence relation* if it is:

1. *Reflexive*: $(s, s) \in \sim$ for all $s \in S$;
2. *Symmetric*: $(s, s') \in \sim$ implies that $(s', s) \in \sim$; and,
3. *Transitive*: $(s, s') \in \sim$ and $(s', s'') \in \sim$ imply $(s, s'') \in \sim$.

- For simplicity we write $s \sim s'$ instead of $(s, s') \in \sim$ and say that s is equivalent to s' .
- Examples of equivalence relations:
 1. Equality
 2. $S \times S$
 3. Let S be inhabitants of U.S. and $x \sim y$ mean that person x lives in the same state as person y . Then \sim is an equivalence relation.
- An equivalence relation partitions S to a number of *equivalence classes*:

$$S = \bigcup_i S_i$$

such that for all $s, s' \in S$, $s, s' \in S_i$ if and only if $s \sim s'$.

- Equivalence classes cover S (Any $s \in S$ belongs to an equivalence class by reflexivity).
- Equivalence classes are disjoint (Assume $\exists s \in S$ such that $s \in S_1, s \in S_2$ for $S_1 \neq S_2$. Then $\exists s_1 \in S_1$ and $s_2 \in S_2$ such that $s_1 \sim s$ and $s_2 \sim s$. By transitivity, $s_1 \sim s_2$, which contradicts the assumption.).
- Example of the equivalence classes
 1. For equality the equivalence classes are the singletons.
 2. For $S \times S$ there is only one equivalence class, S itself.
 3. For 3 in the example above, the equivalence classes are the states of U.S.
- Given an equivalence relation \sim , let $S/\sim = \{S_i\}$ denote the *quotient space*, i.e. the set consisting of all equivalence classes.
- Given a set $P \subseteq S$, let P/\sim represent the part of the quotient space with which P overlaps:

$$P/\sim = \{S_i : S_i \cap P \neq \emptyset\} \subseteq S/\sim$$

- If S are the states of a transition system, $T = (S, \Sigma, \rightarrow, S_0, S_F)$, define the *quotient transition system* as

$$T/\sim = (S/\sim, \Sigma, \rightarrow_\sim, S_0/\sim, S_F/\sim)$$

where for $S_1, S_2 \in S/\sim$, $(S_1, \sigma, S_2) \in \rightarrow_\sim$ if and only if there exist $s_1 \in S_1$ and $s_2 \in S_2$ such that $(s_1, \sigma, s_2) \in \rightarrow$.

- Notice that the quotient transition system may be “non-deterministic”, even if the original system is.
- For $\sigma \in \Sigma$ define the $\text{Pre}_\sigma : 2^S \rightarrow 2^S$ operator as:

$$\text{Pre}_\sigma(P) = \{s \in S : \exists s' \in P \text{ such that } (s, \sigma, s') \in \rightarrow\}$$

- In the example of Figure 1, if $P = \{q_3, q_4, q_5, q_6\}$ then $\text{Pre}_\sigma(P) = \{q_1, q_2\}$.

References

- [1] A.B. Kurzhanski and P. Varaiya, “Ellipsoidal techniques for reachability analysis”, (preprint), 1998.
- [2] Anuj Puri and Pravin Varaiya, “Driving safely in smart cars”, in *American Control Conference*, 1995, pp. 3597–3599.
- [3] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Safety*, Springer-Verlag, New York, 1995.
- [4] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H.B. Weinberg, “Hybrid I/O automata”, in *Hybrid Systems III*, number 1066 in LNCS, pp. 496–510. Springer Verlag, 1996.
- [5] John Lygeros and Nancy Lynch, “Strings of vehicles: Modeling and safety conditions”, in *Hybrid Systems: Computation and Control*, S. Sastry and T.A. Henzinger, Eds., number 1386 in LNCS, pp. 273–288. Springer Verlag, 1998.
- [6] R. P. Kurshan, *Computer-aided verification of coordinating processes; the automata-theoretic approach*, Princeton University Press, 1994.
- [7] R. Alur and D. Dill, “A theory of timed automata”, *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.
- [8] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, “The algorithmic analysis of hybrid systems”, *Theoretical Computer Science*, vol. 138, pp. 3–34, 1995.
- [9] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya, “What’s decidable about hybrid automata”, in *27th Annual Symposium on the Theory of Computing, STOC’95*. 1995, pp. 373–382, ACM Press.
- [10] M.R. Greenstreet and I. Mitchell, “Integrating projections”, in *Hybrid Systems: Computation and Control*, S. Sastry and T.A. Henzinger, Eds., number 1386 in LNCS, pp. 159–174. Springer Verlag, 1998.
- [11] T. Dang and O. Maler, “Reachability analysis via face lifting”, in *Hybrid Systems: Computation and Control*, S. Sastry and T.A. Henzinger, Eds., number 1386 in LNCS, pp. 96–109. Springer Verlag, 1998.

ee291E Lecture 13: Bisimulation

Claudio Pinello & John Lygeros

March 8, 1999

1 Recall that ...

Definition 1 (Transition System) A transition system is a collection $T = (S, \Sigma, \rightarrow, S_0, S_F)$, where

- S is a set of states (finite or infinite);
- Σ is an alphabet of events;
- $\rightarrow \subseteq S \times \Sigma \times S$ is a transition relation;
- $S_0 \subseteq S$ is a set of initial states; and,
- $S_F \subseteq S$ is a set of final states.

Definition 2 (Equivalence Relation) A relation $\sim \subseteq S \times S$ is called an equivalence relation if it is:

1. *Reflexive*: $(s, s) \in \sim$ for all $s \in S$;
2. *Symmetric*: $(s, s') \in \sim$ implies that $(s', s) \in \sim$; and,
3. *Transitive*: $(s, s') \in \sim$ and $(s', s'') \in \sim$ imply $(s, s'') \in \sim$.

- For simplicity we write $s \sim s'$ instead of $(s, s') \in \sim$ and say s is equivalent to s' .
- An equivalence relation partitions S into a number of *equivalence classes*:

$$S = \bigcup_i S_i$$

such that for all $s, s' \in S$, $s, s' \in S_i$ if and only if $s \sim s'$.

- Given an equivalence relation \sim , let $S / \sim = \{S_i\}$ denote the quotient space, i.e. the set consisting of all equivalence classes.
- Given a set $P \subseteq S$, let P / \sim represent the part of the quotient space with which P overlaps:

$$P / \sim = \{S_i \in S / \sim : S_i \cap P \neq \emptyset\}, \text{ hence } P / \sim \subseteq S / \sim$$

- If S are the states of a transition system, $T = (S, \Sigma, \rightarrow, S_0, S_F)$, define the *quotient transition system* as

$$T / \sim = (S / \sim, \Sigma, \rightarrow_{\sim}, S_0 / \sim, S_F / \sim)$$

where for $S_1, S_2 \in S / \sim$, $(S_1, \sigma, S_2) \in \rightarrow_{\sim}$ if and only if there exist $s_1 \in S_1$ and $s_2 \in S_2$ such that $(s_1, \sigma, s_2) \in \rightarrow$.

- For $\sigma \in \Sigma$ define the $\text{Pre}_{\sigma} : 2^S \rightarrow 2^S$ operator as:

$$\text{Pre}_{\sigma}(P) = \{s \in S : \exists s' \in P \text{ such that } (s, \sigma, s') \in \rightarrow\}$$

2 Bisimulation

Definition 3 (Bisimulation) Given $T = (S, \Sigma, \rightarrow, S_0, S_F)$, and \sim an equivalence relation over S , \sim is called a bisimulation if:

1. S_0 is a union of equivalence classes;
 2. S_F is a union of equivalence classes;
 3. For all $\sigma \in \Sigma$, if P is a union of equivalence classes, $\text{Pre}_\sigma(P)$ is also a union of equivalence classes.
- If \sim is a bisimulation, T and T/\sim are called bisimilar.
 - Equality is a bisimulation.
 - $S \times S$, the “total relation”, is in general not a bisimulation. The only equivalence classes are S, \emptyset .
 - In a sense bisimilar transition systems generate the same sequences of transitions (language).
 - Therefore, if \sim is a bisimulation, we need not distinguish between the elements of an equivalence class.
 - More specifically, if a state in S_F is reachable from a state in S_0 , a state in S_F/\sim is reachable from a state in S_0/\sim .

Proposition 1 \sim is a bisimulation if and only if for all $(s_1 \sim s_2)$:

1. $(s_1 \in S_0) \Rightarrow (s_2 \in S_0)$;
2. $(s_1 \in S_F) \Rightarrow (s_2 \in S_F)$; and,
3. $((s_1, \sigma, s'_1) \in \rightarrow) \Rightarrow \exists s'_2 \text{ such that } (s'_1 \sim s'_2) \wedge ((s_2, \sigma, s'_2) \in \rightarrow)$.

Proof: (\Rightarrow) :

1. If for all $s_1 \in S_0$, $s_1 \sim s_2$ implies $s_2 \in S_0$, then S_0 must be a union of equivalence classes.
2. Similarly for S_F .
3. If P is an equivalence class and $s_1 \in \text{Pre}_\sigma(P)$, then $s_2 \in \text{Pre}_\sigma(P)$ for all $s_2 \sim s_1$. Hence $\text{Pre}_\sigma(P)$ must be a union of equivalence classes.

(\Leftarrow) : similar ■

Aside: More generally, two transition systems, $T = (S, \Sigma, \rightarrow, S_0, S_F)$ and $T' = (S', \Sigma, \rightarrow', S'_0, S'_F)$ are called *bisimilar* if there exists a relation $\sim \subseteq S \times S'$ such that:

1. $(s_1 \sim s_2) \wedge (s_1 \in S_0) \Rightarrow (s_2 \in S'_0)$;
2. $(s_1 \sim s_2) \wedge (s_2 \in S'_0) \Rightarrow (s_1 \in S_0)$;
3. $(s_1 \sim s_2) \wedge (s_1 \in S_F) \Rightarrow (s_2 \in S'_F)$;
4. $(s_1 \sim s_2) \wedge (s_2 \in S'_F) \Rightarrow (s_1 \in S_F)$;
5. $(s_1 \sim s_2) \wedge ((s_1, \sigma, s'_1) \in \rightarrow) \Rightarrow \exists s'_2 \text{ such that } (s'_1 \sim s'_2) \wedge ((s_2, \sigma, s'_2) \in \rightarrow')$.
6. $(s_1 \sim s_2) \wedge ((s_2, \sigma, s'_2) \in \rightarrow') \Rightarrow \exists s'_1 \text{ such that } (s'_1 \sim s'_2) \wedge ((s_1, \sigma, s'_1) \in \rightarrow)$.

Three of the above conditions are taken care of automatically if $T' = T / \sim_1$. To see this simply take an equivalence relation $\sim_1 \subseteq S \times S$ and consider the corresponding $T' = T / \sim_1$ defined as usual:

- $S' = S / \sim_1$
- $S'_0 = S_0 / \sim_1$
- $S'_F = S_F / \sim_1$
- $\rightarrow' = \rightarrow / \sim_1$

Correspondingly let $\sim \subseteq S \times S'$ be the relation (not equivalence since $S \neq S'$) defined by

$$s_1 \sim s_2 \Leftrightarrow s_1 \in S_1.$$

Namely \sim is the belonging relation \in .

Again T and T' being bisimilar can be interpreted as T and T' accepting the same sequences of events.

3 Computing Bisimulations

- Bisimulations look useful since they preserve the language of the transition system.
- How does one find a bisimulation?

Finding an equivalence relation is equivalent to finding the equivalence classes in S / \sim . Along this line, the following algorithm tries to determine a partition of S so that the corresponding equivalence relation is a bisimulation.

Algorithm 1 (Bisimulation)

```

Initialization:  $S / \sim = \{S_0, S_F, S \setminus (S_0 \cup S_F)\}$ 
while  $\exists P, P' \in S / \sim$  and  $\sigma \in \Sigma$  such that  $P \cap \text{Pre}_\sigma(P') \neq P$  and  $P \cap \text{Pre}_\sigma(P') \neq \emptyset$  do
  begin
     $P_1 = P \cap \text{Pre}_\sigma(P')$ 
     $P_2 = P \setminus \text{Pre}_\sigma(P')$ 
     $S / \sim = (S / \sim \setminus \{P\}) \cup \{P_1, P_2\}$ 
  end

```

- Note: the initialization assumes that $S_0 \cap S_F = \emptyset$, $S_0 \neq \emptyset$ and $S_F \neq \emptyset$ which is the most interesting case when doing reachability analysis. If $S_0 \cap S_F \neq \emptyset$ we can start by setting $S / \sim = \{S_0 \setminus S_F, S_F \setminus S_0, S_0 \cap S_F, S \setminus (S_0 \cup S_F)\}$.
- If the algorithm terminates, \sim is a bisimulation, since:
 1. S_0 is a union of equivalence classes (note that $S_0 \in S / \sim$ initially, and we only split sets)
 2. S_F is a union of equivalence classes (for the same reason).
 3. Termination implies that for all $P' \in S / \sim$ and for all σ , $P \cap \text{Pre}_\sigma(P')$ is either equal to P or equal to \emptyset , therefore, $\text{Pre}_\sigma(P')$ is a union of equivalence classes.
- This is again a pseudo algorithm. Implementation and termination for general transition systems are not obvious. For finite state systems we can implement the algorithm and guarantee that it terminates because we can enumerate the states for the finite state system.
- Figure 1 shows the results of applying this algorithm to the finite state example of Lecture 12.

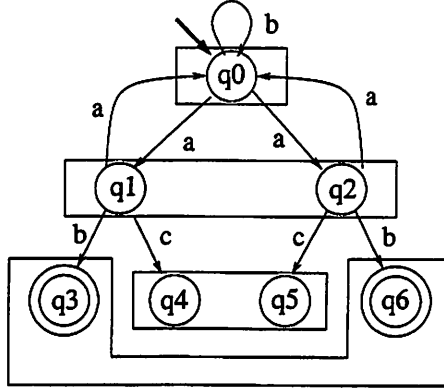


Figure 1: Bisimulation of previous example

- Why is this an improvement?
 1. To test for reachability, we need not search through the states, just the equivalence classes. Therefore may have a computational advantage.
 2. Extends to systems with infinite states. If the bisimulation quotient can be computed and is finite, then the reachability computation is decidable.

4 Bisimulations of Timed Automata

- The original definition of a timed automaton, found in [1] is given below. Some extensions will be discussed in subsequent lectures.
- Consider $X = \{x_1, \dots, x_n\}$ a finite collection of variables, each of which takes values in \mathbb{R} .
- Let $x = (x_1, \dots, x_n) \in \mathbb{R}^n = \mathbf{X}$ denote a valuation for all $x_i \in X$.

Definition 4 (Clock Constraints) The set, $\Phi(X)$, of clock constraints of X , is a set of finite logical expressions defined inductively by $\delta \in \Phi(X)$ if:

$$\delta := (x_i \leq c) \mid (x_i \geq c) \mid \neg \delta_1 \mid \delta_1 \wedge \delta_2$$

where $\delta_1, \delta_2 \in \Phi(X)$, $x_i \in X$ and $c \geq 0$ is a rational number.

Note: finite means that δ can involve a finite number of sub-formulas of the form $(x_i \leq c)$ or $(x_i \geq c)$.

- Examples: let $X = \{x_1, x_2\}$.
 - $(x_1 \leq 1) \in \Phi(X)$
 - $(0 \leq x_1 \leq 1) \in \Phi(X)$, since $(0 \leq x_1 \leq 1) \Leftrightarrow (x_1 \geq 0) \wedge (x_1 \leq 1)$
 - $(x_1 = 1) \in \Phi(X)$, since $(x_1 = 1) \Leftrightarrow (x_1 \geq 1) \wedge (x_1 \leq 1)$
 - $(x_1 < 1) \in \Phi(X)$, since $(x_1 < 1) \Leftrightarrow \neg(x_1 \geq 1)$
 - $\text{True} \in \Phi(X)$, since $\text{True} \Leftrightarrow \neg((x_1 \leq 0) \wedge (x_1 \geq 1))$
 - $(x_1 \leq x_2) \notin \Phi(X)$
- Given $\delta \in \Phi(X)$, we say $x \in \mathbf{X}$ satisfies δ IFF $\delta(x) = \text{True}$.

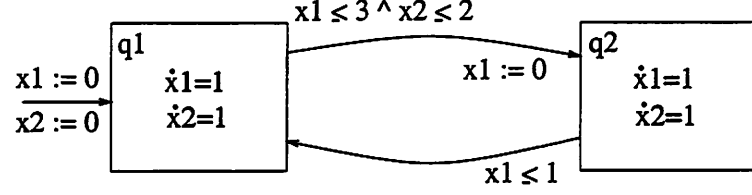


Figure 2: A timed automaton

- To each $\delta \in \Phi(X)$ we can associate a set:

$$\hat{\delta} = \{x \in X : \delta(x) = \text{True}\}$$

Definition 5 (Timed Automaton) A *timed automaton* is a hybrid automaton $H = (Q, X, \text{Init}, f, \text{Inv}, E, G, R)$, where

- Q is a set of discrete variables, $Q = \{q_1, \dots, q_m\}$;
- $X = \{x_1, \dots, x_n\}$, $X = \mathbb{R}^n$;
- $\text{Init} = \{\{q_i\} \times \widehat{\text{Init}_{q_i}}\}_{i=1}^m$ where $\text{Init}_{q_i} \in \Phi(X)$;
- $f(q, x) = (1, \dots, 1)$ for all (q, x) ;
- $\text{Inv}(q) = X$ for all $q \in Q$;
- $E \subset Q \times Q$;
- $G(e) = \widehat{G_e}$ where $G_e \in \Phi(X)$, for all $e = (q, q') \in E$; and
- For all e , $R(e, x)$ either leaves x_i unaffected or resets it to 0.

Remarks

- Notice that R is single valued.
- Consider a set of final states of the form $F = \{\{q_i\} \times \widehat{F_{q_i}}\}_{i=1}^m$ where $F_{q_i} \in \Phi(X)$.
- A timed automaton together with a set of final states, F , can be viewed as a transition system, $T = (S, \Sigma, \rightarrow, S_0, S_F)$, with:
 - $S = Q \times X$;
 - $\Sigma = E \cup \{\tau\}$, where τ is a symbol denoting time passage;
 - $((q, x), e, (q', x')) \in \rightarrow$ if $e = (q, q') \in E$, $G_e(x) = \text{True}$ and $x' \in R(e, x)$.
 - $((q, x), \tau, (q', x')) \in \rightarrow$ if $q = q'$, and there exists $t \geq 0$ such that $x' = x + t(1, \dots, 1)$.
 - $S_0 = \text{Init}$; and,
 - $S_F = F$.
- The claim is that every timed automaton is bisimilar to a finite state system! This will be proved in Lecture 14.

References

- [1] R. Alur and D. Dill, "A theory of timed automata", *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

ee291E Lecture 14: Bisimulation of Timed Automata

Carlo Cloet & John Lygeros

March 10, 1999

1 Recall that ...

- Let $X = \{x_1, \dots, x_n\}$ be a finite collection of variables, each of which takes values in \mathbb{R} .
- Let $x = (x_1, \dots, x_n) \in \mathbb{R}^n = \mathbf{X}$ denote a valuation for all $x_i \in X$.

Definition 1 (Clock Constraints) *The set, $\Phi(X)$, of clock constraints of X , is a set of logical expressions defined inductively by $\delta \in \Phi(X)$ if:*

$$\delta := (x_i \leq c) \mid (x_i \geq c) \mid \neg \delta \mid \delta_1 \wedge \delta_2$$

where $x_i \in X$ and $c \geq 0$ is a rational number.

- To each $\delta \in \Phi(X)$ we can associate a set:

$$\hat{\delta} = \{x \in \mathbf{X} : \delta(x) = \text{True}\}$$

- The original definition of a timed automaton, found in [1], is given below.

Definition 2 (Timed Automaton) *A timed automaton is a hybrid automaton $H = (Q, X, \text{Init}, f, I, E, G, R)$, where*

- Q is a set of discrete variables, $Q = \{q_1, \dots, q_m\}$;
- $X = \{x_1, \dots, x_n\}$, $\mathbf{X} = \mathbb{R}^n$;
- $\text{Init} = \{\{q_i\} \times \widehat{\text{Init}_{q_i}}\}_{i=1}^m$ where $\text{Init}_{q_i} \in \Phi(X)$ with $x_i \geq 0$;
- $f(q, x) = (1, \dots, 1)$ for all (q, x) ;
- $\text{Inv}(q) = \mathbf{X}$ for all $q \in Q$;
- $E \subset Q \times Q$;
- $G(e) = \widehat{G_e}$ where $G_e \in \Phi(X)$, for all $e = (q, q') \in E$; and
- For all e , $R(e, x)$ either leaves x_i unaffected or resets it to 0.
- Notice that R is single valued. Also, x_i will always be ≥ 0 from the above definitions.

Example: As an example consider the timed automaton of Figure 1.

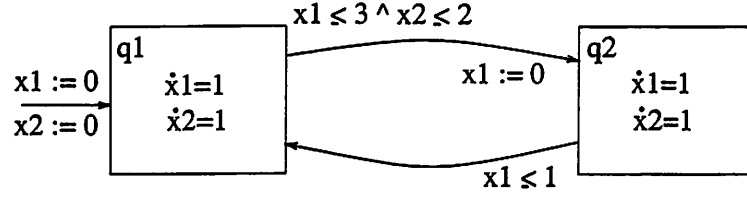


Figure 1: Example of a timed automaton

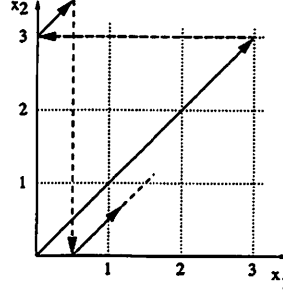


Figure 2: Example start of an execution of the timed automaton

- $Q = \{q\}$, $\mathbf{Q} = \{q_1, q_2\}$;
- $X = \{x_1, x_2\}$, $\mathbf{X} = \mathbb{R}^2$;
- $\text{Init} = \{(q_1, 0, 0)\}$;
- $f(q, x) = (1, 1)$ for all (q, x) ;
- $\text{Inv}(q) = \mathbb{R}^2$ for all $q \in \mathbf{Q}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 : (x_1 \geq 3) \wedge (x_2 \geq 2)\}$, $G(q_2, q_1) = \{x \in \mathbb{R}^2 : (x_1 \leq 1)\}$;
- $R(q_1, q_2, x) = \{(0, x_2)\}$, $R(q_2, q_1, x) = \{(x_1, 0)\}$

One possible start of an execution is shown in figure 2. Note that the system is non-deterministic.

1.1 Timed Automata and Transition Systems

- Consider a set of final states of the form $F = \{\{q_i\} \times \widehat{F_{q_i}}\}_{i=1}^m$ where $F_{q_i} \in \Phi(X)$.
- A timed automaton together with a set of final states, F , can be viewed as a transition system, $T = (S, \Sigma, \rightarrow, S_0, S_F)$, with:
 - $S = \mathbf{Q} \times \mathbf{X}$;
 - $\Sigma = E \cup \{\tau\}$, where τ is a symbol denoting time passage;
 - $((q, x), e, (q', x')) \in \rightarrow$ if $e = (q, q') \in E$, $G_e(x) = \text{True}$ and $x' \in R(e, x)$.
 - $((q, x), \tau, (q', x')) \in \rightarrow$ if $q = q'$, and there exists $t \geq 0$ such that $x' = x + t(1, \dots, 1)$.
 - $S_0 = \text{Init}$; and,
 - $S_F = F$.

2 Timed Automata are Bisimilar to Finite Systems

- First, we will show that without loss of generality, all constants can be assumed to be integers.
- Let T be the transition system defined by a timed automaton, H .
- Consider an arbitrary $\lambda > 0$, rational.
- Let H_λ denote the timed automaton obtained by replacing all constants, c , in H by λc .
- Let T_λ denote the transition system associated with H_λ .

Proposition 1 T and T_λ are bisimilar.

Proof: Consider the relation $(q, x) \sim (q, \lambda x)$, with $\sim \subseteq S \times S_\lambda$. Note that, since $\lambda > 0$, $(x_i \leq c) \Leftrightarrow (\lambda x_i \leq \lambda c)$ and $(x_i \geq c) \Leftrightarrow (\lambda x_i \geq \lambda c)$. Therefore:

$$(q, x) \in \text{Init} \Leftrightarrow (q, \lambda x) \in \text{Init}_\lambda \quad (1)$$

$$(q, x) \in F \Leftrightarrow (q, \lambda x) \in F_\lambda \quad (2)$$

$$(q, x) \xrightarrow{e} (q', x') \Leftrightarrow (q, \lambda x) \xrightarrow{e} (q', \lambda x') \quad (3)$$

$$(q, x) \xrightarrow{\tau} (q', x') \Leftrightarrow (q, \lambda x) \xrightarrow{\tau} (q', \lambda x') \quad (4)$$

For the discrete transition, $(q, x) \xrightarrow{e} (q', x')$ if $e = (q, q') \in E$, $Ge(x) = \text{True}$ and $x' \in R(e, x)$. Therefore, $(q, \lambda x) \xrightarrow{e} (q', \lambda x')$, since $e = (q, q') \in E$, $G_{e_\lambda}(\lambda x) = \text{True}$ and $\lambda x' \in R_\lambda(e, \lambda x)$. For the continuous transition, recall that $(q, x) \xrightarrow{\tau} (q', x')$ if $q = q'$, and there exists $t \geq 0$ such that $x' = x + t(1, \dots, 1)$. Therefore, $(q, \lambda x) \xrightarrow{\tau} (q', \lambda x')$ since $q = q'$ and $\exists t' \geq 0 : \lambda x' = \lambda x + t'(1, \dots, 1)$, where $t' = \lambda t$. ■

- We can therefore assume all constants are integers. If they are not, we let λ be a common multiple of their denominators and consider the bisimilar system T_λ . Recall that clock constraints require all constants to be rational.
- Let c_i denote the largest constant with which x_i is compared.
- In the above example, $c_1 = 3$ and $c_2 = 2$.
- Let $\lfloor x_i \rfloor$ denote the integer part of x_i and $\langle x_i \rangle$ denote the fractional part of x_i . In other words, $x_i = \lfloor x_i \rfloor + \langle x_i \rangle$, $\lfloor x_i \rfloor \in \mathbb{Z}$ and $\langle x_i \rangle \in [0, 1)$.
- Consider the relation $\sim \subseteq \mathbb{Q} \times \mathbb{X}$ with $(q, x) \sim (q', x')$ if:

1. $q = q'$;
2. for all x_i , $\lfloor x_i \rfloor = \lfloor x'_i \rfloor$ or $(x_i > c_i) \wedge (x'_i > c_i)$
3. for all x_i, x_j with $x_i \leq c_i$ and $x_j \leq c_j$

$$(\langle x_i \rangle \leq \langle x_j \rangle) \Leftrightarrow (\langle x'_i \rangle \leq \langle x'_j \rangle)$$

4. for all x_i with $x_i \leq c_i$,

$$(\langle x_i \rangle = 0) \Leftrightarrow (\langle x'_i \rangle = 0)$$

Proposition 2 \sim is an equivalence relation.

Proof: Probably part of homework 3! ■

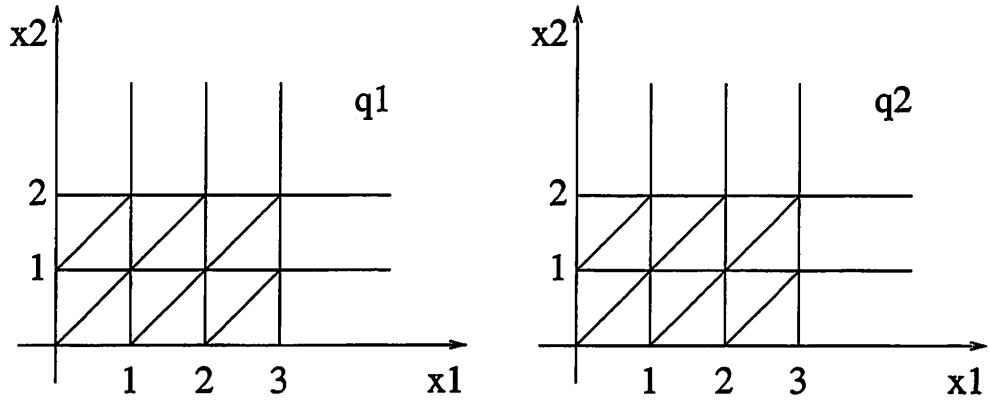


Figure 3: Equivalence classes for the example

- What do the equivalence classes look like?
- The equivalence classes are either open triangles, open line segments, open rectangles or points.
- For the example introduced above, they are shown in Figure 3.
- Notice that the number of classes is $2 \times (12\text{points} + 30\text{lines} + 18\text{open sets})$. Quite a few, but definitely finite!

Proposition 3 \sim is a bisimulation.

References

- [1] R. Alur and D. Dill, "A theory of timed automata", *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

ee291E Lecture 15: Bisimulation of Timed Automata (continued)

Jianghai Hu & John Lygeros

March 15, 1999

1 Last time ...

- Introduced timed automata.
- Showed that timed automata can be viewed as transition systems.
- Showed that without loss of generality all constants can be considered as integers.
- Introduced an equivalence relation, $\sim \subseteq \mathbf{Q} \times \mathbf{X} \times \mathbf{Q} \times \mathbf{X}$ with $(q, x) \sim (q', x')$ iff:

1. $q = q'$;
2. for all x_i , $\lfloor x_i \rfloor = \lfloor x'_i \rfloor$ or $(x_i > c_i) \wedge (x'_i > c_i)$
3. for all x_i, x_j with $x_i \leq c_i$ and $x_j \leq c_j$

$$(\langle x_i \rangle \leq \langle x_j \rangle) \Leftrightarrow (\langle x'_i \rangle \leq \langle x'_j \rangle)$$

4. for all x_i with $x_i \leq c_i$,

$$(\langle x_i \rangle = 0) \Leftrightarrow (\langle x'_i \rangle = 0)$$

- Here $\lfloor x_i \rfloor$ denotes the integer part and $\langle x_i \rangle$ the fractional part of x_i and c_i denotes the largest constant with which x_i is compared in the guards, initial and final states of the timed automaton.
- In \mathbb{R}^2 the equivalence classes of \sim are open triangles, open rectangles, open line segments (parallel to the axes and at 45 degrees), and points.
- For the timed automaton of Figure 1 the equivalence classes are shown in Figure 2.

2 Today ...

Proposition 1 \sim is a bisimulation.

Proof: We need to show:

1. Init is a union of equivalence classes.
2. F is a union of equivalence classes.
3. If P is an equivalence class and $e \in E$, $\text{Pre}_e(P)$ is a union of equivalence classes.

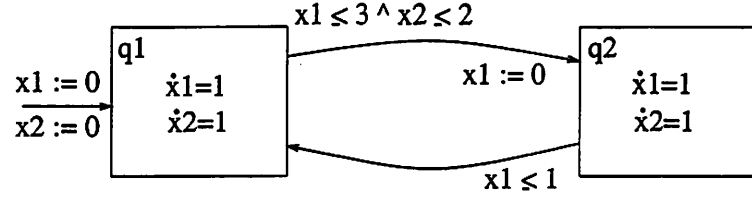


Figure 1: Example of a timed automaton

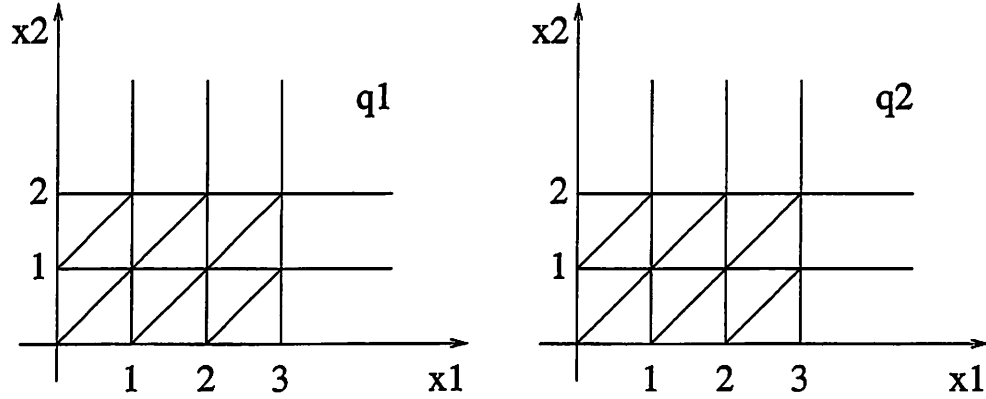


Figure 2: Equivalence classes for the example

- 4. If P is an equivalence class, $\text{Pre}_r(P)$ is a union of equivalence classes.

The proof will be somewhat informal. ■

- First, note that if $\delta \in \Phi(X)$,

$$\hat{\delta} = \{x \in X : \delta(x) = \text{True}\}$$

is “a union of equivalence classes” (for the X variables only). Recall that $\hat{\delta}$ can be written as the product of unions and intersections of sets of the form:

$$\{x_i \geq c\}, \{x_i \leq c\}, \{x_i < c\}, \{x_i > c\}, \{x_i = c\}$$

where c is an integer constant. All these sets are unions of equivalence classes for the X variables.

- This takes care of the requirements on Init and F .
- To deal with $\text{Pre}_e(P)$, let:

$$R^{-1}(e, P) = \{(q, x) \in Q \times X : \exists (q', x') \in P \text{ with } e = (q, q'), x' \in R(e, x)\}$$

- Notice that:

$$\text{Pre}_{(q, q')}(P) = R^{-1}(q, q', P) \cap (\{q\} \times G(q, q'))$$

Proposition 2 *If P is an equivalence class, $R^{-1}(e, P)$ is a union of equivalence classes. Hence $\text{Pre}_e(P)$ is a union of equivalence classes.*

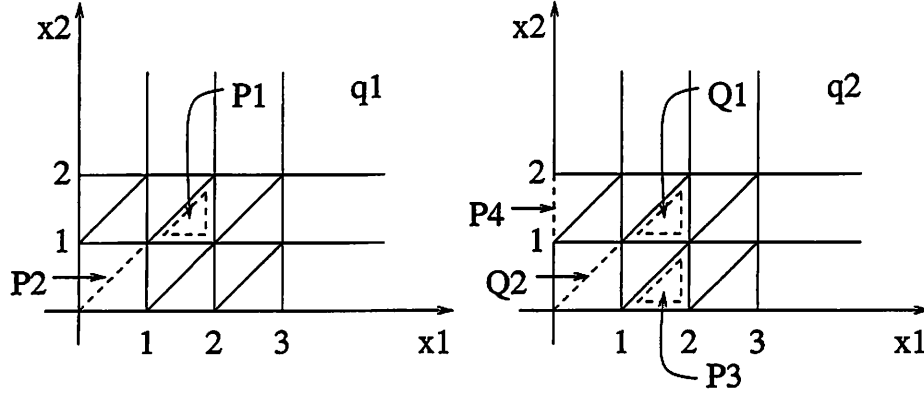


Figure 3: Examples of Pre_e computation

- Easier to demonstrate by examples. For the timed automaton of Figure 1, consider the equivalence classes P_1, \dots, P_4 shown in Figure 3. Denote $e_1 = (q_1, q_2), e_2 = (q_2, q_1)$. Notice that:

$$\begin{aligned}
 \text{Pre}_{e_1}(P_1) &= \emptyset, \\
 \text{Pre}_{e_2}(P_1) &= Q_1 \cap (\{q_2\} \times \{x_1 \leq 1\}) = \emptyset \\
 \text{Pre}_{e_1}(P_2) &= \emptyset, \\
 \text{Pre}_{e_2}(P_2) &= Q_2 \cap (\{q_2\} \times \{x_1 \leq 1\}) = Q_2 \\
 \text{Pre}_{e_1}(P_3) &= R^{-1}((q_1, q_2), P_3) \cap (\{q_1\} \times G(q_1, q_2)) = \emptyset \cap (\{q_1\} \times \{x_1 \leq 1 \wedge x_2 \leq 2\}) = \emptyset, \\
 \text{Pre}_{e_2}(P_3) &= \emptyset \\
 \text{Pre}_{e_1}(P_4) &= \{q_1\} \times (\{x_1 \geq 0 \wedge 1 < x_2 < 2\} \cap \{x_1 \leq 3 \wedge x_2 \leq 2\}) = \{q_1\} \times \{0 \leq x_1 \leq 3 \wedge 1 < x_2 < 2\}, \\
 \text{Pre}_{e_2}(P_4) &= \emptyset
 \end{aligned}$$

In all cases the result is a union of equivalence classes.

- Finally, notice that

$$\text{Pre}_r(P) = \{(q, x) \in Q \times X : \exists (q', x') \in P, t \geq 0 \text{ with } q = q', x' = x + t(1, \dots, 1)\}$$

These are all points that if we move in the $(1, \dots, 1)$ direction we will eventually reach P . If P is an equivalence class, this set is also a union of equivalence classes.

- For example, in Figure 4, $\text{Pre}_r(P) = P \cup P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5$

3 Complexity Estimates and Generalizations

- The above discussion indicates that reachability questions for timed automata can be answered on a finite state system, defined on the quotient space of the bisimulation \sim (also known as the *region graph*).
- What is the “size” of this finite state system?
- For the example, the number of equivalence classes is $2(12\text{points} + 30\text{lines} + 18\text{open sets}) = 120$. Quite a few!
- For an arbitrary system, one can expect up to $m(n!)(2^n) \prod_{i=1}^n (2c_i + 2)$ discrete states.
- Of course, in general, one need not construct the entire region graph:

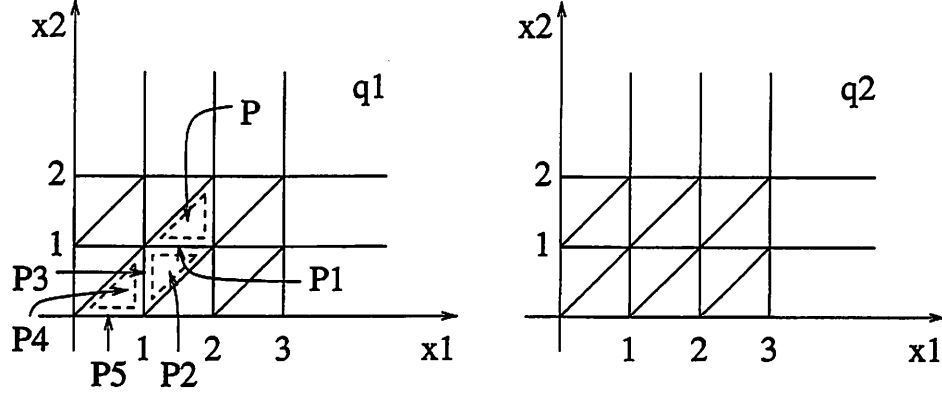


Figure 4: Examples of Pre^* computation

1. On the fly reachability: run the reachability algorithm (Lecture 12). Typically it will terminate, without constructing the entire region graph.
 2. Construct coarser bisimulations: run the bisimulation algorithm (Lecture 13). Typically the bisimulation generated will have fewer equivalence classes than the region graph.
- Still the problem is PSPACE complete!
 - The finite bisimulation result is preserved under some simple extensions:
 1. $I(q) = \hat{I}_q$ for some $I_q \in \Phi(X)$.
 2. $f(q, x) = (k_1, \dots, k_n)$ for some rational k_1, \dots, k_n and all (q, x) .
 3. R mapping equivalence classes to equivalence classes.

4 Rectangular Hybrid Automata

- The largest class of systems of this form that is known to be decidable is the class of *initialized rectangular automata*.
- A set $R \subset \mathbb{R}^n$ is called a rectangle if $R = \prod_{i=1}^n R_i$ where R_i are intervals whose finite end points are rational.

Definition 1 (Rectangular Automaton) A rectangular automaton is a hybrid automaton $H = (Q, X, \text{Init}, f, I, E, G, R)$, where

- Q is a set of discrete variables, $Q = \{q_1, \dots, q_m\}$;
- $X = \{x_1, \dots, x_n\}$, $\mathbf{X} = \mathbb{R}^n$;
- $\text{Init} = \cup_{i=1}^m \{q_i\} \times \text{Init}(q_i)$ where $\text{Init}(q_i)$ is a rectangle;
- $f(q, x) = F(q)$ for all (q, x) , where $F(q)$ is a rectangle;
- $I(q)$ is a rectangle for all $q \in Q$;
- $E \subset Q \times Q$;
- $G(e)$ is a rectangle for all $e = (q, q') \in E$; and
- For all e , $R(e, x)$ either leaves x_i unaffected or resets to an arbitrary value in a rectangle.

ee291E Lecture 16: Rectangular Automata

Arnab Nilim & John Lygeros

March 17, 1999

1 Initialized Rectangular Automata

- The largest class of systems of this form that is known to be decidable is the class of *initialized rectangular automata*.
- A set $R \subset \mathbb{R}^n$ is called a rectangle if $R = \prod_{i=1}^n R_i$ where R_i are intervals whose finite end points are rational.
- Examples: $R_1 = (1, \infty)$, $R_2 = [-3, 3/4]$, $R_3 = \{3\}$, etc.

Definition 1 (Rectangular Automaton) A rectangular automaton is a hybrid automaton $H = (Q, X, \text{Init}, f, I, E, G, R)$, where

- Q is a set of discrete variables, $Q = \{q_1, \dots, q_m\}$;
- $X = \{x_1, \dots, x_n\}$, $X = \mathbb{R}^n$;
- $\text{Init} = \cup_{i=1}^m \{q_i\} \times \text{Init}(q_i)$ where $\text{Init}(q_i) = \text{Init}_1(q_i) \times \dots \times \text{Init}_n(q_i)$ is a rectangle;
- $f(q, x) = F(q)$ for all (q, x) , where $F(q) = F_1(q) \times \dots \times F_n(q)$ is a rectangle;
- $I(q)$ is a rectangle for all $q \in Q$;
- $E \subseteq Q \times Q$;
- $G(e) = G_1(e) \times \dots \times G_n(e)$ is a rectangle for all $e = (q, q') \in E$; and
- For all e , $R(e, x) = R_1(e, x) \times \dots \times R_n(e, x)$ where:

$$R_i(e, x) = \begin{cases} \{x_i\} \\ \text{a fixed (independent of } x) \text{ interval} \end{cases} \quad \text{or}$$

Remarks:

- A *differential inclusion* is a generalization of a differential equation. Let $F : \mathbb{R}^n \rightarrow 2^{\mathbb{R}^n}$. An execution of the differential inclusion:

$$\dot{x} \in F(x), \quad x(0) = x_0 \in \mathbb{R}^n$$

on $[0, T] \subset \mathbb{R}_+$ is a differentiable function $x : [0, T] \rightarrow \mathbb{R}^n$ with $x(0) = x_0$ such that for all $t \in [0, T]$:

$$\dot{x}(t) \in F(x(t))$$

Notice that differential inclusions are “non-deterministic”, in the sense that many executions may exist for a single initial condition.

- Differential inclusions are commonly used as:

1. Abstractions of differential equations, such as $\dot{x}(t) = f(x) \in F(x)$, for example for reachability or viability computations [1, 2]. In fact one can show that the reachable set of a Lipschitz differential equation over a finite time horizon can be approximated arbitrarily closely by the reach set of a rectangular automaton [3].
2. Abstractions of control systems [1]:

$$\dot{x}(t) = f(x(t), v(t)), \text{ with } v(t) \in V(x(t)) \subseteq \mathbb{R}^m$$

is equivalent to:

$$\dot{x} = F(x) = \cup_{v \in V(x)} \{f(x, v)\}$$

In particular, a rectangular differential inclusion:

$$\dot{x} \in F_1(q) \times \dots \times F_n(q) = [\underline{F}_1(q), \overline{F}_1(q)] \times \dots \times [\underline{F}_n(q), \overline{F}_n(q)]$$

can be thought of as a trivial control system with n inputs:

$$\dot{x} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}, \quad v_i \in [\underline{F}_i(q), \overline{F}_i(q)]$$

- There is *no coupling* between the continuous variables: rectangles restrict each variable to an interval which is independent of the values of the other variables.
- For timed automata, if coupling is introduced by allowing comparisons of the form $x_i - x_j \leq c$, for some rational c in $\Phi(X)$, the finite bisimulation property is preserved. This is not the case for rectangular hybrid automata.
- $\text{Init}(q)$, $F(q)$, $I(q)$, $G(e)$ are convex subsets of \mathbb{R}^n .
- We will restrict attention to compact rectangles and trivial invariants ($I(q) = X$). Discussion holds in general, but technical points are much more subtle.

Rectangular automata can also be thought of as transition systems. Given a rectangular automaton and a set of final states of the form:

$$\text{Final} = \cup_{i=1}^m \{q_i\} \times \text{Final}(q_i)$$

where for all i , $\text{Final}(q_i)$ is a rectangle, consider the transition system $T = (S, \Sigma, \rightarrow, S_0, S_F)$, with:

- $S = Q \times X$;
- $\Sigma = E \cup \{\tau\}$, where τ is a symbol denoting time passage;
- $((q, x), e, (q', x')) \in \rightarrow$ if $e = (q, q') \in E$, $x \in G(e)$ and $x' \in R(e, x)$.
- $((q, x), \tau, (q', x')) \in \rightarrow$ if $q = q'$, and either $x = x'$ or $x \in I(q)$, $x' \in I(q)$ and there exists $t > 0$ such that $\frac{x' - x}{t} \in F(q)$.
- $S_0 = \text{Init}$; and,
- $S_F = \text{Final}$.

Definition 2 (Initialized Rectangular Automaton) A rectangular automaton is called *initialized* if for all transitions $e = (q, q') \in E$:

$$F_i(q) \neq F_i(q') \Rightarrow R_i(e, x) \neq \{x_i\}$$

2 Decidability Results

Theorem 1 *The reachability problem for initialized rectangular automata is complete for PSPACE.*

Implications:

- Good: Reachability is decidable, i.e. there exist algorithms that can compute the set of reachable states with a finite computation.
- Bad: The computation scales very badly.

It is noteworthy that $F(q) \neq F(q') \Rightarrow R_i(q, q', x) = [l, u]$ but not $R_i(q, q', x) = \{x_i\}$.

Proof: (Outline. For details see [4]. More digestible versions of the proof in [5, 6]). Introduce two new classes of hybrid automata:

1. Singular automata: rectangular hybrid automata where $F(q)$ is a singleton for all q .
2. Stop watch automata: singular automata with $F_i(q) \in \{0, 1\}$ for all q and i .

Consider two additional types of relations between transition systems:

1. Forward simulation: a transition system T forward simulates a transition system T' if there exists a relation $r \subseteq S \times S'$ such that:
 - (a) if $s' \in S'_0$ then there exists $s \in S_0$ such that $(s, s') \in r$.
 - (b) for all $\sigma \in \Sigma$, $s'_1, s'_2 \in S'$ and $s_1 \in S$ such that $(s_1, s'_1) \in r$, if $s'_1 \xrightarrow{\sigma} s'_2$ then there exists $s_2 \in S$ such that $(s_2, s'_2) \in r$ and $s_1 \xrightarrow{\sigma} s_2$.
2. Backward simulation: a transition system T backward simulates a transition system T' if there exists a relation $r \subseteq S \times S'$ such that:
 - (a) if $s' \in \text{Reach}(T')$ then there exists $s \in \text{Reach}(T)$ such that $(s, s') \in r$.
 - (b) if $s' \in S'_0$, $s \in \text{Reach}(T)$ and $(s, s') \in r$, then $s \in S_0$.
 - (c) for all $\sigma \in \Sigma$, $s'_1, s'_2 \in \text{Reach}(T')$ and $s_2 \in \text{Reach}(T)$ such that $(s_2, s'_2) \in r$, if $s'_1 \xrightarrow{\sigma} s'_2$ then there exists $s_1 \in \text{Reach}(T)$ such that $(s_1, s'_1) \in r$ and $s_1 \xrightarrow{\sigma} s_2$.

If T forward simulates T' then every sequence accepted by T' is also accepted by T . The same is true for backward simulation. Bisimulation is strictly stronger than forward and backward simulation: it is a forward simulation of T' by T whose inverse relation is a forward simulation of T by T' .

Build up the proof out of a sequence of Lemmas:

Lemma 1 *For every initialized rectangular automaton H there exists an initialized singular automaton S_H such that S_H forward simulates H and H backward simulates S_H .*

The construction doubles the number of continuous variables (for the compact case), but maintains the number of discrete states. For each x_i introduce two variables l_i and u_i to keep track of lower and upper bounds on x_i (Figure ??). For all q :

$$l_i = \underline{F}_i(q), \quad u_i = \overline{F}_i(q)$$

Each guard $G(e) = [\underline{G}_1(e), \overline{G}_1(e)] \times \dots \times [\underline{G}_n(e), \overline{G}_n(e)]$ is replaced by:

$$l_i \leq \overline{G}_i(e) \wedge u_i \geq \underline{G}_i(e)$$

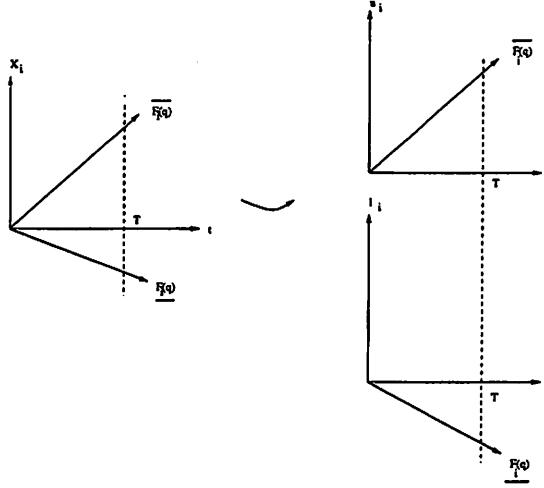


Figure 1: Replacing the differential inclusion by its upper and lower bound

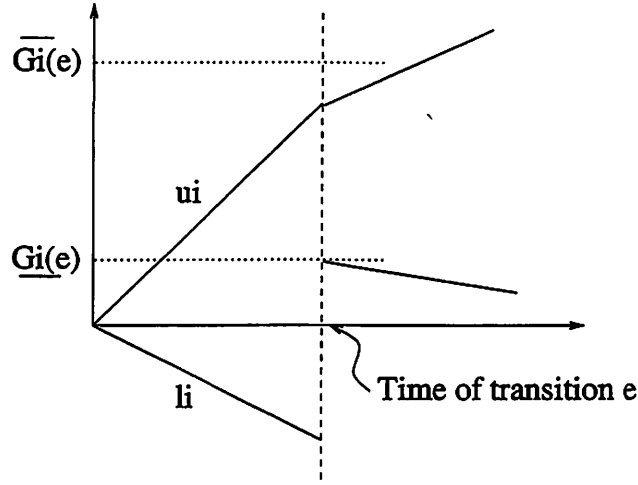


Figure 2: Shrinking the lower and upper bounds when a transition is taken

For the reset relations, if $R_i(e, x) = [\underline{R}_i(e), \overline{R}_i(e)]$ the new reset relation becomes:

$$l_i = \underline{R}_i(e), \quad u_i = \overline{R}_i(e)$$

If $R_i(e, x) = \{x_i\}$, the new reset relation becomes:

$$(l_i \leq \underline{G}_i(e) \Rightarrow l_i := \underline{G}_i(e)) \wedge (u_i \geq \overline{G}_i(e) \Rightarrow u_i := \overline{G}_i(e))$$

(Figure ??) otherwise, u_i or l_i remain constant.

Lemma 2 *For every singular automaton S , there exists a bisimilar stopwatch automaton W_S .*

Scale the continuous variables.

Lemma 3 *For every stopwatch automaton W , there exists a bisimilar timed automaton T_W .*

Add new discrete states to keep track of the value of variables whose rates become 0.

Overall, for every hybrid automaton H there exists a timed automaton $T_{W_{S_H}}$ that forward simulates H and is backward simulated by H (hence accepts the same sequences as H). ■

3 Undecidability Results

The class of initialized rectangular automata is, in a sense, at the boundary of the class of hybrid systems for which reachability is decidable. The problem becomes undecidable if:

1. we allow comparisons between X_i with different rates,
2. we allow non-initialized variables,
3. we allow assignment or continuous variables $X_i = X_j$,

The rest of the discussion elaborates somewhat on these points. For details and proofs please refer to [4].

A variable x_i is said to have slopes $\{k_1, \dots, k_l\}$ if $F_i(q) \in \{[k_1, k_1], \dots, [k_l, k_l]\}$. A rectangular automaton is simple if:

1. All but one x_i have slope 1.
2. $\text{Init} = \{q_i\} \times (0, \dots, 0)$ for some q_i .
3. If $R_i(e, x)$ is an interval it is of the form $[0, 0]$.
4. $I(q)$ is compact and constant.
5. For all q , $F(q)$ is compact.
6. For all e , $G(e)$ is compact.

Theorem 2 *For every two rational k_1 and k_2 , the reachability problem is undecidable for simple rectangular automata with a variable with slopes k_1 and k_2 .*

An automaton with assignments allows the value of one continuous variable to be assigned to another during a transition.

Theorem 3 *For every rational $k \neq 1$, the reachability problem is undecidable for simple rectangular automata with assignments and a variable with slope k .*

An automaton with triangular guards, invariants and resets allows the values of two continuous variables to be compared in the guards, invariants and resets respectively.

Theorem 4 *For every rational $k \neq 1$, the reachability problem is undecidable for simple rectangular automata with triangular guards, invariants or resets and a variable with slope k .*

An automaton with triangular flow allows the slopes of continuous variables to be compared in $F(q)$. An automaton is called three-simple if all but 3 of the x_i have slope 1 and it satisfies all the remaining constraints of a simple automaton.

Theorem 5 *The reachability problem is undecidable for three-simple automata with constant triangular flow.*

References

- [1] Jean-Pierre Aubin, *Viability Theory*, Birkhäuser, Boston, 1991.
- [2] M.R. Greenstreet and I. Mitchell, "Integrating projections", in *Hybrid Systems: Computation and Control*, S. Sastry and T.A. Henzinger, Eds., number 1386 in LNCS, pp. 159–174. Springer Verlag, 1998.

- [3] A. Puri, P. Varaiya, and V. Borkar, “ ϵ -approximation of differential inclusions”, in *IEEE Conference on Decision and Control*, New Orleans, LA, 1995, pp. 2892–2897.
- [4] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya, “What’s decidable about hybrid automata”, in *27th Annual Symposium on the Theory of Computing, STOC’95*. 1995, pp. 373–382, ACM Press.
- [5] Anuj Puri, *Theory of Hybrid Systems and Discrete Event Systems*, PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1995.
- [6] Anuj Puri and Pravin Varaiya, “Decidability of hybrid systems with rectangular differential inclusions”, in *Computer Aided Verification*, 1994, pp. 95–104.

ee291E Lecture 20: Controller Synthesis: Introduction

John Lygeros

April 9, 1999

1 The main idea

- Steer an automaton using input variables, so that the closed loop system exhibits certain desirable characteristics.
- Characteristics of interest:
 1. Stability
 2. Optimality
 3. Sequence properties
- Here we deal primarily with sequence properties, and in particular safety properties.
- Need inputs to define control problems. Return to definition of *open hybrid automata* and introduce some minor modifications.
 - Assume that the full state is measurable. To simplify the notation drop the output variables and the output map h and treat state variables as outputs.
 - Introduce state dependent input constraints.

2 The plant

The plant is modeled by a hybrid automaton of the form $H = (Q, X, V, \text{Init}, f, I, E, G, R, \phi)$, where

- Q is a finite collection of discrete state variables;
- X is a finite collection of continuous state variables;
- V is a finite collection of input variables. We assume $V = V_D \cup V_C$, where V_D contains discrete and V_C contains continuous variables.
- $\text{Init} \subseteq Q \times X$ is a set of initial states;
- $f : Q \times X \times V \rightarrow \mathbb{R}^n$ is an input dependent vector field;
- $I : Q \rightarrow 2^{X \times V}$ assigns to each $q \in Q$ an input dependent invariant set;
- $E \subseteq Q \times Q$ is a collection of discrete transitions;
- $G : E \rightarrow 2^{X \times V}$ assigns to each $e = (q, q') \in E$ a guard;

- $R : E \times X \times V \rightarrow 2^X$ assigns to each $e = (q, q') \in E$, $x \in X$ and $v \in V$ a reset relation; and,
- $\phi : Q \times X \rightarrow 2^V$ assigns to each state a set of admissible inputs.

Remarks:

1. Case where full state is not measurable is much more challenging.
2. Formally defining composition if the full state is not measurable and state dependent input constraints are allowed is also challenging.

The execution can be defined more or less as before: a collection $\chi = (\tau, q, x, v)$ with $\tau \in \mathcal{T} = \{[\tau_i, \tau'_i]\}_{i=1}^N$, $q : \tau \rightarrow Q$, $x : \tau \rightarrow X$, and $v : \tau \rightarrow V$ satisfying:

- Initial condition: $(q(\tau_0), x(\tau_0)) \in \text{Init}$;
- Continuous evolution: for all i with $\tau_i < \tau'_i$, q , x , v and y are continuous over $[\tau_i, \tau'_i]$ and for all $t \in [\tau_i, \tau'_i]$, $(x(t), v(t)) \in I(q(t))$ and $\frac{d}{dt}x(t) = f(q(t), x(t), v(t))$;
- Discrete Evolution: for all i , either $(q(\tau'_i), x(\tau'_i)) = (q(\tau_{i+1}), x(\tau_{i+1}))$, or $e_i = (q(\tau'_i), q(\tau_{i+1})) \in E$, $(x(\tau'_i), v(\tau'_i)) \in G(e_i)$, and $x(\tau_{i+1}) \in R(e_i, x(\tau'_i), v(\tau'_i))$; and,
- Input Evolution: for all $t \in \tau$, $v(t) = \phi(q(t), x(t))$.

Remarks:

1. To prevent technical problems we assume that $\phi(q, x) \neq \emptyset$ and f is Lipschitz in x and continuous in v . Notice that if $\phi(q, x) = \emptyset$ execution can not be defined beyond (q, x) .
2. We denote by \mathcal{H} the set of all executions of H , by \mathcal{H}^* the set of all finite executions of H and by \mathcal{X}^* the set of all finite execution, projected only on the state variables.
3. Given $\chi = (\tau, q, x, v) \in \mathcal{H}$ and $t \in \tau$ we use $\chi \downarrow_t$ to denote the finite prefix of χ ending at t .

3 Specifications

- The control objective is encoded by means of a property of the system. Here we restrict our attention to safety properties.
- Recall that the property $(Q \cup X, \Box F)$ (always F) with $F \subseteq Q \times X$ is defined as:

$$\Box F(\chi) = \text{True iff } \forall t \in \tau, (q(t), x(t)) \in F$$

- Recall that this is a safety property.
- Given a plant hybrid automaton and a property of the form $(Q \cup X, \Box F)$, our goal is to choose the valuations of the input variables so that all executions of the system satisfy the property.

4 Controls and Disturbances & Controllers

- The inputs are chosen by a “controller”.
- Typically some input variables can be controlled (i.e. their valuations can be assigned at will by the controller), while others can not be controlled (their valuations are chosen by the “environment”).

- Uncontrollable variables (also known as *disturbances*) typically represent:
 1. Noise in the sensors, numerical computations, etc.
 2. External forces, wind for the aircraft, etc.
 3. Unmodeled dynamics
 4. Uncertainty about the actions of other agents (in examples such as air traffic control and highway systems).
- In the approach presented here the behavior of the uncontrolled input variables is assumed to be adversarial. We would like the controllable inputs to guarantee the specification despite the action of the disturbances.
- In some cases this approach is too conservative. Attempts to relax it (by introducing probabilities for example) are underway).
- More formally, consider a plant hybrid automaton H and a safety property $P = (Q \cup X, \Box F)$.
- Assume the input variables of H are partitioned into *controls*, U and *disturbances*, D :

$$V = U \cup D$$

- Define a controller as a map from state executions to sets of allowable inputs:

$$C : \mathcal{X}^* \rightarrow 2^U$$

- The interpretation is that given a finite execution, the controller restricts the valuations of the control input variables that are allowed at the final state.
- The set of *closed loop causal executions* is defined as:

$$\mathcal{H}_C = \{(\tau, q, x, (u, d)) \in \mathcal{H} \mid \forall t \in \tau, u(t) \in C((\tau, q, x) \downarrow_t)\}$$

- Clearly, $\mathcal{H}_C \subseteq \mathcal{H}$
- We say C satisfies property P if:

$$\Box F(\chi) = \text{True for all } \chi \in \mathcal{H}_C$$

ee291E Lecture 21: Controller Synthesis: Formulation

John Lygeros

April 12, 1999

1 Overview

A control problem involves:

1. A plant
2. A specification
3. A controller

Controller synthesis involves coming up with a methodology for designing controllers that meet the specification. The discussion in this lecture is based on [1].

Recall that the *plant* is modeled by an open hybrid automaton, $H = (Q, X, V, \text{Init}, f, I, E, G, R, \phi)$, where

- Q is a finite collection of discrete state variables;
- X is a finite collection of continuous state variables;
- V is a finite collection of input variables. We assume $V = V_D \cup V_C$, where V_D contains discrete and V_C contains continuous variables.
- $\text{Init} \subseteq Q \times X$ is a set of initial states;
- $f : Q \times X \times V \rightarrow \mathbb{R}^n$ is an input dependent vector field;
- $I : Q \rightarrow 2^{X \times V}$ assigns to each $q \in Q$ an input dependent invariant set;
- $E \subseteq Q \times Q$ is a collection of discrete transitions;
- $G : E \rightarrow 2^{X \times V}$ assigns to each $e = (q, q') \in E$ a guard;
- $R : E \times X \times V \rightarrow 2^X$ assigns to each $e = (q, q') \in E$, $x \in X$ and $v \in V$ a reset relation; and,
- $\phi : Q \times X \rightarrow 2^V$ assigns to each state a set of admissible inputs.

To avoid technical difficulties we introduce the additional assumption that $\phi(q, x) \neq \emptyset$ and f is Lipschitz in x and continuous in v .

Also recall that the control objective is assumed to be encoded by means of a safety property $(Q \cup X, \Box F)$ (always F) with $F \subseteq Q \times X$.

Finally recall that we assume the input variables of H are partitioned into *controls*, U and *disturbances*, D :

$$V = U \cup D$$

The disturbances represent uncontrolled inputs such as noise, unmodeled dynamics, the actions of other subsystems (in a distributed system), etc.

2 Controllers

A controller can be defined as a map from state executions to sets of allowable inputs:

$$C : \mathcal{X}^* \rightarrow 2^U$$

The interpretation is that given a finite execution, the controller restricts the valuations of the control input variables that are allowed at the final state. With this in mind we can define the set of *closed loop causal executions* as:

$$\mathcal{H}_C = \{(\tau, q, x, (u, d)) \in \mathcal{H} \mid \forall t \in \tau, u(t) \in C((\tau, q, x) \downarrow_t)\}$$

Clearly, $\mathcal{H}_C \subseteq \mathcal{H}$. We say C satisfies property $(Q \cup X, \Box F)$ if:

$$\Box F(\chi) = \text{True for all } \chi \in \mathcal{H}_C$$

To prevent technical problems, assume that for all $(\{\tau_i, \tau'_i\}_{i=0}^N, q, x, v) \in \mathcal{H}$, $\emptyset \neq C(x) \subseteq \phi(q(\tau'_N), x(\tau'_N))|_U$. This ensures that the controller will not attempt to “cheat” by stalling the execution. This is not enough however. The controller may still be able to cheat by:

1. Blocking the execution at a state (q, x) by applying $u \in \phi(q, x)|_U$ such that for all $d \in \phi(q, x)|_D$ $(x, (u, d)) \notin I(q)$ and for all $e \in E$ either $(x, (u, d)) \notin G(e)$ or $R(e, x, (u, d)) = \emptyset$.
2. Forcing the execution to be Zeno (take infinite number of transitions in a finite amount of time). Recall the water tanks example.

Both of these caveats have to do with modeling over-abstraction: the model of the plant should be such that the controller is not able to cheat in this way. The first loophole can be eliminated by fairly simple assumptions on the plant, similar to the non-blocking assumptions for autonomous hybrid automata. The second loophole is more difficult to deal with. Typically one assumes that all loops among discrete states require a non zero amount of time. This assumption may be somewhat restrictive and is difficult to enforce by manipulating the primitives of the model. Here we will overlook these technical problems.

3 Basic Controller Properties

Given a plant hybrid automaton and a property our goal is to find a controller that satisfies the property.

Memoryless Controllers

A controller, C , is called *memoryless* (or sometimes pure feedback) if for all $\chi, \chi' \in \mathcal{H}^*$ ending at the same state, $C(\chi) = C(\chi')$. A memoryless controllers can be characterized by a feedback map:

$$g : \mathbf{Q} \times \mathbf{X} \rightarrow 2^U$$

To prevent technical problems we again restrict our attention to memoryless controllers such that for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$ $\emptyset \neq g(q, x) \subseteq \phi(q, x)$. Given a plant, H , and a memoryless controller, g , we can define the closed loop open hybrid automaton, $H_g = (Q, X, V, \text{Init}, f, I, E, G, R, \phi_g)$, where $\phi_g(q, x) = \phi(q, x) \cap g(q, x)$. It is easy to show that:

Proposition 1 *If C is memoryless controller with feedback map g , then $\mathcal{H}_g = \mathcal{H}_C$.*

This property allows one to nest controller synthesis problems, provided they can be solved by memoryless controllers. In general, is unclear whether the set of closed loop causal executions is the set of executions of some hybrid automaton.

For properties of the form $(Q \cup X, \Box F)$, it turns out that it suffices to look for a solution among memoryless controllers.

Proposition 2 *A controller that satisfies property $(Q \cup X, \Box F)$ exists if and only if a memoryless controller that satisfies $(Q \cup X, \Box F)$ exists.*

Proof: The if part is obvious. For the only if part, assume that there exists a controller C that solves the synthesis problem $(H, \Box F)$, but there does not exist a feedback controller that solves the synthesis problem. Therefore, there must exist $(q, x) \in F$ and two different finite executions $\chi_1 = (\tau_1, q_1, x_1, (u_1, d_1)) \in \mathcal{H}_C$ and $\chi_2 = (\tau_2, q_2, x_2, (u_2, d_2)) \in \mathcal{H}_C$ ending in (q, x) such that $C(q_1, x_1) \neq C(q_2, x_2)$. Moreover, the “information” about whether (q, x) was reached via χ_1 or whether it was reached via χ_2 must be essential for subsequent control decisions.

More formally, assume (q, x) is reached via χ_2 , and let χ' denote a subsequent execution, that is assume that the concatenation $\chi_2\chi'$ belongs to \mathcal{H} . Note that, since χ_1 also ends in (q, x) , $\chi_1\chi'$ also belongs to \mathcal{H} . Let $\chi_2\chi' = (\tau'_2, q'_2, x'_2, (u'_2, d'_2))$ and $\chi_1\chi' = (\tau'_1, q'_1, x'_1, (u'_1, d'_1))$. Assume that for all $t \in \tau'_2 \setminus \tau_2$, a control $u(t) \in C((q'_1, x'_1) \downarrow_t)$ is applied (instead of a control $u(t) \in C((q'_2, x'_2) \downarrow_t)$). Then, as the fact that (q, x) was reached via χ_2 is essential, there must exist a subsequent execution χ' such that $\chi_2\chi' \in \mathcal{H}$ (in fact $\chi_2\chi' \in \mathcal{H} \setminus \mathcal{H}_C$) and $\Box F(\chi_2\chi') = \text{False}$. This implies that there exists $t \in \tau'_2$ such that $(q'_2(t), x'_2(t)) \in F^c$. Since C is assumed to solve the synthesis problem and $\chi_2 \in \mathcal{H}_C$, $\Box F(\chi_2) = \text{True}$, therefore $t \in \tau'_2 \setminus \tau_2$.

However, since for all $t \in \tau'_2 \setminus \tau_2$, $u(t) \in C((q'_1, x'_1) \downarrow_t)$, and $(\tau'_1, q'_1, x'_1, (u'_1, d'_1)) \in \mathcal{H}$, we have that $\chi_1\chi' \in \mathcal{H}_C$. But the above discussion indicates that there exists $t \in \tau'_1$ (in fact $t \in \tau'_1 \setminus \tau_1$) such that $(q'_1(t), x'_1(t)) \in F^c$. This contradicts the assumption that C solves the synthesis problem $(H, \Box F)$. ■

Motivated by Proposition 2, we restrict our attention to feedback controllers. For brevity, we refer to the problem of finding a controller for a plant H that satisfies a specification $(Q \cup X, \Box F)$ as the *controller synthesis problem* $(H, \Box F)$.

Controlled Invariant Sets

Typically, for a controller synthesis problem one treats the set of initial conditions, Init , as variable and attempts to establish the largest set of states for which there exists a controller that satisfies the specification. This set of initial conditions turns out to be a “controlled invariant set”.

Definition 1 (Controlled Invariant) *A set $W \subseteq \mathbf{Q} \times \mathbf{X}$ is called controlled invariant if there exists a controller that solves the controller synthesis problem $(H', \Box W)$ when H' is identical to H except for Init' which is equal to W .*

A controlled invariant set W is called *maximal* if it is not a proper subset of another controlled invariant set. We say a controller *renders W invariant* if it solves the controller synthesis problem $(H', \Box W)$ where $\text{Init}' = W$.

Proposition 3 *A controller that solves the synthesis problem $(H, \Box F)$ exists if and only if there exists a unique maximal controlled invariant $W \subseteq \mathbf{Q} \times \mathbf{X}$ such that $\text{Init} \subseteq W \subseteq F$.*

Proof: If there exists any control invariant $W \subseteq F$ (in particular, if there exists a unique maximal one) then, by definition, the synthesis problem $(H, \Box F)$ can be solved for $I = W$.

For the only if part, if the synthesis problem can be solved for some Init , there exists a set $\widehat{\text{Init}}$ and a feedback controller g such that for all d and for all $(q_0, x_0) \in \widehat{\text{Init}}$ the execution $(\tau, q, x, (u, d))$ with $u(t) \in g(q(t), x(t))$ for all $t \in \tau$ satisfies $(q(t), x(t)) \in F$ for all $t \in \tau$. Consider the set:

$$W = \bigcup_d \bigcup_{(q_0, x_0) \in \widehat{\text{Init}}} \bigcup_{t \in \tau} (q(t), x(t))$$

Then clearly $W \subseteq F$. Moreover, for any $(q_0, x_0) \in W$ consider the execution $(\tau, q, x, (u, d))$ with arbitrary $d \in \mathcal{D}$ and $u(t) \in g(q(t), x(t))$. Then, by definition of W , $(q(t), x(t)) \in W$ for all $t \in \tau$. Therefore, controller g renders the set W invariant.

Having established the existence of controlled invariant subsets of F , consider now two such sets $W_1 \subseteq F$ and $W_2 \subseteq F$. We show that their union is also a controlled invariant subset of F . Clearly $W_1 \cup W_2 \subseteq F$. For $i = 1, 2$, as W_i is controlled invariant, there exists a feedback controller g_i that solves the controller synthesis problem $(H, \Box W_i)$, with $\text{Init} = W_i$. Consider the feedback controller g with:

$$g(q, x) = \begin{cases} g_1(q, x) & \text{if } (q, x) \in W_1 \\ g_2(q, x) & \text{otherwise} \end{cases}$$

Consider an arbitrary $(q_0, x_0) \in W_1 \cup W_2$. Then either $(q_0, x_0) \in W_1$ or $(q_0, x_0) \in (W_1 \cup W_2) \setminus W_1 \subseteq W_2$. In the first case, all executions are guaranteed to satisfy $\Box W_1$ as g_1 renders W_1 invariant. For the second case, consider an arbitrary execution $\chi = (\tau, q, x, (u, d))$ with $u(t) \in g(q(t), x(t))$ for all $t \in \tau$. Since g_2 solves the controller synthesis problem $(H, \Box W_2)$ with $\text{Init} = W_2$, either $\Box(W_2 \setminus W_1)(\chi) = \text{True}$ or $(q, x) \in W_2 \setminus W_1$ until $(q, x) \in W_1$, which brings us back to the first case. Hence, g solves the controller synthesis problem $(H, \Box(W_1 \cup W_2))$ with $\text{Init} = W_1 \cup W_2$, and the set $W_1 \cup W_2$ is controlled invariant.

Summarizing, the class of controlled invariant subsets of F is closed under union. Hence, it possesses a unique maximal element. ■

Least Restrictive Controllers

We would like to derive a memoryless controller that solves the problem while imposing minimal restrictions on the controls it allows. There are at least two reasons why such a controller is desirable:

1. As discussed above, safety properties can sometimes be satisfied using trivial controllers (that cause deadlocks or zeno executions for example). Imposing as few restrictions as possible allows us to find a meaningful controller whenever possible.
2. In many cases multiple, prioritized specifications are given for a particular problem. Imposing fewer restrictions on the controls when designing controllers for higher priority specifications allows us greater flexibility when trying to satisfy lower priority specifications.

Memoryless controllers that solve the synthesis problem $(H, \Box F)$ can be partially ordered by the relation:

$$g_1 \preceq g_2 \Leftrightarrow g_1(x) \subseteq g_2(x) \text{ for all } x \in X$$

Definition 2 *A memoryless controller that solves $(H, \Box F)$ is called least restrictive if it is maximal among the controllers that solve $(H, \Box F)$.*

There is a conjecture that for every controller synthesis problem $(H, \Box F)$ either there is no solution or there exists a unique least restrictive controller that solves the problem. As of now there is no proof of this fact however.

Some Remarks on “Implementation”

The notion of a controller introduced above may be inadequate when it comes to implementation. For one thing, the set valued map g allows non-deterministic choices of control inputs. Since in practice only one input can be applied to the system at any time, this nondeterminism has to somehow be resolved when it comes time to implement such a controller. The set valued map can in this sense be thought of as a family of single valued controllers; implementation involves choosing one controller from this family.

Normally, one would “implement” a controller by another hybrid automaton, which, when composed with the plant automaton yields the desired behavior. To do this one would need to introduce output variables to the hybrid automaton and define formal semantics for composition, as in Lecture 8. The process is slightly more

complicated for the models considered here because of the presence of the state dependent input constraints, encoded by ϕ .

We assume that the entire state is available to the controller. In general this will not be the case. If a controller is to be implemented by a hybrid automaton, the information the controller has about the plant is obtained through the valuations of the output variables of the plant, which are not necessarily in one to one correspondence with the valuations of the state variables. The controller synthesis problem under partial observation (output feedback) is much more complicated than the full observation (state feedback) problem addressed here (partly because it makes it harder to define composition as discussed above).

References

- [1] John Lygeros, Claire Tomlin, and Shankar Sastry, "Controllers for reachability specifications for hybrid systems", *Automatica*, 1999, (accepted for publication, to appear in March 1999).

ee291E Lecture 22: Controller Synthesis: Game Theoretic Approach

Omid Shakernia John Lygeros

April 14, 1999

1 Game Theoretic Controller Synthesis

To guarantee that a safety specification is met despite the action of the disturbances we cast the design problem as a zero sum dynamic game. The two players in the game are the control u and the disturbance d and they compete over cost a function that encodes the safety specification. We seek the best possible control action and the worst possible disturbance. Note that if the specifications can be met for this pair then they can also be met for any other choice of the disturbance.

Consider a controller synthesis problem $(H, \Box F)$. The game can be cast in the standard min-max setting by introducing a cost function induced by the discrete metric. The discrete metric is simply a map $m : (\mathbf{Q} \times \mathbf{X}) \times (\mathbf{Q} \times \mathbf{X}) \rightarrow \mathbb{R}$ defined by:

$$m((q_1, x_1), (q_2, x_2)) = \begin{cases} 0 & \text{if } (q_1, x_1) = (q_2, x_2) \\ 1 & \text{if } (q_1, x_1) \neq (q_2, x_2) \end{cases}$$

It is easy to check that m satisfies the axioms of a metric. The metric induces a map on subsets of $\mathbf{Q} \times \mathbf{X}$ by defining:

$$\begin{aligned} M : 2^{\mathbf{Q} \times \mathbf{X}} \times 2^{\mathbf{Q} \times \mathbf{X}} &\rightarrow \mathbb{R} \\ (W_1, W_2) &\mapsto \min_{((q_1, x_1), (q_2, x_2)) \in W_1 \times W_2} m((q_1, x_1), (q_2, x_2)) \end{aligned}$$

In other words, $M(W_1, W_2) = 0$ if $W_1 \cap W_2 \neq \emptyset$ and $M(W_1, W_2) = 1$ if $W_1 \cap W_2 = \emptyset$.

Consider an execution, $\chi = (\tau, q, x, (u, d))$, of the hybrid automaton H starting at an initial state $(q_0, x_0) \in I$. Define the *cost* of this execution by:

$$\begin{aligned} J : \mathcal{H} &\rightarrow \mathbb{R} \\ \chi &\mapsto \min_{t \in \tau} M(\{(q(t), x(t))\}, F^c) \end{aligned}$$

Note that J can only take on two values, $J(\chi) \in \{0, 1\}$. Therefore, J implicitly defines a property of the hybrid automaton. In fact:

Proposition 1 $J(\chi) = 1$ if and only if $\Box F(\chi) = \text{True}$.

Intuitively, u tries to maximize the cost function J (prevent the state from leaving F). Because we have no control over the actions of d , we assume that it tries to minimize J (force the state to leave F). As we would like to establish conditions under which $\Box F$ is guaranteed to be satisfied, we bias the game in favor of the disturbance whenever there is ambiguity over how the game will proceed. For example, multiple executions may be possible for the same initial condition, control and disturbance trajectories, due to nondeterminism. Moreover, the order in which the two players play in the game may be important, if one player is assumed to

have access to the decision of the other player before choosing his/her action. In both these cases we would like to give the disturbance the “benefit of the doubt”.

Consider the max-min solution :

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau, q, x, (u, d)) \in \mathcal{H}_g} J(\chi) \right) \quad (1)$$

Motivated by Proposition 1 we restrict our attention to feedback strategies for u in equation (1). Following the standard game theoretic convention, the “player” who appears first in the right hand side of equation (1) (the controller g) is also assumed to play first. The player who appears second (the disturbance d) is assumed to have access to the strategy of the first player, when called upon to make his/her decision. The minimum over χ removes all nondeterminism. Therefore, provided a solution to this equation can be found, J^* is a well defined function of the initial state (q_0, x_0) . In addition, the minimum over χ implicitly restricts attention to control and disturbance trajectories that satisfy the state based input constraint ϕ .

Using J^* we define a set $W^* \subseteq \mathbf{Q} \times \mathbf{X}$ by:

$$W^* = \{(q_0, x_0) \in \mathbf{Q} \times \mathbf{X} \mid J^*(q_0, x_0) = 1\} \quad (2)$$

Proposition 2 W^* is the maximal controlled invariant subset of F .

Proof: We first show W^* is controlled invariant. Assume for the sake of contradiction that it is not. Then for all g there exists an $(q_0, x_0) \in W^*$, a d , a $\chi = (\tau, q, x, (u, d)) \in \mathcal{H}_g$ and a $t \in \tau$ with $(q(t), x(t)) \notin W^*$. By Proposition 1, $J(\chi) = 0$, which, by equation (1), implies that $J^*(q_0, x_0) = 0$. This contradicts the assumption that $(q_0, x_0) \in W^*$.

Next we show that W^* is maximal. Assume for the sake of contradiction that it is not, that is there exists another controlled invariant \hat{W} with $W^* \subset \hat{W} \subseteq F$. Then, by definition, there exists a controller g such that \mathcal{H}_g satisfies $\Box F$ with $I = \hat{W}$. In other words, for all $(q_0, x_0) \in \hat{W}$, for all d and for all $\chi = (\tau, q, x, (u, d)) \in \mathcal{H}_g$, $\Box F(\chi) = \text{True}$, or, equivalently, $J(\chi) = 1$. But, from equation (1) this would imply that $J^*(q_0, x_0) = 1$. This contradicts the assumption that $W^* \subset \hat{W}$. ■

If a solution to equation 1 can be computed, then there exists a feedback controller (namely one that achieves the maximum) that renders W^* invariant. We would like to find a least restrictive such controller. Our ability to solve the synthesis problem using this technique hinges on finding a solution to equation 1. In some cases this can be done by brute force; this typically involves guessing a controller and showing that it achieves the maximum. More systematically, this can be done using optimal control techniques, in particular dynamic programming.

2 Example: The Steam Boiler

Recall the steam boiler problem from Lecture 8 (Figure 1). The problem was introduced first in [1, 2]. The model presented here is taken from [3]. The controller synthesis for this model can be found in [4]. The continuous dynamics of the boiling process are summarized by the differential equations:

$$\begin{aligned} \dot{w} &= p_1 + p_2 - r \\ \dot{r} &= d \end{aligned}$$

The dynamics of pump i are summarized by the open hybrid automaton of Figure 2. Notice that p_i is both an output variable of the pump and an input variable of the boiling process. For a formal definition of the model (with slight differences in the notation) please refer to Lecture 8.

The composite automaton has 4 continuous, real valued state variables:

$$x = (w, r, T_1, T_2) \in \mathbb{R}^4$$

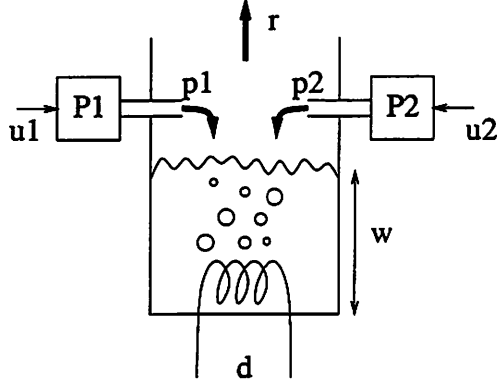


Figure 1: The Steam Boiler

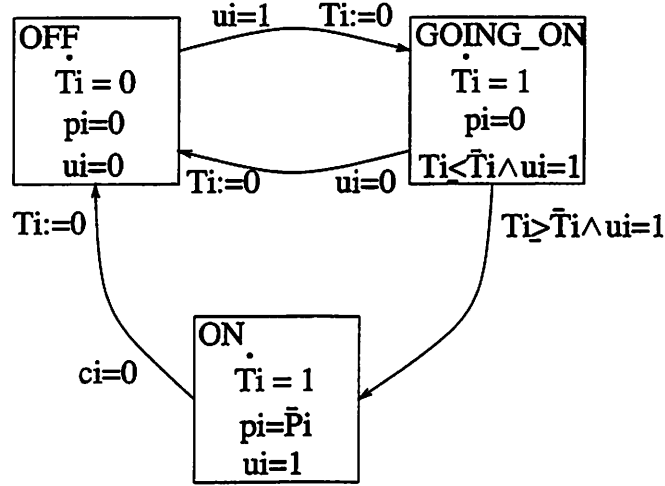


Figure 2: The pump hybrid automaton

9 discrete states:

$$q \in \{(OFF, OFF), (OFF, GOING_ON), \dots, (ON, ON)\}$$

2 discrete input variables:

$$(u_1, u_2) \in \{0, 1\} \times \{0, 1\} = U$$

and one continuous input variable:

$$d \in [-D_1, D_2] = D$$

As the notation suggests, u_1 and u_2 will play the role of controls and d will play the role of the disturbance. The additional requirement that $r \in [0, R]$ can be encoded by a state dependent input constraint:

$$\phi(q, x) = \begin{cases} U \times [0, D_2] & \text{if } r \leq 0 \\ U \times D & \text{if } r \in (0, R) \\ U \times [-D_1, 0] & \text{if } r \geq R \end{cases}$$

Proposition 3 *If $\text{Init} \subseteq Q \times \mathbb{R} \times [0, R] \times \mathbb{R}^2$, then for all $\chi = (\tau, q, x, u_1, u_2, d)$ and for all $t \in \tau$, $r(t) \in [0, R]$.*

Our goal is to design a controller that keeps the water level in a given range, $[M_1, M_2]$, with $0 \leq M_1 < M_2$. This requirement can easily be encoded by a safety property $(Q \cup X, \Box F)$ with

$$F = Q \times [M_1, M_2] \times \mathbb{R}^3$$

We will try to achieve this goal by treating the situation as a game between (u_1, u_2) and d over the cost function J . Recall that this involves solving the equation:

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau, q, x, (u, d)) \in \mathcal{H}_g} J(\chi) \right)$$

Fortunately, for this example, the equation simplifies considerably.

First, notice that the steam boiler system is deterministic, in the sense that for each initial state and each input sequence consistent with ϕ the automaton accepts a unique execution. In this case, we can represent an execution more compactly by $((q_0, x_0), (u_1, u_2), d)$ with the interpretation that (u_1, u_2) and d represent the entire sequence for these variables. Moreover, if the memoryless controller we pick is single valued, this implies we need not worry about the innermost minimization.

Next notice that J can be encoded by means of two real valued cost functions:

$$J_1(x^0, u_1, u_2, d) = \inf_{t \geq 0} w(t) \quad \text{and} \quad J_2(x^0, u_1, u_2, d) = -\sup_{t \geq 0} w(t) \quad (3)$$

Clearly:

$$J = 1 \Leftrightarrow (J_1 \geq M_1) \wedge (J_2 \geq -M_2)$$

The problem of finding a solution to the game over the discrete cost function (known as *qualitative game* or *game of kind*) reduces to finding solutions to two real valued games (known as *quantitative games* or *games of degree*). Even though there is no obvious benefit to doing this, it allows us to use tools from continuous optimal control to address the problem.

Start with the game over J_1 . Guess a possible solution:

$$u_i^*(q, x) = 1 \text{ for all } (q, x), \quad \text{and} \quad d^*(q, x) = \begin{cases} D_2 & \text{if } r < R \\ 0 & \text{if } r = R \end{cases} \quad (4)$$

Notice that both players resort to a feedback strategy (a trivial one).

Lemma 1 (u_1^*, u_2^*, d^*) is globally a saddle solution for the game between (u_1, u_2) and d over J_1 .

Proof: See [4]. ■

A *saddle solution* is a solution to equation (1) for which the order in which the players make their decisions turns out to be unimportant. In other words, a solution for which for all (q, x) :

$$J_1^*(q, x) = \max_{(u_1, u_2)} \min_d J_1((q, x), (u_1, u_2), d) = \min_d \max_{(u_1, u_2)} J_1((q, x), (u_1, u_2), d)$$

Or, in other words, a solution for which for all $(q, x), u_1, u_2$ and d :

$$J_1((q, x), (u_1, u_2), d^*) \leq J_1((q, x), (u_1^*, u_2^*), d^*) \leq J_1((q, x), (u_1^*, u_2^*), d)$$

The last definition is usually somewhat easier to work with. It is in fact used to prove the lemma.

The saddle cost:

$$J_1^*(q, x) = J_1((q, x), (u_1^*, u_2^*), d^*)$$

Can be computed in closed form. This allows us then to compute the set of states for which there exists a control that for all actions of the disturbance prevents draining. This set turns out to be of the form:

$$W_1^* = \{(q, x) : J_1^*(q, x) \geq M_1\} = \{(q, x) : w \geq \hat{w}(r, T_1, T_2)\}$$

Two level sets of this function are shown in Figure 3.

The expression for J^* also allows us to compute the least restrictive controller that renders the set W_1^* invariant. It turns out to be unique:

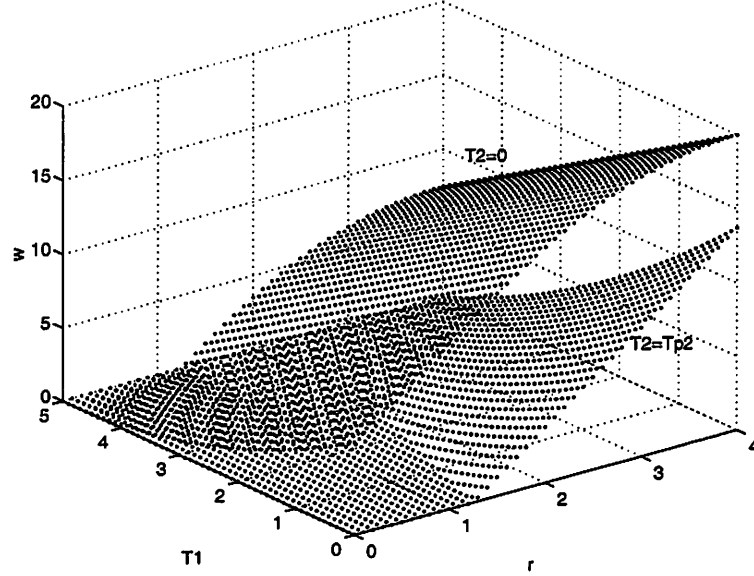


Figure 3: Lower limit on w to avoid draining

Lemma 2 *The feedback controller g_1^1 given by:*

$$\begin{aligned}
 &u_1 \in \{0, 1\} \text{ and } u_2 \in \{0, 1\} \text{ if } [w > \hat{w}(r, 0, 0)] \vee [w < \hat{w}(r, T_1, T_2)] \\
 &u_1 = 1 \text{ and } u_2 \in \{0, 1\} \text{ if } \hat{w}(r, 0, 0) \geq w > \hat{w}(r, T_1, 0) \\
 &u_1 \in \{0, 1\} \text{ and } u_2 = 1 \text{ if } \hat{w}(r, 0, 0) \geq w > \hat{w}(r, 0, T_2) \\
 &u_1 = 1 \text{ and } u_2 = 1 \text{ if } w = \hat{w}(r, T_1, T_2)
 \end{aligned}$$

is the unique, least restrictive, non-blocking, feedback controller that renders W_1^{1} invariant.*

Proof: See [4]. ■

Note that the first term applies to states in the interior of the safe set ($w > \hat{w}(r, 0, 0)$) as well as all the states outside the safe set ($w < \hat{w}(r, T_1, T_2)$). The expression for \hat{w} (see [4]) suggests that \hat{w} is monotone in T_1 and T_2 . Therefore, the condition on the last case is enabled if and only if all other conditions fail. The two middle conditions may overlap, however. Therefore there is some nondeterminism in the choice of safe controls (some states may be safe with either one or the other pump on, but not neither).

References

- [1] J.-R. Abrial, E. Börger, and H. Langmaack, “The steam-boiler case study project, an introduction”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS. Springer Verlag, 1996.
- [2] J.-R. Abrial, “The steam-boiler control specification problem”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS. Springer Verlag, 1996.
- [3] Tomas A. Henzinger and Howard Wong-Toi, “Using HYTECH to synthesize control parameters for a steam boiler”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS, pp. 265–282. Springer Verlag, 1996.

- [4] John Lygeros, Claire Tomlin, and Shankar Sastry, "Controllers for reachability specifications for hybrid systems", *Automatica*, pp. 349–370, March 1999.

ee291E Lecture 22: Controller Synthesis: Game Theoretic Approach

Omid Shakernia John Lygeros

April 14, 1999

1 Game Theoretic Controller Synthesis

To guarantee that a safety specification is met despite the action of the disturbances we cast the design problem as a zero sum dynamic game. The two players in the game are the control u and the disturbance d and they compete over cost a function that encodes the safety specification. We seek the best possible control action and the worst possible disturbance. Note that if the specifications can be met for this pair then they can also be met for any other choice of the disturbance.

Consider a controller synthesis problem $(H, \Box F)$. The game can be cast in the standard min-max setting by introducing a cost function induced by the discrete metric. The discrete metric is simply a map $m : (\mathbf{Q} \times \mathbf{X}) \times (\mathbf{Q} \times \mathbf{X}) \rightarrow \mathbb{R}$ defined by:

$$m((q_1, x_1), (q_2, x_2)) = \begin{cases} 0 & \text{if } (q_1, x_1) = (q_2, x_2) \\ 1 & \text{if } (q_1, x_1) \neq (q_2, x_2) \end{cases}$$

It is easy to check that m satisfies the axioms of a metric. The metric induces a map on subsets of $\mathbf{Q} \times \mathbf{X}$ by defining:

$$\begin{aligned} M : 2^{\mathbf{Q} \times \mathbf{X}} \times 2^{\mathbf{Q} \times \mathbf{X}} &\rightarrow \mathbb{R} \\ (W_1, W_2) &\mapsto \min_{((q_1, x_1), (q_2, x_2)) \in W_1 \times W_2} m((q_1, x_1), (q_2, x_2)) \end{aligned}$$

In other words, $M(W_1, W_2) = 0$ if $W_1 \cap W_2 \neq \emptyset$ and $M(W_1, W_2) = 1$ if $W_1 \cap W_2 = \emptyset$.

Consider an execution, $\chi = (\tau, q, x, (u, d))$, of the hybrid automaton H starting at an initial state $(q_0, x_0) \in I$. Define the *cost* of this execution by:

$$\begin{aligned} J : \mathcal{H} &\rightarrow \mathbb{R} \\ \chi &\mapsto \min_{t \in \tau} M(\{(q(t), x(t))\}, F^c) \end{aligned}$$

Note that J can only take on two values, $J(\chi) \in \{0, 1\}$. Therefore, J implicitly defines a property of the hybrid automaton. In fact:

Proposition 1 $J(\chi) = 1$ if and only if $\Box F(\chi) = \text{True}$.

Intuitively, u tries to maximize the cost function J (prevent the state from leaving F). Because we have no control over the actions of d , we assume that it tries to minimize J (force the state to leave F). As we would like to establish conditions under which $\Box F$ is guaranteed to be satisfied, we bias the game in favor of the disturbance whenever there is ambiguity over how the game will proceed. For example, multiple executions may be possible for the same initial condition, control and disturbance trajectories, due to nondeterminism. Moreover, the order in which the two players play in the game may be important, if one player is assumed to

have access to the decision of the other player before choosing his/her action. In both these cases we would like to give the disturbance the “benefit of the doubt”.

Consider the max-min solution :

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau, q, x, (u, d)) \in \mathcal{H}_g} J(\chi) \right) \quad (1)$$

Motivated by Proposition 1 we restrict our attention to feedback strategies for u in equation (1). Following the standard game theoretic convention, the “player” who appears first in the right hand side of equation (1) (the controller g) is also assumed to play first. The player who appears second (the disturbance d) is assumed to have access to the strategy of the first player, when called upon to make his/her decision. The minimum over χ removes all nondeterminism. Therefore, provided a solution to this equation can be found, J^* is a well defined function of the initial state (q_0, x_0) . In addition, the minimum over χ implicitly restricts attention to control and disturbance trajectories that satisfy the state based input constraint ϕ .

Using J^* we define a set $W^* \subseteq \mathbf{Q} \times \mathbf{X}$ by:

$$W^* = \{(q_0, x_0) \in \mathbf{Q} \times \mathbf{X} \mid J^*(q_0, x_0) = 1\} \quad (2)$$

Proposition 2 *W^* is the maximal controlled invariant subset of F .*

Proof: We first show W^* is controlled invariant. Assume for the sake of contradiction that it is not. Then for all g there exists an $(q_0, x_0) \in W^*$, a d , a $\chi = (\tau, q, x, (u, d)) \in \mathcal{H}_g$ and a $t \in \tau$ with $(q(t), x(t)) \notin W^*$. By Proposition 1, $J(\chi) = 0$, which, by equation (1), implies that $J^*(q_0, x_0) = 0$. This contradicts the assumption that $(q_0, x_0) \in W^*$.

Next we show that W^* is maximal. Assume for the sake of contradiction that it is not, that is there exists another controlled invariant \hat{W} with $W^* \subset \hat{W} \subseteq F$. Then, by definition, there exists a controller g such that \mathcal{H}_g satisfies $\Box F$ with $I = \hat{W}$. In other words, for all $(q_0, x_0) \in \hat{W}$, for all d and for all $\chi = (\tau, q, x, (u, d)) \in \mathcal{H}_g$, $\Box F(\chi) = \text{True}$, or, equivalently, $J(\chi) = 1$. But, from equation (1) this would imply that $J^*(q_0, x_0) = 1$. This contradicts the assumption that $W^* \subset \hat{W}$. ■

If a solution to equation 1 can be computed, then there exists a feedback controller (namely one that achieves the maximum) that renders W^* invariant. We would like to find a least restrictive such controller. Our ability to solve the synthesis problem using this technique hinges on finding a solution to equation 1. In some cases this can be done by brute force; this typically involves guessing a controller and showing that it achieves the maximum. More systematically, this can be done using optimal control techniques, in particular dynamic programming.

2 Example: The Steam Boiler

Recall the steam boiler problem from Lecture 8 (Figure 1). The problem was introduced first in [1, 2]. The model presented here is taken from [3]. The controller synthesis for this model can be found in [4]. The continuous dynamics of the boiling process are summarized by the differential equations:

$$\begin{aligned} \dot{w} &= p_1 + p_2 - r \\ \dot{r} &= d \end{aligned}$$

The dynamics of pump i are summarized by the open hybrid automaton of Figure 2. Notice that p_i is both an output variable of the pump and an input variable of the boiling process. For a formal definition of the model (with slight differences in the notation) please refer to Lecture 8.

The composite automaton has 4 continuous, real valued state variables:

$$x = (w, r, T_1, T_2) \in \mathbb{R}^4$$

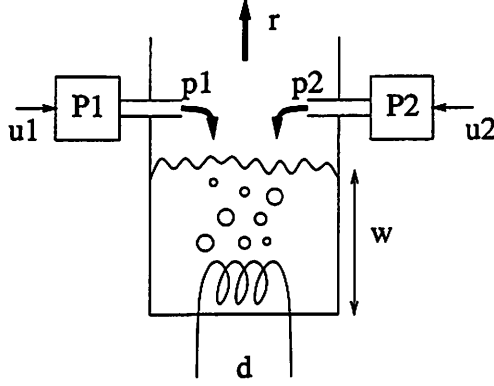


Figure 1: The Steam Boiler

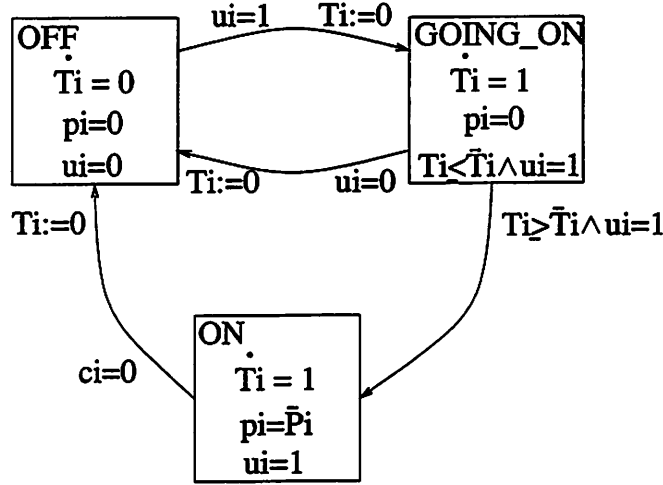


Figure 2: The pump hybrid automaton

9 discrete states:

$$q \in \{(OFF, OFF), (OFF, GOING_ON), \dots, (ON, ON)\}$$

2 discrete input variables:

$$(u_1, u_2) \in \{0, 1\} \times \{0, 1\} = \mathbf{U}$$

and one continuous input variable:

$$d \in [-D_1, D_2] = \mathbf{D}$$

As the notation suggests, u_1 and u_2 will play the role of controls and d will play the role of the disturbance. The additional requirement that $r \in [0, R]$ can be encoded by a state dependent input constraint:

$$\phi(q, x) = \begin{cases} \mathbf{U} \times [0, D_2] & \text{if } r \leq 0 \\ \mathbf{U} \times \mathbf{D} & \text{if } r \in (0, R) \\ \mathbf{U} \times [-D_1, 0] & \text{if } r \geq R \end{cases}$$

Proposition 3 *If $\text{Init} \subseteq \mathbf{Q} \times \mathbb{R} \times [0, R] \times \mathbb{R}^2$, then for all $\chi = (\tau, q, x, u_1, u_2, d)$ and for all $t \in \tau$, $r(t) \in [0, R]$.*

Our goal is to design a controller that keeps the water level in a given range, $[M_1, M_2]$, with $0 \leq M_1 < M_2$. This requirement can easily be encoded by a safety property $(Q \cup X, \Box F)$ with

$$F = \mathbf{Q} \times [M_1, M_2] \times \mathbb{R}^3$$

We will try to achieve this goal by treating the situation as a game between (u_1, u_2) and d over the cost function J . Recall that this involves solving the equation:

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau, g, x, (u, d)) \in \mathcal{H}_g} J(\chi) \right)$$

Fortunately, for this example, the equation simplifies considerably.

First, notice that the steam boiler system is deterministic, in the sense that for each initial state and each input sequence consistent with ϕ the automaton accepts a unique execution. In this case, we can represent an execution more compactly by $((q_0, x_0), (u_1, u_2), d)$ with the interpretation that (u_1, u_2) and d represent the entire sequence for these variables. Moreover, if the memoryless controller we pick is single valued, this implies we need not worry about the innermost minimization.

Next notice that J can be encoded by means of two real valued cost functions:

$$J_1(x^0, u_1, u_2, d) = \inf_{t \geq 0} w(t) \quad \text{and} \quad J_2(x^0, u_1, u_2, d) = -\sup_{t \geq 0} w(t) \quad (3)$$

Clearly:

$$J = 1 \Leftrightarrow (J_1 \geq M_1) \wedge (J_2 \geq -M_2)$$

The problem of finding a solution to the game over the discrete cost function (known as *qualitative game* or *game of kind*) reduces to finding solutions to two real valued games (known as *quantitative games* or *games of degree*). Even though there is no obvious benefit to doing this, it allows us to use tools from continuous optimal control to address the problem.

Start with the game over J_1 . Guess a possible solution:

$$u_i^*(q, x) = 1 \text{ for all } (q, x), \quad \text{and} \quad d^*(q, x) = \begin{cases} D_2 & \text{if } r < R \\ 0 & \text{if } r = R \end{cases} \quad (4)$$

Notice that both players resort to a feedback strategy (a trivial one).

Lemma 1 (u_1^*, u_2^*, d^*) is globally a saddle solution for the game between (u_1, u_2) and d over J_1 .

Proof: See [4]. ■

A *saddle solution* is a solution to equation (1) for which the order in which the players make their decisions turns out to be unimportant. In other words, a solution for which for all (q, x) :

$$J_1^*(q, x) = \max_{(u_1, u_2)} \min_d J_1((q, x), (u_1, u_2), d) = \min_d \max_{(u_1, u_2)} J_1((q, x), (u_1, u_2), d)$$

Or, in other words, a solution for which for all $(q, x), u_1, u_2$ and d :

$$J_1((q, x), (u_1, u_2), d^*) \leq J_1((q, x), (u_1^*, u_2^*), d^*) \leq J_1((q, x), (u_1^*, u_2^*), d)$$

The last definition is usually somewhat easier to work with. It is in fact used to prove the lemma.

The saddle cost:

$$J_1^*(q, x) = J_1((q, x), (u_1^*, u_2^*), d^*)$$

Can be computed in closed form. This allows us then to compute the set of states for which there exists a control that for all actions of the disturbance prevents draining. This set turns out to be of the form:

$$W_1^* = \{(q, x) : J_1^*(q, x) \geq M_1\} = \{(q, x) : w \geq \hat{w}(r, T_1, T_2)\}$$

Two level sets of this function are shown in Figure 3.

The expression for J^* also allows us to compute the least restrictive controller that renders the set W_1^* invariant. It turns out to be unique:

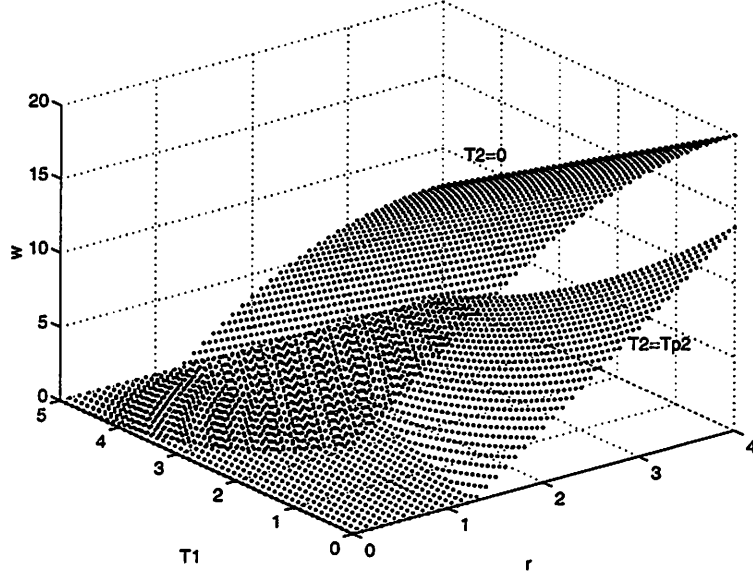


Figure 3: Lower limit on w to avoid draining

Lemma 2 *The feedback controller g_1^1 given by:*

$$\begin{aligned}
 &u_1 \in \{0, 1\} \text{ and } u_2 \in \{0, 1\} \text{ if } [w > \hat{w}(r, 0, 0)] \vee [w < \hat{w}(r, T_1, T_2)] \\
 &u_1 = 1 \text{ and } u_2 \in \{0, 1\} \text{ if } \hat{w}(r, 0, 0) \geq w > \hat{w}(r, T_1, 0) \\
 &u_1 \in \{0, 1\} \text{ and } u_2 = 1 \text{ if } \hat{w}(r, 0, 0) \geq w > \hat{w}(r, 0, T_2) \\
 &u_1 = 1 \text{ and } u_2 = 1 \text{ if } w = \hat{w}(r, T_1, T_2)
 \end{aligned}$$

is the unique, least restrictive, non-blocking, feedback controller that renders W_1^{1} invariant.*

Proof: See [4]. ■

Note that the first term applies to states in the interior of the safe set ($w > \hat{w}(r, 0, 0)$) as well as all the states outside the safe set ($w < \hat{w}(r, T_1, T_2)$). The expression for \hat{w} (see [4]) suggests that \hat{w} is monotone in T_1 and T_2 . Therefore, the condition on the last case is enabled if and only if all other conditions fail. The two middle conditions may overlap, however. Therefore there is some nondeterminism in the choice of safe controls (some states may be safe with either one or the other pump on, but not neither).

References

- [1] J.-R. Abrial, E. Börger, and H. Langmaack, “The steam-boiler case study project, an introduction”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS. Springer Verlag, 1996.
- [2] J.-R. Abrial, “The steam-boiler control specification problem”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS. Springer Verlag, 1996.
- [3] Tomas A. Henzinger and Howard Wong-Toi, “Using HYTECH to synthesize control parameters for a steam boiler”, in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, J.-R. Abrial, E. Börger, and H. Langmaack, Eds., number 1165 in LNCS, pp. 265–282. Springer Verlag, 1996.

- [4] John Lygeros, Claire Tomlin, and Shankar Sastry, "Controllers for reachability specifications for hybrid systems", *Automatica*, pp. 349–370, March 1999.

ee291E Lecture 23: Controller Synthesis: Discrete Systems

Pedro Arroyo & John Lygeros

April 19, 1999

1 Overview

- Abstractly introduced a gaming technique for determining the maximal controlled invariant subset of a given set and a controller that renders this set invariant.
- Illustrated how the technique can be applied in an example, where the solution to the game can be guessed.
- In the next few lectures we will show how the solution can be constructed more systematically using optimal control tools, more specifically dynamic programming.
- We will start by discussing gaming techniques for purely discrete and purely continuous systems and then show how to extend the results to hybrid systems.

2 Finite State Machines

We begin our wondrous exploration by expressing a purely discrete system using the hybrid formalism. A purely discrete system can be viewed as a hybrid automaton containing trivial continuous dynamics.

Consider a plant automaton $H = (Q, X, V, \text{Init}, f, I, E, G, R, \phi)$, with:

- $Q = \{q\}$, $|Q| < \infty$ (finite discrete states);
- $X = \{x\}$, $\dot{X} = \{0\}$ (trivial continuous state);
- $V = \{u, d\}$, $|V| < \infty$ (finite inputs);
- $\text{Init} \subseteq Q \times \{0\}$ (drop the trivial dependence on the continuous state from now on);
- $f(q, x, v) = 0$ (trivial continuous dynamics);
- $I(q) = \emptyset$ (no time evolution);
- $E \subset Q \times Q$;
- $G(e) \subseteq \{0\} \times V$ (drop the trivial dependence on the continuous state from now on);
- $R(e, 0, v) = \{0\}$; and,
- $\phi(q, 0) = V$.

Notice the similarity between this hybrid automaton and the finite automata considered earlier in the class. Recall that a finite automaton is a collection $(\mathbf{Q}, \Sigma, \Delta, q_0, Q_F)$ consisting of a finite set of states, a finite alphabet, a transition relation, an initial state and a set of final states. Comparing the two definitions we see that \mathbf{Q} plays the same role in both cases, \mathbf{V} plays the role of Σ , Init plays the role of q_0 , while $Q_F = \mathbf{Q}$ (restrictions on where we should and should not end up will be encoded in terms of the specification later on). The transition relation is given by:

$$(q, v, q') \in \Delta \Leftrightarrow ((q, q') \in E) \wedge (v \in G(q, q'))$$

Recall that the transition relation can also be encoded by means of a transition map:

$$\delta(q, v) = \{q' \in \mathbf{Q} : ((q, q') \in E) \wedge (v \in G(q, q'))\}$$

To prevent trivial safe solutions we add a non-blocking assumption:

$$\forall q \in \mathbf{Q}, \forall u \in \mathbf{U}, \exists d \in \mathbf{D} \text{ such that } \delta(q, (u, d)) \neq \emptyset$$

Remarks:

1. The dependence on the trivial continuous state will be dropped to simplify the notation.
2. Strictly speaking the hybrid automaton definition of a finite state system is slightly more general, since many initial states are allowed. In any case, Init will not be very important, since we will be trying to determine the largest set of initial conditions for which a safety specification can be satisfied.
3. All the action takes place at a single point in time (the executions are over time sequences of the form $[0, 0][0, 0], \dots$). An alternative interpretation is that time has been abstracted away. This can be encoded by setting $I(q) = \{0\}$ for all q ; the interpretation is that the transition can take place after an unspecified amount of delay. In either case, the execution can be reduced down to a pair of sequences, one for the states $(q[i])$ and one for the inputs $(v[i])$ such that:

$$q[0] \in \text{Init} \text{ and } \forall i \geq 0, q[i+1] \in \delta(q[i], v[i])$$

4. Because time has been abstracted away and $\phi(q) = \mathbf{V}$, the non-blocking assumption eliminates all possibilities of cheating by the controller (zeno executions are meaningless in this setting).
5. Players u and d play simultaneously. Subsumes turn based play [1] and priority play [2] (see [1] and [3]).

3 Computation of Controlled Invariant Sets

Consider a set of states $F \subseteq \mathbf{Q}$. Try to establish the largest set of initial states for which there exists a controller that manages to keep all executions inside F . Easiest to demonstrate by means of an example.

Consider a finite automaton with $\mathbf{Q} = \{q_1, \dots, q_{10}\}$, $\mathbf{U} = \mathbf{D} = \{1, 2\}$, $F = \{q_1, \dots, q_8\}$, and the transition structure of Figure 1 (the inputs are listed in the order (u, d) and $*$ denotes a “wild-card”). Try to establish the largest set of initial states, W^* such that there exists a feedback controller that keeps the execution inside F . Clearly:

$$W^* \subseteq F = \{q_1, \dots, q_8\}$$

Next, look at states that can end up in F^C in one transition (q_4, q_5 , and q_8). Notice that from q_4 if $u = 1$ whatever d chooses to do we remain in F , while from q_5 and q_8 whatever u chooses to do we leave F . Therefore:

$$W^* \subseteq \{q_1, q_2, q_3, q_4, q_6, q_7\}$$

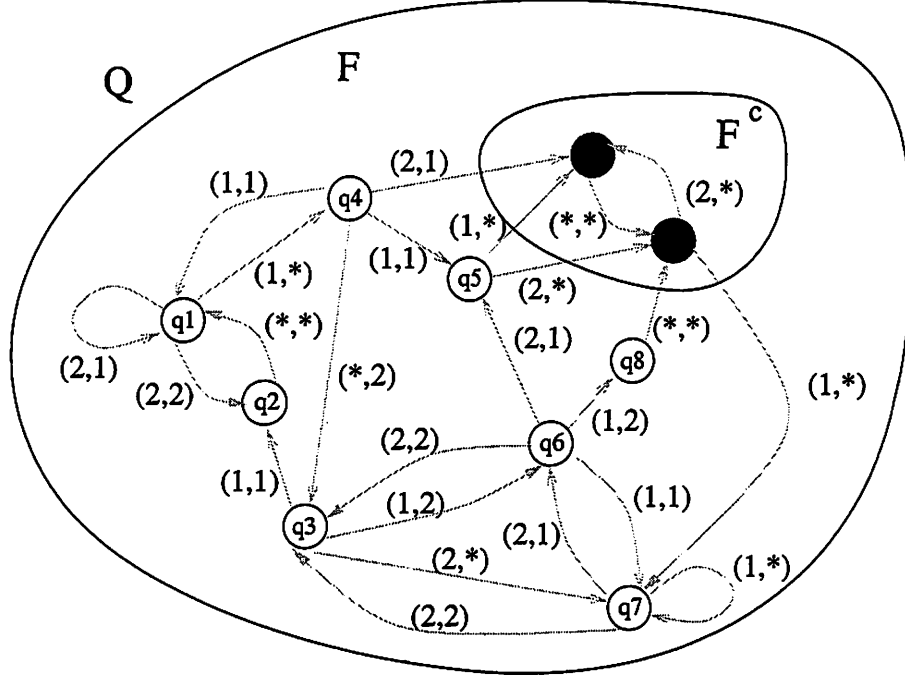


Figure 1: Example of discrete controller synthesis

Next look at states that can leave the above set in one transition (q_4 and q_6). Notice that from q_4 if $d = 1$ whatever u chooses we leave the set, while from q_6 if $u = 1$ d can choose $d = 2$ to force us to leave the set while if $u = 2$ d can choose $d = 1$ to force us to leave the set. Therefore:

$$W^* \subseteq \{q_1, q_2, q_3, q_7\}$$

From the remaining states, if u chooses according to the following feedback map:

$$g(q) = \begin{cases} \{1\} & \text{if } q = q_7 \\ \{2\} & \text{if } q \in \{q_1, q_3\} \\ \{1, 2\} & \text{if } q = q_2 \end{cases}$$

we are guaranteed to remain in $\{q_1, q_2, q_3, q_7\}$ for ever. Therefore,

$$W^* = \{q_1, q_2, q_3, q_7\}$$

and g is the least restrictive controller that renders W^* invariant (proof of both facts is by enumeration).

More generally this scheme can be implemented by the following algorithm [4]:

Algorithm 1 (Controlled Invariant Set)

Initialization:

$$W^0 = F, W^1 = \emptyset, i = 0$$

while $W^{i+1} \neq W^i$ **do**

begin

$$W^{i+1} = W^i \cap \{q \in Q : \exists u \in U \forall d \in D \delta(q, (u, d)) \subseteq W^i\}$$

$$i = i + 1$$

end

The index decreases as a reminder that the algorithm involves a predecessor operation.

This is a real algorithm (can be implemented by enumerating the set of states and inputs). The algorithm terminates after a finite number of steps since:

$$W^{i-1} \subseteq W^i \text{ and } |W^0| = |F| \leq |Q| < \infty$$

4 Relation to Gaming

What does any of this have to do with gaming? Introduce a value function:

$$J : \mathbf{Q} \times \mathbb{Z}_- \rightarrow \{0, 1\} \quad (1)$$

Consider the difference equation:

$$\begin{aligned} J(q, 0) &= \begin{cases} 1 & q \in F \\ 0 & q \in F^c \end{cases} \\ J(q, i-1) - J(q, i) &= \min\{0, \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} [\min_{q' \in \delta(q, (u, d))} J(q', i) - J(q, i)]\} \end{aligned} \quad (2)$$

We will refer to equation (2) as a *discrete Hamilton-Jacobi* equation.

Proposition 1 (Winning States for u) *A fixed point $J^* : \mathbf{Q} \rightarrow \{0, 1\}$ of (2) is reached in a finite number of iterations. The set of states produced by the algorithm is $W^* = \{x \in \mathbf{X} \mid J^*(x) = 1\}$.*

Proof: We show more generally that $W^i = \{x \in \mathbf{X} \mid J(x, i) = 1\}$. The proof is by induction (see [5]). ■

This is beginning to look more game-like. Consider what happens when a fixed point of (2) is reached. Ignoring the outermost min for the time being leads to:

$$J^*(q) = \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} \min_{q' \in \delta(q, (u, d))} J^*(q') \quad (3)$$

Notice the similarity between this and (excuse the overloading of the notation):

$$J^*(q_0, x_0) = \max_g \min_d \left(\min_{\chi=(\tau, q, z, (u, d)) \in \mathcal{H}_g} J(\chi) \right) \quad (4)$$

The equations look very similar. The only difference is that instead of having to worry about all feedback controllers, all disturbance trajectories and all possible executions we reduce the problem to a pointwise argument. Clearly equation (3) is much simpler to work with than equation (4). This is a standard trick in dynamic programming. Equation (3) is a special case of what is known in dynamic programming as Bellman's equation, while the difference equation (2) is a form of value iteration for computing a fixed point to Bellman's equation.

What about the outer min? This is an inevitable consequence of turning the sequence argument of equation (4) to a pointwise argument. It is there to prevent states that have been labeled as unsafe at some point from being relabeled as safe later on. If it were not there, then in the example of Figure 1 state q_{10} would be labeled as safe after one step of the algorithm (i.e. $J(q_{10}, -1) = 1$). The extra min operation implements the intersection with W^i in the algorithm of the previous section, ensures the monotonicity of W^i with respect to i and guarantees termination.

The fixed point of equation (2) also provides a simple characterization of a least restrictive controller that renders W^* invariant:

$$g(q) = \begin{cases} \{u \in \mathbf{U} : \min_{d \in \mathbf{D}} \min_{q' \in \delta(q, (u, d))} J^*(q') = 1\} & \text{if } q \in W^* \\ \mathbf{U} & \text{if } q \in (W^*)^c \end{cases}$$

Notice that $g(q) \neq \emptyset$ for all q by construction. Any $u \in \mathbf{U}$ needs to be allowed outside W^* to make the controller least restrictive.

Summarizing:

Proposition 2 (Characterization of W^* and g) *W^* is the maximal controlled invariant subset of F and g is the unique, least restrictive feedback controller that renders W^* invariant.*

References

- [1] Anuj Puri, *Theory of Hybrid Systems and Discrete Event Systems*, PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1995.
- [2] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems", *Proceedings of the IEEE*, vol. Vol.77, no. 1, pp. 81–98, 1989.
- [3] John Lygeros, Claire Tomlin, and Shankar Sastry, "Multi-objective hybrid controller synthesis", Tech. Rep. UCB/ERL M96/59, Electronic Research Laboratory, University of California Berkeley, 1997.
- [4] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems", in *Theoretical Aspects of Computer Science*. 1995, number 900 in LNCS, pp. 229–242, Springer Verlag.
- [5] John Lygeros, Claire Tomlin, and Shankar Sastry, "Controllers for reachability specifications for hybrid systems", *Automatica*, pp. 349–370, March 1999.

ee291E Lecture 24: Optimal Control

Jun Zhang & John Lygeros

April 21, 1999

1 Controller Synthesis Problem for Continuous Systems

A continuous system can be thought of as a hybrid automaton with:

- $Q = \{q\}$, $Q_0 = \{q_0\}$;
- $X = \{x\}$, $X = \mathbb{R}^n$;
- $V = \{u, d\}$, $U = \mathbb{R}^{m_u}$, $D = \mathbb{R}^{m_d}$;
- $\text{Init} \subseteq \mathbb{R}^n$;
- $f : X \times U \times D \rightarrow \mathbb{R}^n$;
- $E = \emptyset$;
- $I(q_0) = X$;
- $\phi(x) = V$.

Assumption 1 u and d are piecewise continuous functions of time.

Remarks:

- The trivial dependence on q will be dropped from now on.
- Strictly speaking execution takes place over a hybrid time trajectory of the form $[\tau_0, \tau'_0][\tau_1, \tau'_1] \dots$, where u and d are continuous over each interval (refer to lecture 8). Notice that $x(\cdot)$ is continuous and piecewise differentiable.
- Modulo this technicality, this is a garden variety continuous system with:

$$\begin{aligned} x(0) &\in \text{Init}, \\ \dot{x}(t) &= f(x(t), u(t), d(t)). \end{aligned}$$

- Since u and d are piecewise continuous execution can be extended over arbitrarily long time horizons. Notice that the continuous system is by definition non-blocking (since $I(q_0) = X$). The piecewise continuity assumption eliminates the only other possibility for a controller to “cheat” in a controller synthesis problem with a reachability specification; under this assumption the system can not be Zeno either.

Consider a set of states:

$$F \subseteq \mathbb{R}^n,$$

We will try to find the largest set of initial conditions for which u can keep x in F despite the actions of d .

Assume F is closed and can be characterized by a smooth function l :

$$\begin{aligned} l(x) &< 0 & \text{if } x \in F^c \\ l(x) &= 0 & \text{if } x \in \partial F \\ l(x) &> 0 & \text{if } x \in F^\circ \end{aligned}$$

We will use optimal control tools to try to maximize the value of l . This is turning a “game of kind” into a “game of degree”. Besides conceptual difference this (fairly trivial) transformation allows us to use optimal control tools to find a solution to the reachability problem.

2 A Quick Excursion into Optimal Control Theory

Consider a pair of functions

$$\begin{aligned} (x, u) : [t_0, t_f] &\rightarrow \mathbb{R}^n \times \mathbb{R}^{m_u}, \text{ such that} \\ \dot{x}(t) &= f(x(t), u(t)), \text{ where } f : \mathbb{R}^n \times \mathbb{R}^{m_u} \rightarrow \mathbb{R}^n, \\ x(t_0) &= x_0, \\ \psi(x(t_f)) &= 0, \text{ where } \psi : \mathbb{R}^n \rightarrow \mathbb{R}^p \end{aligned}$$

Among this class of functions try to find the one that minimizes the functional:

$$\begin{aligned} J(x, u) &= \phi(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t)) dt, \\ \text{where } \phi : \mathbb{R}^n &\rightarrow \mathbb{R}, \text{ and } L : \mathbb{R}^n \times \mathbb{R}^{m_u} \rightarrow \mathbb{R}, \end{aligned}$$

This general class of optimal control problems is known as the *Bolza Problem*. A number of variations exist ...

- Meyer Problem ($L = 0$)
- Lagrange Problem ($\phi = 0$)
- Fix the final time
- Allow state and input constraints
- Allow time varying f, L, ϕ, ψ .

In many cases the various formulations can be shown to be equivalent by adding auxiliary states, etc. (see [1, 2]).

Two main approaches exist to solving this problem:

1. Calculus of variations (making use of the Maximum principle);
2. Dynamic programming (making use of the Principle of optimality).

Both involve turning the minimization over the function space to a pointwise minimization. Both involve a simple observation:

1. For the calculus of variations, the observation is the optimal curve should be such that neighboring curves do not lead to smaller costs. In a sense the “derivative” of J about the optimal curve should be zero; one takes small variations about the candidate optimal solution and attempts to make the change in J zero.
2. For dynamic programming, the observation is that the optimal curve remains optimal at intermediate points.

We will briefly discuss both approaches, without giving the details of any of the theorems.

3 Calculus of variations

Consider a nonlinear possibly time varying dynamical system described by

$$\dot{x} = f(x, u, t) \quad (1)$$

with state $x(t) \in \mathbb{R}^n$ and the control input $u \in \mathbb{R}^{n_i}$. Consider the problem of minimizing the performance index

$$J = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t) dt \quad (2)$$

where t_0 is the initial time, t_f the final time (free), $L(x, u, t)$ is the running cost, and $\phi(x(t_f), t_f)$ is the cost at the terminal time. The initial time t_0 is assumed to be fixed and t_f variable. Problems involving a cost only on the final and initial state are referred to as Mayer problems, those involving only the integral or running cost are called Lagrange problems and costs of the form of equation (2) are referred to as Bolza problems. We will also have a constraint on the final state given by

$$\psi(x(t_f), t_f) = 0 \quad (3)$$

where $\psi : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^p$ is a smooth map. To derive necessary conditions for the optimum, we will perform the calculus of variations on the cost function of (2) subject to the constraints of equations (1), (3). To this end, define the modified cost function, using the Lagrange multipliers $\lambda \in \mathbb{R}^p, p(t) \in \mathbb{R}^n$,

$$\bar{J} = \phi(x(t_f), t_f) + \lambda^T \psi(x(t_f), t_f) + \int_{t_0}^{t_f} [L(x, u, t) + p^T(f(x, u, t) - \dot{x})] dt \quad (4)$$

Defining the *Hamiltonian* $H(x, u, t)$ using what is referred to as a *Legendre transformation*

$$H(x, p, u, t) = L(x, u, t) + p^T f(x, u, t) \quad (5)$$

The variation of (4) is given by assuming that a variation by $\delta u(\cdot)$ produces a variation $\delta x(\cdot), \delta p(\cdot), \delta \lambda$ so that

$$\begin{aligned} \delta \bar{J} = & (D_1 \phi + D_1 \psi^T \lambda) \delta x|_{t_f} + (D_2 \phi + D_2 \psi^T \lambda) \delta t|_{t_f} + \psi^T \delta \lambda \\ & + (H - p^T \dot{x}) \delta t|_{t_f} \\ & + \int_{t_0}^{t_f} [D_1 H \delta x + D_3 H \delta u - p^T \delta \dot{x} + (D_2 H^T - \dot{x})^T \delta p] dt \end{aligned} \quad (6)$$

The notation $D_i H$ stands for the derivative of H with respect to the i th argument. Thus, for example,

$$D_3 H(x, p, u, t) = \frac{\partial H}{\partial u} \quad D_1 H(x, p, u, t) = \frac{\partial H}{\partial x}$$

Integrating by parts for $\int p^T \delta \dot{x} dt$ yields

$$\begin{aligned} \delta \bar{J} = & (D_1 \phi + D_1 \psi^T \lambda - p^T) \delta x(t_f) + (D_2 \phi + D_2 \psi^T \lambda + H) \delta T + \psi^T \delta \lambda \\ & + \int_{t_0}^{t_f} [(D_1 H + \dot{p}^T) \delta x + D_3 H \delta u + p^T \delta \dot{x} + (D_2 H^T - \dot{x})^T \delta p] dt \end{aligned} \quad (7)$$

An extremum of \bar{J} is achieved when $\delta\bar{J} = 0$ for all independent dent variations $\delta\lambda, \delta x, \delta u, \delta p$. These conditions are recorded in the following

Table of necessary conditions for optimality:

Table 1		
Description	Equation	Variation
Final State constraint	$\psi(x(T), T) = 0$	$\delta\lambda$
State Equation	$\dot{x} = \frac{\partial H}{\partial p}^T$	δp
Costate equation	$\dot{p} = -\frac{\partial H}{\partial x}^T$	δx
Input stationarity	$\frac{\partial H}{\partial u} = 0$	δu
Boundary conditions	$D_1\phi - p^T = -D_1\psi^T\lambda _{t_f}$ $H + D_2\phi = -D_2\psi^T\lambda _{t_f}$	$\delta x(t_f)$ δt_f

The conditions of Table (3) and the boundary conditions $x(t_0) = x_0$ and the constraint on the final state $\psi(x(t_f), t_f) = 0$ constitute the necessary conditions for optimality. The end point constraint equation is referred to as the transversality condition:

$$\begin{aligned} D_1\phi - p^T &= -D_1\psi^T\lambda \\ H + D_2\phi &= -D_2\psi^T\lambda \end{aligned} \quad (8)$$

The optimality conditions may be written explicitly as

$$\begin{aligned} \dot{x} &= \frac{\partial H}{\partial p}^T(x, u^*, p, t) \\ \dot{p} &= -\frac{\partial H}{\partial x}^T(x, u^*, p, t) \end{aligned} \quad (9)$$

with the stationarity condition reading

$$\frac{\partial H}{\partial u}(x, u^*, p, t) = 0$$

and the endpoint constraint $\psi(x(t_f), t_f) = 0$.

Remarks:

- These are necessary but not sufficient conditions for optimality.
- To decide whether the candidate optimum satisfying this condition is indeed a minimum we have to take second variations.
- Notice that the number of equations ($2n$ differential equations, $2n$ constraints on the initial conditions, m pointwise equations for u , p equations from ψ and 1 equation from H) are equal to the number of unknown in the problem ($2n$ functions of time for x and p , m values of u along these functions, p real variables λ and the final time t_f)
- The key point to the derivation of the necessary conditions of optimality is that the Legendre transformation of the Lagrangian to be minimized into a Hamiltonian converts a functional minimization problem into a static optimization problem on the function $H(x, u, p, t)$.

- In the presence of input constraints the stationarity conditions can be replaced by:

$$H^*(x, p) = \min_{u \in U} H(x, p, u)$$

Let the optimal control be:

$$u^*(x, p) = \arg \min_{u \in U} H(x, p, u)$$

- The system of equations for p and x form a *two point boundary value problem*. Notice that the “initial condition” for x is given at time t_0 while the “initial condition” for p is given at time t_f . Therefore, this system of equations can not be solved by straight forward simulation, in fact it may be very difficult to solve.
- The solution is given “open loop” as a function of $p(t)$. Typically the calculus of variations does produce feedback solutions directly.

4 Dynamic Programming

Recall we are trying to minimize

$$J(x, u) = \phi(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t)) dt$$

subject to

$$\begin{aligned}\dot{x} &= f(x, u), \\ x(t_0) &= x_0, \\ \psi(x(t_f)) &= 0.\end{aligned}$$

For any arbitrary $t \in [t_0, t_f]$ and any $x \in \mathbb{R}^n$ introduce the *value function* (also known as *optimal return function* or *cost-to-go function*):

$$J^*(x, t) = \min_{u(\cdot)} \left\{ \phi(x(t_f)) + \int_t^{t_f} L(x(\tau), u(\tau)) d\tau \right\}$$

Assume that J^* is continuously differentiable as a function of both x and t and consider the following argument:

- Assume the system starts at x at time t and proceeds for Δt “seconds” using some arbitrary u .
- At time $t + \Delta t$ we find ourselves at x' .
- Assume we move optimally from x' onwards. Then the cost accumulated by time t_f will be $J^*(x, t + \Delta t)$.
- If Δt is small enough then $x' \approx x + f(x, u)\Delta t$,
- Let $J'(x, t)$ be the total cost of moving from (x, t) to the end via $(x', t + \Delta t)$. Then:

$$J'(x, t) = J^*(x', t + \Delta t) + L(x, u)\Delta t$$

- Clearly, $J^*(x, t) \leq J'(x, t)$, since $J^*(x, t)$ is the minimum cost we could ever hope to achieve starting from x at time t . In fact, if the choice of u over $[t, t + \Delta t]$ were the optimal equality would be achieved.

$$\begin{aligned}
J^*(x, t) &= \min_{u \in U} \{ J^*(x', t + \Delta t) + L(x, u) \Delta t \} \\
&= \min_{u \in U} \{ J^*(x + f(x, u) \Delta t, t + \Delta t) + L(x, u) \Delta t \} \\
\text{if } \Delta t \text{ is small} &= \min_{u \in U} \left\{ J^*(x, t) + \frac{\partial J^*}{\partial x} f(x, u) \Delta t + \frac{\partial J^*}{\partial t} \Delta t + L(x, u) \Delta t \right\} \\
\frac{\partial J^*}{\partial x} \Delta t &= - \min_{u \in U} \left\{ \frac{\partial J^*}{\partial x} f(x, u) \Delta t + L(x, u) \Delta t \right\}
\end{aligned}$$

Taking the limit as $\Delta t \rightarrow 0$, leads to the Hamilton-Jacobi-Bellman partial differential equation:

$$\frac{\partial J^*}{\partial x} = - \min_{u \in U} \left\{ \frac{\partial J^*}{\partial x} f(x, u) + L(x, u) \right\}$$

with the boundary condition $J^*(x, t_f) = \phi(x)$ wherever $\psi(x) = 0$. If a differentiable solution to the equation exists, then it can be used to produce the optimal control:

$$u^*(x, t) = \arg \min_{u \in U} \left\{ L(x, u) + \frac{\partial J^*}{\partial x} f(x, u) \right\}$$

Using the definition of the optimal Hamiltonian from the previous section, the Hamilton-Jacobi-Bellman partial differential equation can be written more compactly as:

$$\begin{aligned}
L(x, u) + \frac{\partial J^*}{\partial x} f(x, u) &= H(x, \left(\frac{\partial J^*}{\partial x}\right)^T, u) \\
\frac{\partial J^*}{\partial t} &= -H^*(x, \left(\frac{\partial J^*}{\partial t}\right)^T, u) \quad H^* = \min_u H.
\end{aligned}$$

Remarks:

- The relation between the two techniques can be highlighted by observing that $\frac{\partial J^*}{\partial x}$ plays the role of p .
- The Hamilton-Jacobi-Bellman equation requires one to assume that the value function is continuously differentiable. This assumption is not necessary to apply the maximum principle. On the other hand, the Hamilton-Jacobi-Bellman equation naturally leads to feedback controls.

References

- [1] Arthur E. Bryson and Yu-Chi Ho, *Applied Optimal Control*, Hemisphere Publishing Corporation, 1975.
- [2] L.D. Berkovitz, *Optimal Control Theory*, Springer-Verlag, 1974.

Lectures in Optimal Control and Dynamical Games

S. S. Sastry

April 20, 1999

1 Optimal Control

There are numerous excellent books on optimal control. Commonly used books which we will draw from are Athans and Falb [1], Pontryagin et al [3], Young [4], Kirk [5], Lewis [6] and Fleming and Rishel[7]. The history of optimal control is quite well rooted in antiquity, with allusion being made to Dido, the first Queen of Carthage, who when asked to take as much land as could be covered by an ox-hide, cut the ox-hide into a tiny strip and proceeded to enclose the entire area of what came to be know as Carthage in a circle of the appropriate radius¹. The calculus of variations is really the ancient precursor to optimal control. Iso perimetric problems of the kind that gave Dido her kingdom were treated in detail by Tonelli and later by Euler. Both Euler and Lagrange laid the foundations of mechanics in a variational setting culminating in the Euler Lagrange equations. Newton used variational methods to determine the shape of a body that minimizes drag, and Bernoulli formulated his brachistochrone problem in the seventeenth century, which attracted the attention of Newton and L'Hôpital. This intellectual heritage was revived and generalized by Bellman [2] in the context of dynamic programming and by Pontryagin and his school in the so-called Pontryagin principle for optimal control ([3]).

Consider a nonlinear possibly time varying dynamical system described by

$$\dot{x} = f(x, u, t) \quad (1)$$

with state $x(t) \in \mathbb{R}^n$ and the control input $u \in \mathbb{R}^{n_i}$. Consider the problem of minimizing the performance index

$$J = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t) dt \quad (2)$$

where t_0 is the initial time, t_f the final time (free), $L(x, u, t)$ is the running cost, and $\phi(x(t_f), t_f)$ is the cost at the terminal time. The initial time t_0 is assumed to be fixed and t_f variable. Problems involving a cost only on the final and initial state are referred to as Mayer problems, those involving only the integral or running cost are called Lagrange problems and costs of the form of equation (2) are referred to as Bolza problems. We will also have a constraint on the final state given by

$$\psi(x(t_f), t_f) = 0 \quad (3)$$

¹The optimal control problem here is to enclose the maximum area using a closed curve of given length.

where $\psi : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^p$ is a smooth map. To derive necessary conditions for the optimum, we will perform the calculus of variations on the cost function of (2) subject to the constraints of equations (1), (3). To this end, define the modified cost function, using the Lagrange multipliers $\lambda \in \mathbb{R}^p, p(t) \in \mathbb{R}^n$,

$$\tilde{J} = \phi(x(t_f), t_f) + \lambda^T \psi(x(t_f), t_f) + \int_{t_0}^{t_f} [L(x, u, t) + p^T(f(x, u, t) - \dot{x})] dt \quad (4)$$

Defining the *Hamiltonian* $H(x, u, t)$ using what is referred to as a *Legendre transformation*

$$H(x, p, u, t) = L(x, u, t) + p^T f(x, u, t) \quad (5)$$

The variation of (4) is given by assuming that a variation by $\delta u(\cdot)$ produces a variation $\delta x(\cdot), \delta p(\cdot), \delta \lambda$ so that

$$\begin{aligned} \delta \tilde{J} = & (D_1 \phi + D_1 \psi^T \lambda) \delta x|_{t_f} + (D_2 \phi + D_2 \psi^T \lambda) \delta t|_{t_f} + \psi^T \delta \lambda \\ & + (H - p^T \dot{x}) \delta t|_{t_f} \\ & + \int_{t_0}^{t_f} [D_1 H \delta x + D_3 H \delta u - p^T \delta \dot{x} + (D_2 H^T - \dot{x})^T \delta p] dt \end{aligned} \quad (6)$$

The notation $D_i H$ stands for the derivative of H with respect to the i th argument. Thus, for example,

$$D_3 H(x, p, u, t) = \frac{\partial H}{\partial u} \quad D_1 H(x, p, u, t) = \frac{\partial H}{\partial x}$$

Integrating by parts for $\int p^T \delta \dot{x} dt$ yields

$$\begin{aligned} \delta \tilde{J} = & (D_1 \phi + D_1 \psi^T \lambda - p^T) \delta x(t_f) + (D_2 \phi + D_2 \psi^T \lambda + H) \delta T + \psi^T d\lambda \\ & + \int_{t_0}^{t_f} [(D_1 H + \dot{p}^T) \delta x + D_3 H \delta u - p^T \delta \dot{x} + (D_2^T - \dot{x})^T \delta p] dt \end{aligned} \quad (7)$$

An extremum of \tilde{J} is achieved when $\delta \tilde{J} = 0$ for all independent variations $\delta \lambda, \delta x, \delta u, \delta p$. These conditions are recorded in the following

Table of necessary conditions for optimality:

Table 1

Description	Equation	Variation
Final State constraint	$\psi(x(T), T) = 0$	$\delta \lambda$
State Equation	$\dot{x} = \frac{\partial H}{\partial p}$	δp
Costate equation	$\dot{p} = -\frac{\partial H}{\partial x}$	δx
Input stationarity	$\frac{\partial H}{\partial u} = 0$	δu
Boundary conditions	$D_1 \phi - p^T = -D_1 \psi^T \lambda _{t_f}$ $H + D_2 \phi = -D_2 \psi^T \lambda _{t_f}$	$\delta x(t_f)$ δt_f

The conditions of Table (1) and the boundary conditions $x(t_0) = x_0$ and the constraint on the final state $\psi(x(t_f), t_f) = 0$ constitute the necessary conditions for optimality. The end point constraint equation is referred to as the transversality condition:

$$\begin{aligned} D_1\phi - p^T &= -D_1\psi^T\lambda \\ H + D_2\phi &= -D_2\psi^T\lambda \end{aligned} \quad (8)$$

The optimality conditions may be written explicitly as

$$\begin{aligned} \dot{x} &= \frac{\partial H^T}{\partial p}(x, u^*, p) \\ \dot{p} &= -\frac{\partial H^T}{\partial x}(x, u^*, p) \end{aligned} \quad (9)$$

with the stationarity condition reading

$$\frac{\partial H}{\partial u}(x, u^*, p) = 0$$

and the endpoint constraint $\psi(x(t_f), t_f) = 0$. *The key point to the derivation of the necessary conditions of optimality is that the Legendre transformation of the Lagrangian to be minimized into a Hamiltonian converts a functional minimization problem into a static optimization problem on the function $H(x, u, p, t)$.*

The question of when these equations also constitute sufficient conditions for (local) optimality is an important one and needs to be ascertained by taking the second variation of \tilde{J} . This is an involved procedure but the input stationarity condition in Table (1) hints at the **sufficient condition for local minimality** of a given trajectory $x^*(\cdot), u^*(\cdot), p^*(\cdot)$ being a local minimum as being that the Hessian of the Hamiltonian,

$$D_2^2 H(x^*, u^*, p^*, t) \quad (10)$$

being positive definite along the optimal trajectory. A sufficient condition for this is to ask simply that the $n_i \times n_i$ Hessian matrix

$$D_2^2 H(x, u, p, t) \quad (11)$$

be positive definite. As far as conditions for **global minimality** are concerned, again the stationarity condition hints at a sufficient condition for global minimality being that

$$u^*(t) = \underset{\{\min \text{ over } u\}}{\operatorname{argmin}} H(x^*(t), u, p^*(t), t) \quad (12)$$

Sufficient conditions for this are, for example, the convexity of the Hamiltonian $H(x, u, p, t)$ in u .

Finally, there are instances in which the Hamiltonian $H(x, u, p, t)$ is not a function of u at some values of x, p, t . These cases are referred to as *singular extremals* and need to be treated with care, since the value of u is left unspecified as far as the optimization is concerned.

1.1 Fixed Endpoint problems

In the instance that the final time t_f is fixed, the equations take on a simpler form, since there is no variation in δt_f . Then, the boundary condition of equation (8) becomes

$$p^T(t_f) = D_1\phi + D_1\psi^T\lambda|_{t_f} \quad (13)$$

Further, if there is no final state constraint the boundary condition simplifies even further to

$$p(t_f) = D_1\phi^T|_{t_f} \quad (14)$$

1.2 Time Invariant Systems

In the instance that $f(x, u, t)$ and the running cost $L(x, u, t)$ are not explicitly functions of time, there is no final state constraint and the final time t_f is fixed, the formulas of Table (1) can be rewritten as

State Equation	$\dot{x} = \frac{\partial H^T}{\partial p} = f(x, u^*)$
Costate Equation	$\dot{p} = -\frac{\partial H^T}{\partial x} = -D_1f^Tp + D_1L^T$
Stationarity Condition	$0 = \frac{\partial H}{\partial u} = D_2L^T + D_2f^Tp$
Transversality Conditions	$D_1\phi - p^T = -D_1\psi^T\lambda$ $H(t_f) = 0$

In addition, it may be verified that

$$\frac{dH^*}{dt} = \frac{\partial H^*}{\partial x}(x, p)\dot{x} + \frac{\partial H^*}{\partial p}\dot{p} = 0 \quad (15)$$

thereby establishing that $H(t) \equiv 0$.

1.3 Connections with Classical Mechanics

Hamilton's principle of least action states (under certain conditions ²) that a conservative system moves so as to minimize the time integral of its "action", defined to be the difference between the kinetic and potential energy. To make this more explicit we define $q \in \mathbb{R}^n$ to be the vector of generalized coordinates of the system and denote by $U(q)$ the potential energy of the system and $T(q, \dot{q})$ the kinetic energy of the system. Then Hamilton's principle of least action asks to solve an optimal control problem for the system

$$\dot{q} = u$$

²For example, there is no dissipation or no nonholonomic constraints. Holonomic or integrable constraints are dealt with by adding appropriate Lagrange multipliers. If nonholonomic constraints are dealt with in the same manner, we get equations of motion, dubbed vakonomic by Arnold [8] which do not correspond to experimentally observed motions. On the other hand, if there are only holonomic constraints, the equations of motion that we derive from Hamilton's principle of least action is equivalent to Newton's laws.

with Lagrangian

$$L(q, u) = T(q, u) - U(q)$$

The equations (9) in this context have $H(q, u, p) = L(q, u) + p^T u$. $u^* = u^*(p, q)$ is chosen so as to minimize the Hamiltonian H . A necessary condition for stationarity is that $u^*(p, q)$ satisfies

$$0 = \frac{\partial H}{\partial u} = \frac{\partial L}{\partial u} + p \quad (16)$$

The form of the equations (9) in this context is that of the familiar *Hamilton Jacobi equations*. The costate p has the interpretation of momentum.

$$\begin{aligned} \dot{q} &= \frac{\partial H^*}{\partial p}(p, q) = u^*(p, q) \\ \dot{p} &= -\frac{\partial H^*}{\partial q}(p, q) \end{aligned} \quad (17)$$

Combining the second of these equations with (16) yields the familiar *Euler Lagrange equations*

$$\frac{\partial L}{\partial q} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right) = 0 \quad (18)$$

1.4 Hamilton Jacobi Bellman Equation

To begin this discussion, we will embed the optimization problem which we are solving in a larger class of problems, more specifically we will consider the original cost function of equation (2) from an initial time $t \in [t_0, t_f]$ by considering the cost function on the interval $[t, t_f]$:

$$J(x(t), t) = \phi(x(t_f), t_f) + \int_t^{t_f} L(x(\tau), u(\tau), \tau) d\tau$$

Bellman's principle of optimality says that if we have found the optimal trajectory on the interval from $[t, t_f]$ by solving the optimal control problem on that interval, the resulting trajectory is also optimal on all subintervals of this interval of the form $[t, t_f]$ with $t > t_0$, provided that the initial condition at time t was obtained from running the system forward along the optimal trajectory from time t . The optimal value of $J(x(t), t)$ is referred to as the "cost-to go". To be able to state the following key theorem of optimal control we will need to define the "optimal Hamiltonian" to be

$$H^*(x, p, t) := H(x, u^*, p, t)$$

Theorem 1 The Hamilton Jacobi Bellman equation

Consider, the time invariant optimal control problem of (2) with fixed endpoint t_f and time invariant dynamics. If the optimal value function, i.e. $J^*(x(t_0), t_0)$ is a smooth function of x, t , then $J^*(x, t)$ satisfies the **Hamilton Jacobi Bellman partial differential equation**

$$\frac{\partial J^*}{\partial t}(x, t) = -H^*(x, \frac{\partial J^*}{\partial x}(x, t), t) \quad (19)$$

with boundary conditions given by $J^*(x, t_f) = \phi(x, t_f)$ for all $x \in \{x : \psi(x, t_f) = 0\}$.

Proof: The proof uses the principle of optimality. This principle says that if we have found the optimal trajectory on the interval from $[t, t_f]$ by solving the optimal control problem on that interval, the resulting trajectory is also optimal on all subintervals of this interval of the form $[t_1, t_f]$ with $t_1 > t$, provided that the initial condition at time t_1 was obtained from running the system forward along the optimal trajectory from time t . Thus, from using $t_1 = t + \Delta t$, it follows that

$$J^*(x, t) = \min_{u(\tau)} \left[\int_t^{t+\Delta t} L(x, u, \tau) d\tau + J^*(x + \Delta x, t + \Delta t) \right] \quad (20)$$

Taking infinitesimals and letting $\Delta t \rightarrow 0$ yields that

$$-\frac{\partial J^*}{\partial t} = \min_{u(t)} \left(L + \left(\frac{\partial J^*}{\partial x} \right) f \right) \quad (21)$$

with the boundary condition being that the terminal cost is

$$J^*(x, t_f) = \phi(x, t_f)$$

on the surface $\psi(x) = 0$. Using the definition of the Hamiltonian in equation (5), it follows from equation (21) that the Hamilton Jacobi equation of equation (19) holds.

□

Remarks:

1. The preceding theorem was stated as a necessary condition for extremal solutions of the optimal control problem. As far as minimal and global solutions of the optimal control problem, the Hamilton Jacobi Bellman equations read as in equation (21). In this sense, the form of the Hamilton Jacobi Bellman equation in (21) is more general.
2. The **Eulerian conditions** of Table (1) are easily obtained from the Hamilton Jacobi Bellman equation by proving that $p^T(t) := \frac{\partial J^*}{\partial x}(x, t)$ satisfies the costate equations of that Table. Indeed, consider the equation (21). Since $u(t)$ is unconstrained, it follows that it should satisfy

$$\frac{\partial L}{\partial u}(x^*, u^*) + \frac{\partial f^T}{\partial u} p = 0 \quad (22)$$

Now differentiating the definition of $p(t)$ above with respect to t yields

$$\frac{dp^T}{dt} = \frac{\partial^2 J^*}{\partial t \partial x}(x^*, t) + \frac{\partial^2 J^*}{\partial x^2} f(x^*, u^*, t) \quad (23)$$

Differentiating the Hamilton Jacobi equation (21) with respect to x and using the relation (22) for a stationary solution yields

$$-\frac{\partial^2 J^*}{\partial t \partial x}(x^*, t) = \frac{\partial L}{\partial x} + \frac{\partial^2 J^*}{\partial x^2} f + p^T \frac{\partial f}{\partial x} \quad (24)$$

Using equation (24) in equation (23) yields

$$-\dot{p} = \frac{\partial f^T}{\partial x} p + \frac{\partial L^T}{\partial x} \quad (25)$$

establishing that p is indeed the co-state of Table 1. The boundary conditions on $p(t)$ follow from the boundary conditions on the Hamilton Jacobi Bellman equation.

1.5 Constrained Input Problems

In the instance that there are no constraints on the input, the extremal solutions of the optimal control problem are found by simply extremizing the Hamiltonian and deriving the stationarity condition. Thus, if the specification is that $u(t) \in U \subset \mathbb{R}^{n_i}$ then, the optimality condition is that

$$H(x^*, u^*, p^*, t) \leq H(x^*, u, p^*, t) \quad \forall u \in U \quad (26)$$

If the Hamiltonian is convex in u and U is a convex set, there are no specific problems with this condition. In fact, when there is a single input and the set U is a single closed interval, there are several interesting examples of Hamiltonians for which the optimal inputs switch between the endpoints of the interval, resulting in what is referred to as **bang bang control**. However, problems can arise when U is either not convex or compact. In these cases, a concept of a **relaxed control** taking values in the convex hull of U needs to be introduced. As far as an implementation of a control $u(t) \in \text{conv}U$, but not in U , a probabilistic scheme involving switching between values of U whose convex combination u is needs to be devised.

1.6 Free end time problems

In the instance that the final time t_f is free, the transversality conditions are that

$$\begin{aligned} p^T(t_f) &= D_1\phi + D_1\psi^T\lambda \\ H(t_f) &= -(D_2\phi + D_2\psi^T\lambda) \end{aligned} \quad (27)$$

1.6.1 Minimum time problems

A special class of minimum time problems of especial interest is minimum time problems, where t_f is to be minimized subject to the constraints. This is accounted for by setting the Lagrangian to be 1, and the terminal state cost $\phi \equiv 0$, so that the Hamiltonian is $H(x, u, p, t) = 1 + p^T f(x, u, t)$. Note that by differentiating $H(x, u, p, t)$ with respect to time, we get

$$\frac{dH^*}{dt} = D_1H^*\dot{x} + D_2H^*\dot{u} + D_3H^*\dot{p} + \frac{\partial H^*}{\partial t} \quad (28)$$

Continuing the calculation using the Hamilton Jacobi equation,

$$\frac{dH^*}{dt} = \left(\frac{\partial H^*}{\partial x} + \dot{p}\right)f(x, u^*, t) + \frac{\partial H^*}{\partial t} = \frac{\partial H^*}{\partial t} \quad (29)$$

In particular, if H^* is not an explicit function of t , it follows that $H^*(x, u, p, t) \equiv H$. Thus, for minimum time problems for which $f(x, u, t)$ and $\psi(x, t)$ are not explicitly functions of t , it follows that $0 = H(t_f) \equiv H(t)$.

2 Two person zero sum dynamical games

The theory of games also has a long and distinguished, if not as long a history. Borel encountered saddle points in a matrix in 1915, but it really took von Neumann to prove his fundamental theorem about the existence of mixed strategies for achieving a saddle solution in games in 1936. In a classic book, von Neumann and Morgenstern ([9]) laid out the connections between static games and economic behavior. Nash, von Stackelberg and others extended the work to N person non-cooperative games. This work has continued in many important new directions in economics. Differential or dynamical games showed up in the work of Isaacs in 1969 and rapidly found fertile ground in the control community where it has progressed. There are several nice books on the subject of dynamical games, our treatment is drawn heavily from Basar and Olsder ([10]).

Consider a so-called dynamical, two person zero sum, perfect state information game modeled by

$$\dot{x} = f(x, u, d, t) \quad x(t_0) = x_0 \quad (30)$$

on a fixed duration $[t_0, t_f]$. Here $x \in \mathbb{R}^n$ models the state of the system and $u \in \mathbb{R}^{n_1}, d \in \mathbb{R}^{n_2}$ represent the actions of the two players. The game is said to be *zero sum* if one player seeks to minimize and the other to maximize the same cost function taken to be of the form

$$J = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x, u, d, t) dt \quad (31)$$

We will assume that player 1 (u) is trying to minimize J and player 2 (d) is trying to maximize J . For simplicity we have omitted the final state constraint and also assumed the end time t_f to be fixed. These two assumptions are made for simplicity but we will discuss the t_f free case when we study pursuit evasion games. The game is said to have perfect information if both players have access to the full state $x(t)$. The solution of two person zero sum games proceeds very much along the lines of the optimal control problem by setting up the Hamiltonian

$$H(x, u, d, p, t) = L(x, u, d, t) + p^T f(x, u, d, t) \quad (32)$$

Rather than simply minimizing $H(x, u, d, p, t)$ the game is said to have a *saddle point solution* if the following analog of the saddle point condition for two person zero sum static games holds:

$$\min_u \max_d H(x, u, d, p, t) = \max_d \min_u H(x, u, d, p, t) \quad (33)$$

If the minmax is equal to the maxmin, the resulting optimal Hamiltonian is denoted $H^*(x, p, t)$ and the optimal inputs u^*, d^* are determined to be respectively,

$$u^*(t) = \underset{u}{\operatorname{argmin}} \left(\max_d H(x, u, d, p, t) \right) \quad (34)$$

and

$$d^*(t) = \underset{d}{\operatorname{argmax}} \left(\min_u H(x, u, d, p, t) \right) \quad (35)$$

The equations for the state and costate and the transversality conditions are given as before by

$$\begin{aligned}\dot{x} &= \frac{\partial H^*}{\partial p}(x, p) \\ \dot{p} &= -\frac{\partial H^*}{\partial x}(x, p)\end{aligned}\tag{36}$$

with boundary conditions $x(t_0) = x_0$ and $p^T(t_f) = D_1\phi(x_{t_f})$, and the equation is the familiar *Hamilton Jacobi equation*. As before, one can introduce the optimal cost to go $J^*(x(t), t)$ and we have the following analog of Theorem (1):

Theorem 2 The Hamilton Jacobi Isaacs equation

Consider, the two person zero sum differential game problem of (31) with fixed endpoint t_f . If the optimal value function, i.e. $J^(x(t_0), t_0)$ is a smooth function of x, t , then $J^*(x, t)$ satisfies the Hamilton Jacobi Isaacs partial differential equation*

$$\frac{\partial J^*}{\partial t}(x, t) = -H^*(x, \frac{\partial J^*}{\partial x}(x, t), t)\tag{37}$$

with boundary conditions given by $J^*(x, t_f) = \phi(x)$ for all x .

Remarks

1. We have dealt with saddle solutions for unconstrained input signals u, d thus far in the development. If the inputs are constrained to lie in sets U, D respectively the saddle solutions can be guaranteed to exist if

$$\min_{u \in U} \max_{d \in D} H(x, u, d, p, t) = \max_{d \in D} \min_{u \in U} H(x, u, d, p, t)\tag{38}$$

Again, if the input sets are not convex, relaxed controls may be needed to achieve the min-max.

2. The sort of remarks that were made about free endpoint optimal control problems can also be made of games.
3. In our problem formulation for games, we did not include explicit terminal state constraints of the form $\psi(x(t_f), t_f) = 0$. These can be easily included, and we will study this situation in greater detail under the heading of pursuit evasion games.
4. The key point in the theory of dynamical games is that the Legendre transformation of the Lagrangian cost function into the Hamiltonian function converts the solution of the “dynamic” game into a “static” game, where one needs to find a saddle point of the Hamiltonian function $H(x, u, d, p, t)$. This is very much in the spirit of the calculus of variations and optimal control.

3 N person dynamical games

When there are N persons playing a game, many new and interesting new possibilities arise. There is a scenario in which the N agents are non-cooperative, and another in which they cooperate in teams. Of course, if the information available to each of them is different, this makes the solution even more interesting. In this section, we will assume that each of the agents has access to the full state of the system. In this section, we will only discuss non-cooperative solution concepts: first the Nash solution concept and then briefly the Stackelberg solution concept. Cooperative games with total cooperation are simply optimal control problems. If there is cooperation among teams, this can be viewed as a noncooperative game between the teams. When however there are side payments between teams the scope of the problem increases quite considerably.

3.1 Non-cooperative Nash solutions

When there are N players each able to influence the process by controls $u_i \in \mathbb{R}^{n_i}, i = 1, \dots, N$, modeled by

$$\dot{x} = f(x, u_1, \dots, u_N, t) \quad (39)$$

and each cost functional (to be minimized) is of the form

$$J_i(u_1(\cdot), \dots, u_N(\cdot)) = \phi_i(x(t_f), t_f) + \int_{t_0}^{t_f} L_i(x, u_1, \dots, u_N, t) dt \quad (40)$$

different solution concepts need to be invoked. The simplest non-cooperative solution strategy is a so-called *non-cooperative Nash equilibrium*. A set of controls $u_i^*, i = 1, \dots, N$ is said to be a Nash strategy if for each player modifying that strategy, and assuming that the others play their Nash strategies, results in an increase in his payoff, that is for $i = 1, \dots, N$

$$J_i(u_1^*, \dots, u_i, \dots, u_N^*) \geq J_i(u_1^*, \dots, u_i^*, \dots, u_N^*) \quad \forall u_i(\cdot) \quad (41)$$

It is important to note that Nash equilibria may not be unique. It is also easy to see that for 2 person zero sum games, a Nash equilibrium is a saddle solution.

As in the previous section on saddle solutions, we can write Hamilton Jacobi equations for Nash equilibria by defining Hamiltonians $H_i(x, u_1, \dots, u_N, p, t)$ according to

$$H_i(x, u_1, \dots, u_N, p, t) = L_i(x, u_1, \dots, u_N) + p^T f(x, u_1, \dots, u_N, t) \quad (42)$$

The conditions for a Nash equilibrium of equation (41) are there exist $u_i^*(x, p, t)$ such that

$$H_i(x, u_1^*, \dots, u_i, \dots, u_N^*, p, t) \geq H_i(x, u_1^*, \dots, u_i^*, \dots, u_N^*, p, t) \quad (43)$$

Then, we have N sets of Hamilton Jacobi equations for the N players satisfying the Hamilton Jacobi equations with $H_i^* = H_i^*(x, u_1^*, \dots, u_N^*, p_i, t)$. Note that we have changed the costate variables to p_i to account for different Hamiltonians and boundary conditions.

$$\begin{aligned} \dot{x} &= \frac{\partial H_i^*}{\partial p_i} \\ \dot{p}_i &= -\frac{\partial H_i^*}{\partial x} \end{aligned} \quad (44)$$

with transversality conditions $p_i^T(t_f) = -D_1 \phi_i(x(t_f), t_f)$.

3.2 Noncooperative Stackelberg solutions

Stackelberg or hierarchical equilibria refer to noncooperative solutions, where one or more of the players act as leaders. We will illustrate the solution concept for a two person game where player 1 is the leader and player 2 the follower. Once player 1 announces a strategy $u_1^o(\cdot)$, if player 2 is rational he choose his strategy so as to minimize his cost J_2 with the dynamics

$$\dot{x} = f(x, u_1^o, u_2, t) \quad (45)$$

and

$$J_2(u_2) = \phi_2(x(t_f), t_f) + \int_{t_0}^{t_f} L_2(x, u_1^o(t), u_2, t) dt$$

Thus, $u_2^*(u_1^o)$ is chosen to minimize $H_2(x, u_1^o, u_2, p_2, t)$, where p_2 satisfies the differential equation

$$\dot{p}_2 = -\frac{\partial H_2}{\partial x}^T(x, u_1^o, u_2(u_1^o), p, t) \quad p_2(t_f) = D_1^T \phi_2(x(t_f), t_f)$$

In turn the leader chooses his strategy to be that u_1^* which minimizes J_1 subject to the assumption that player 2 will rationally play $u_2^*(u_1^*)$. Thus, the system of equations that he has to solve to minimize J_1 subject to

$$\begin{aligned} \dot{x} &= f(x, u_1, u_2, t) & x(t_0) &= x_0 \\ \dot{p}_2 &= -\frac{\partial H_2}{\partial x}^T(x, u_1^o, u_2(u_1^o), p, t) & p_2(t_f) &= D_1^T \phi_2(x(t_f), t_f) \\ 0 &= D_3 H_2(x, u_1, u_2, p_2, t) \end{aligned} \quad (46)$$

The last equation in (46) is the stationarity condition for minimizing H_2 . The optimization problem of the system in (46) is not a standard optimal control in \mathbb{R}^{2n} because there is an equality to be satisfied. Thus, Lagrange multipliers (co-states) taking values in \mathbb{R}^{2n+n_2} for $t \in [t_0, t_f]$ are needed. We will omit the details in these notes.

References

- [1] M. Athans and P. Falb, *Optimal Control*, Mc Graw Hill, 1966.
- [2] R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.
- [3] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mischenko, *The Mathematical Theory of Optimal Processes*, Wiley-New York, 1962.
- [4] L. Young, *Optimal Control Theory*, Cambridge University Press, 2nd edition, 1980.
- [5] D. Kirk, *Optimal Control Theory: An Introduction*, Prentice Hall, 1970.
- [6] F. Lewis, *Optimal Control*, Wiley-New York, 1986.
- [7] W. Fleming and R. Rishel, *Deterministic and Stochastic Optimal Control*, Springer Verlag, 1975.

- [8] V. Arnold, *Mathematical Methods of Classical Mechanics*, Springer Verlag, 2nd edition, 1989.
- [9] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1947.
- [10] T. Basar and G. Olsder, *Dynamic Noncooperative Games*, Academic Press, 2nd Edition, 1995.

ee291E Lecture 25: Dynamic Programming & Game Theory

John Lygeros

April 26, 1999

1 Dynamic Programming

Recall that we would like to find functions:

$$(x, u) : \mathbb{R} \rightarrow \mathbb{R}^n \times \mathbb{R}^m$$

that minimize:

$$J(x, u) = \phi(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t)) dt \quad (1)$$

subject to:

$$\dot{x}(t) = f(x(t), u(t)), \quad x(t_0) = x_0 \quad (2)$$

$$\psi(x(t_f)) = 0 \quad (3)$$

where $L : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$, $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$, $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ Lipschitz continuous in x and continuous in u and $\psi : \mathbb{R}^n \rightarrow \mathbb{R}$.

In Lecture 24 we discussed how this problem can be related to the solution of a partial differential equation, known as the Hamilton-Jacobi-Bellman equation, by introducing the *value function* (or “cost to go” function), $J^* : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$, given by:

$$J^*(x, t) = \min_{u(\cdot)} \left\{ \phi(x(t_f)) + \int_t^{t_f} L(x(\tau), u(\tau)) d\tau \right\}$$

We argued that if a continuously differentiable solution to the Hamilton-Jacobi-Bellman partial differential equation [1]:

$$\frac{\partial J^*}{\partial t}(x, t) = - \min_{u \in U} \left\{ \frac{\partial J^*}{\partial x}(x, t) f(x, u) + L(x, u) \right\} \quad (4)$$

with boundary condition $J^*(x, 0) = \phi(x)$ on $\psi(x) = 0$ can be found, then the optimal controls are given by:

$$u^*(x, t) = \arg \min_{u \in U} \left\{ \frac{\partial J^*}{\partial x} f(x, u) + L(x, u) \right\}$$

Equation (4) can be written more compactly if we introduce the Hamiltonian, $J^* : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$:

$$H(x, p, u) = L(x, u) + p^T f(x, u)$$

and define the optimal Hamiltonian as:

$$H^*(x, p) = \min_{u \in U} H(x, p, u)$$

Then equation (4) becomes:

$$\frac{\partial J^*}{\partial t} = -H^* \left(x, \frac{\partial J^*}{\partial x}(x, t) \right)$$

Notice that we have effectively turned the problem of optimizing a cost over the space of curves (an infinite dimensional vector space), to optimizing a cost pointwise over a finite dimensional vector space. Of course to achieve this we are still required to solve a partial differential equation.

Remarks:

1. The solution is close to the solution obtained through the calculus of variations. Simply replace the co-state p by $\frac{\partial J^*}{\partial x}(x, t)$.
2. On the positive side, dynamic programming (unlike the calculus of variations) inherently leads to feedback solutions.
3. On the negative side, dynamic programming requires one to assume differentiability of the value functions.
4. Differentiability is difficult to guarantee. Even if all the data is “smooth”, the solution to the PDE may still develop “corners” (known as *shocks*). The situation is exasperated by the fact that the min operation is continuous but not smooth.
5. Optimal controls are often *bang-bang*. Consider the system:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \quad (\text{affine in } u) \\ L &: \mathbb{R}^n \rightarrow \mathbb{R} \quad (\text{independent of } u) \\ u &\in [U_1, U_2] \quad (\text{compact control set}) \end{aligned}$$

Then:

$$H \left(x, \frac{\partial J^*}{\partial x}(x, t), u \right) = L(x) + \frac{\partial J^*}{\partial x}(x, t)f(x) + \frac{\partial J^*}{\partial x}(x, t)g(x)u$$

therefore:

$$u^*(x, t) = \begin{cases} U_1 & \text{if } \frac{\partial J^*}{\partial x}(x, t)g(x) > 0 \\ [U_1, U_2] & \text{if } \frac{\partial J^*}{\partial x}(x, t)g(x) = 0 \\ U_2 & \text{if } \frac{\partial J^*}{\partial x}(x, t)g(x) < 0 \end{cases}$$

Notice that u^* switches between its extreme values whenever $\frac{\partial J^*}{\partial x}(x, t)g(x)$ changes sign. Therefore, even if J^* is continuously differentiable, H^* is continuous, but not continuously differentiable.

6. If U is not compact, then the optimal controls may be undefined pointwise (consider for example the previous situation with $U = (-\infty, \infty)$ or $U = (U_1, U_2)$).
7. The situation may be even worse if U is not convex. The optimal controls may only be defined in the space of relaxed controls, which are not piecewise continuous as a function of time [2].
8. Finally, both dynamic programming and the calculus of variations depend on local arguments (small perturbations about the optimal solution). For certain systems these arguments may fail. For example, there may be a curve connecting the initial point to a desired final point, but no neighboring curves have this property. This leads to *abnormal extremals*, which are not captured by the above arguments and have to be studied separately.

2 Game Theory

Game theory is related to optimal control, with the difference that there are two player (the control variables are divided into two classes) with possibly conflicting objectives. The simplest case is the *two player, zero sum game*. Consider again a curve:

$$(x, u, d) : \mathbb{R} \rightarrow \mathbb{R}^n \times \mathbb{R}^{m_u} \times \mathbb{R}^{m_d}$$

Assume that d is trying to minimize and u is trying to maximize the function:

$$J(x, u, d) = \phi(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t), d(t)) dt$$

subject to:

$$\dot{x}(t) = f(x(t), u(t), d(t)), \quad x(t_0) = x_0 \quad (5)$$

$$\psi(x(t_f)) = 0 \quad (6)$$

Definition 1 A pair (u^*, d^*) is called a saddle equilibrium if for all u, d :

$$J(x, u, d^*) \leq J(x, u^*, d^*) \leq J(x, u^*, d)$$

Dynamic programming arguments can also be used to characterize saddle equilibria [3]. As before introduce a Hamiltonian:

$$H(x, p, u, d) = L(x, u, d) + p^T f(x, u, d)$$

Proposition 1 If there exists a continuously differentiable function $J^* : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$ such that:

$$\begin{aligned} \frac{\partial J^*}{\partial t}(x, t) &= -\max_{u \in U} \min_{d \in D} H\left(x, \frac{\partial J^*}{\partial x}(x, t), u, d\right) \\ &= -\min_{d \in D} \max_{u \in U} H\left(x, \frac{\partial J^*}{\partial x}(x, t), u, d\right) \\ &= H\left(x, \frac{\partial J^*}{\partial x}(x, t), u^*, d^*\right) \end{aligned}$$

then (u^*, d^*) is a saddle equilibrium.

The proof is similar to the one given in the optimal control case, and relies on the fact that by freezing u to u^* we turn the problem to an optimal control problem for d (and vice versa).

Remarks:

1. The above partial differential equation is known as the *Isaacs equation*, and the minmax = maxmin requirement is known as the *Isaacs condition*.
2. Saddle equilibria do not always exist.
3. A variational formulation is also possible.
4. The same remarks about convexity and compactness of the control and disturbance sets apply.
5. The generalization of the saddle equilibrium concept to multi player games is known as the *Nash equilibrium*. Assume there are N players, each trying to minimize their own cost function:

$$J_i(x, u_1, \dots, u_N), \quad i = 1, \dots, N$$

Definition 2 (u_1^*, \dots, u_N^*) is a non-cooperative Nash equilibrium if for all i and for all u_i :

$$J_i(x, u_1^*, \dots, u_i, \dots, u_N^*) \geq J_i(x, u_1^*, \dots, u_i^*, \dots, u_N^*)$$

The non-cooperative qualification needs to be introduced, since in this setting it may be possible for “gangs” to form: players may be able to improve their returns by collaborating with other players. Note that the saddle equilibrium concept is implicitly non-cooperative, because the game is zero sum.

The solution concept we have in mind for controller synthesis is not as symmetric as the saddle equilibrium, since we give the benefit of the doubt to the disturbance. The interpretation is that the control picks its strategy and lets the disturbance know about it. The disturbance then does its best to damage the controllers plans using this information. Consider a two player game with cost functionals J_1 and J_2 that players 1 and 2 are trying to minimize respectively. Assume that player 1 is the *leader*, i.e. decides on a strategy and lets player 2 know before player 2 has to make a decision. Given a strategy, g_1 , for player 1 define the rational responses of player 2 as the set of strategies:

$$R_2(g_1) = \{g : J_2(g_1, g) \leq J_2(g_1, g') \text{ for all } g_2\}$$

Definition 3 g_1^* is a Stackelberg equilibrium strategy for the leader if:

$$\max_{g \in R_2(g_1^*)} J_1(g_1^*, g) = \min_{g_1} \max_{g \in R_2(g_1)} J_1(g_1, g)$$

In general, Stackelberg equilibrium strategies are difficult to compute in feedback form, since the concept is prone to “side payments”, “incentives” and “threats”. These are all techniques that can be employed by the leader to make it more appealing for player 2 to adopt a certain strategy. Fortunately, the games considered for controller synthesis will be zero sum ($J_1 = -J_2$) which ensures means like that can not be employed and the solution can be computed in feedback form using dynamic programming.

References

- [1] Arthur E. Bryson and Yu-Chi Ho, *Applied Optimal Control*, Hemisphere Publishing Corporation, 1975.
- [2] L. C. Young, *Optimal Control Theory*, Chelsea, second edition, 1980.
- [3] T. Başar and G. J. Olsder, *Dynamic Non-cooperative Game Theory*, Academic Press, second edition, 1995.

ee291E Lecture 26: Controller Synthesis for Continuous Systems

John Lygeros

April 28, 1999

1 Problem Formulation

Last time we discussed optimal control and game theoretic problem formulations for continuous dynamical systems. We also highlighted mathematical tools (calculus of variations and dynamic programming) that may be used to solve these problems. This time we will see how these tools (in particular dynamic programming) can be applied to controller synthesis for continuous systems and reachability specifications. Our treatment follows [1] and [2].

Recall that continuous systems are a special class of hybrid systems with:

- $Q = \{q\}$, $\mathbf{Q} = \{q_0\}$ (trivial discrete state);
- $X = \{x\}$, $\mathbf{X} = \mathbb{R}^n$;
- $V = \{u, d\}$, $\mathbf{U} \subseteq \mathbb{R}^{m_u}$ and $\mathbf{D} \subseteq \mathbb{R}^{m_d}$ (no discrete inputs);
- $\text{Init} \subseteq \{q_0\} \times \mathbf{X}$ (drop the trivial dependence on the discrete state from now on);
- $f : \mathbf{X} \times \mathbf{U} \times \mathbf{D} \rightarrow \mathbb{R}^n$;
- $I(q) = \mathbf{X}$;
- $E = \emptyset$ (no discrete dynamics);
- $\phi(q_0, x) = \mathbf{V}$ (no state dependent input constraints).

Dropping the trivial discrete dynamics, the system can be more compactly characterized by an ordinary differential equation:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), d(t)) \\ x(0) &= x_0 \\ u(t) &\in \mathbf{U} \\ d(t) &\in \mathbf{D} \end{aligned} \tag{1}$$

To prevent technical problems, we introduce the following assumption:

Assumption 1 *f is Lipschitz continuous in x and continuous in u and d . u and d are piecewise continuous as functions of time. \mathbf{U} and \mathbf{D} are convex and compact subsets of \mathbb{R}^{m_u} and \mathbb{R}^{m_d} respectively.*

We use \mathcal{U}_I to denote the set of piecewise continuous functions from an interval $I \subseteq \mathbb{R}$ to \mathbf{U} and \mathcal{U} for the union of \mathcal{U}_I over all I (and similarly for \mathcal{D}). The assumptions on f , u and d are needed to ensure that the system is well posed. Under this assumption the system is (in a sense) deterministic and non-blocking,

since, by a standard existence and uniqueness argument for ordinary differential equations, for every $x_0 \in \mathbb{R}^n$, $T \geq 0$, $u \in \mathcal{U}_{[0,T]}$ and $d \in \mathcal{D}_{[0,T]}$, there exists a continuous, piecewise differentiable function $x : [0, T] \rightarrow \mathbb{X}$ with $x(0) = x_0$ and $\dot{x}(t) = f(x(t), u(t), d(t))$ for all t where (u, d) is continuous (“almost everywhere”). For this reason we will use:

$$\chi = (x_0, u, d)$$

to denote the unique execution starting at $x_0 \in \mathbb{R}^n$ under control $u \in \mathcal{U}$ and disturbance $d \in \mathcal{D}$.

In addition, the piecewise continuity assumption prevents the controller from “cheating” by forcing the execution to be Zeno. Strictly speaking, the execution of the system will have to be defined over a sequence of intervals, where u and d are continuous over each interval (recall that the definition of an execution of an open automaton from Lecture 8 requires continuity of the input variables over a piece of continuous evolution). Piecewise continuity of u and d implies that there exists an execution such that every compact interval of time overlaps with a finite number of such intervals, or, in other words, there can not be an infinite number of “transitions” (in this case discontinuities in u and d) in a finite amount of time.

The assumptions on \mathcal{U} and \mathcal{D} will be needed to prevent technical problems when posing the optimal control and gaming problems (recall the discussion on bang-bang and relaxed controls)

Consider a set of states $F \subseteq \mathbb{Q}$. Try to establish the maximal control invariant subset of F , i.e. the largest set of initial states for which there exists a controller that manages to keep all executions inside F . Somewhat informally this set can be characterized as:

$$W^* = \{x_0 \in \mathbb{R}^n : \exists u \in \mathcal{U} \forall d \in \mathcal{D}, \square F(x_0, u, d) = \text{True}\}$$

The only additional caveat is that u is implemented by a memoryless controller (following the discussion of Lecture 21). To eliminate technical complications we assume that:

Assumption 2 *There exists a continuously differentiable function $l : \mathbb{R}^n \rightarrow \mathbb{R}$ such that:*

$$\begin{aligned} l(x) &> 0 & \text{if } x \in F^\circ \\ l(x) &= 0 & \text{if } x \in \partial F \\ l(x) &< 0 & \text{if } x \in F^c \\ l(x) &= 0 & \Rightarrow \frac{\partial l}{\partial x}(x) \neq 0 \end{aligned}$$

The assumption implies that F is a closed set with non-empty interior, whose boundary is a $n - 1$ dimensional manifold.

2 Dynamic Programming Solution

To apply the optimal control tools introduced in the previous lecture let $t_f = 0$, consider an arbitrary $t \leq 0$ and introduce the value function:

$$\hat{J}(x, t) = \max_{u \in \mathcal{U}_{[t, 0]}} \min_{d \in \mathcal{D}_{[t, 0]}} l(x(0))$$

Notice that this optimal control problem involves no Lagrangian ($L \equiv 0$), just a terminal cost. The game obtained in this setting falls in the class of *pursuit-evasion* games [3]. The (u, d) obtained by the above optimal control problem is a Stackelberg equilibrium for the game between control and disturbance, with the control playing the role of the leader. Recall that in general the computation of Stackelberg equilibria in feedback form may be complicated by the possibility of “incentives” and “threats”, employed by the leader to coerce the other player into a beneficial course of action. In this case the situation is simplified by the fact that the game is zero sum (any gain achieved by u is equal to a loss suffered by d), so the possibility of incentives and threats is eliminated.

\hat{J} can be computed using the dynamic programming tools discussed in the last two lectures. Introduce a Hamiltonian:

$$\begin{aligned} H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{m_u} \times \mathbb{R}^{m_d} &\longrightarrow \mathbb{R} \\ (x, p, u, d) &\longmapsto p^T f(x, u, d) \end{aligned}$$

Consider the optimal Hamiltonian:

$$H^*(x, p) = \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} H(x, p, u, d)$$

Notice again that the minimization over u and d is pointwise, as opposed to over functions of time. Then, if \hat{J} is continuously differentiable it satisfies:

$$\begin{aligned} \frac{\partial \hat{J}}{\partial t}(x, t) &= -H^*\left(x, \frac{\partial \hat{J}}{\partial x}(x, t)\right) \\ \hat{J}(x, 0) &= l(x) \end{aligned} \quad (2)$$

Notice that the evolution of the partial differential equation is “backwards” in time.

Consider the set:

$$\widehat{W}_t = \{x_0 \in \mathbf{X} : \hat{J}(x_0, t) \geq 0\}$$

This is the set of all states for which starting at $x(t) = x_0$, there exists a controller for u such that for all disturbance trajectories $d \in \mathcal{D}_{[t,0]}$, $l(x(0)) \geq 0$ or, in other words, $x(0) \in F$. This is not quite what we need yet. We would like the set of all states for which there exists a u such that for all d and for all $t' \in [t, 0]$, $x(t') \in F$. This excludes points in \widehat{W}_t which leave F at some point in $[t, 0]$ but re-enter it before time 0. This requirement can be encoded either after the computation of \hat{J} , or by modifying the Hamilton-Jacobi equation:

$$\begin{aligned} \frac{\partial J}{\partial t}(x, t) &= -H^*\left(x, \frac{\partial J}{\partial x}(x, t)\right) \\ J(x, 0) &= l(x) \end{aligned} \quad (3)$$

Compare this with the discrete Hamilton-Jacobi equation from Lecture 23:

$$\begin{aligned} J(q, 0) &= \begin{cases} 1 & q \in F \\ 0 & q \in F^c \end{cases} \\ J(q, i-1) - J(q, i) &= \min\{0, \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} [\min_{q' \in \delta(q, (u, d))} J(q', i) - J(q, i)]\} \end{aligned} \quad (4)$$

$\delta(q, (u, d))$ essentially implements the spatial partial derivative of J along the dynamics of the system. The innermost minimization is not needed in the continuous case, as the continuous system is “deterministic”. As before, we seek a stationary solution to this equation. Assume that as $t \rightarrow -\infty$, $J(x, t)$ converges to a continuously differentiable function $J^* : \mathbf{X} \rightarrow \mathbb{R}$.

Proposition 1 *The set $W^* = \{x \in \mathbf{X} : J^*(x) \geq 0\}$ is the largest controlled invariant set contained in F .*

The solution to the partial differential equation also leads to a least restrictive controller that renders W^* invariant. Consider:

$$g(x) = \begin{cases} \left\{ u \in \mathbf{U} : \min_{d \in \mathbf{D}} \left(\frac{\partial J^*(x)}{\partial x} \right)^T f(x, u, d) \geq 0 \right\} & \text{if } x \in \partial W^* \\ \mathbf{U} & \text{if } x \in (W^*)^\circ \cup (W^*)^c \end{cases} \quad (5)$$

Proposition 2 *g is the unique, least restrictive memoryless controller that renders W^* invariant.*

Notice that no constraint is imposed on u in the interior of W^* (we can delay action until we reach the boundary) and outside W^* (its too late to do anything about it, so might as well give up!).

3 Geometric interpretation

For an arbitrary time $t \leq 0$ define:

$$W_t = \{x \in \mathbf{X} : J(x, t) \geq 0\}$$

Consider and $x \in \partial W_t$ and assume that:

$$\begin{aligned} H^* \left(x, \frac{\partial J}{\partial x}(x, t) \right) &< 0 \\ \Leftrightarrow \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} H \left(x, \frac{\partial J}{\partial x}(x, t), u, d \right) &< 0 \\ \Leftrightarrow \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} \frac{\partial J}{\partial x}(x, t) f(x, u, d) &< 0 \\ \Leftrightarrow \forall u \in \mathbf{U} \exists d \in \mathbf{D} \text{ such that } \frac{\partial J}{\partial x}(x, t) f(x, u, d) &< 0 \end{aligned}$$

But $\frac{\partial J}{\partial x}(x, t)$ is the normal to the boundary of W_t at x , pointing inside W_t . Moreover, $\frac{\partial J}{\partial x}(x, t) f(x, u, d)$ is the inner product between this normal and the vector $f(x, u, d)$. Let θ be the angle between $\frac{\partial J}{\partial x}(x, t)$ and $f(x, u, d)$. Then:

$$\begin{aligned} \frac{\partial J}{\partial x}(x, t) f(x, u, d) &> 0 \quad \text{if } \theta < \pi/2 \\ \frac{\partial J}{\partial x}(x, t) f(x, u, d) &= 0 \quad \text{if } \theta = \pi/2 \\ \frac{\partial J}{\partial x}(x, t) f(x, u, d) &< 0 \quad \text{if } \theta > \pi/2 \end{aligned}$$

Therefore, the above statement is equivalent to:

for all $u \in \mathbf{U}$ there exists $d \in \mathbf{D}$ such that the normal to ∂W_t at x pointing towards the interior of W_t makes an angle greater than $\pi/2$ with $f(x, u, d)$,

or, equivalently:

for all $u \in \mathbf{U}$ there exists $d \in \mathbf{D}$ such that $f(x, u, d)$ points outside W_t .

These are points where whatever u does d can force them to leave the set W_t instantaneously. Notice that the order of the quantifiers in the above expression implies that d may depend on u , in addition to x and t . The part of the boundary of W_t where $H^* < 0$ is known as the “usable part” in the pursuit-evasion game literature.

Returning back to the Hamilton Jacobi equation, we see that for these points:

$$\begin{aligned} \frac{\partial J}{\partial t}(x, t) &= -\min \left\{ 0, H^* \left(x, \frac{\partial J}{\partial x}(x, t) \right) \right\} \\ &= -H^* \left(x, \frac{\partial J}{\partial x}(x, t) \right) \\ &> 0 \end{aligned}$$

Therefore, as t decreases, J also decreases. For these points on the boundary of W_t , J becomes negative instantaneously, and they “fall out of” W_t .

What if $H^* \geq 0$? A similar argument shows that in this case:

there exists $u \in \mathbf{U}$ such that for all $d \in \mathbf{D}$ the normal to ∂W_t at x pointing towards the interior of W_t makes an angle at most $\pi/2$ with $f(x, u, d)$,

or, equivalently:

there exists $u \in \mathbf{U}$ such that for all $d \in \mathbf{D}$, $f(x, u, d)$ either points inside W_t or is tangent to ∂W_t .

These are points for which there exists a choice of u that for all d forces the state to remain in W_t . Notice that the order of the quantifiers implies that u may only depend x and t , and not d . For these points:

$$\begin{aligned} \frac{\partial J}{\partial t}(x, t) &= -\min \left\{ 0, H^* \left(x, \frac{\partial J}{\partial x}(x, t) \right) \right\} \\ &= 0 \end{aligned}$$

Therefore, as t decreases, J remains constant. These are points that want to move towards the interior of W_t . The role of the outermost minimum is to ensure that the value of J does not increase for these points, so that W_t does not grow. This is to prevent states that have been labeled as unsafe (can reach F^c) from being relabeled as safe later on.

References

- [1] John Lygeros, Claire Tomlin, and Shankar Sastry, "Controllers for reachability specifications for hybrid systems", *Automatica*, pp. 349–370, March 1999.
- [2] C. Tomlin, J. Lygeros, and S. Sastry, "Synthesizing controllers for nonlinear hybrid systems", in *Hybrid Systems: Computation and Control*, S. Sastry and T.A. Henzinger, Eds., number 1386 in LNCS, pp. 360–373. Springer Verlag, 1998.
- [3] Joseph Lewin, *Differential Games*, Springer-Verlag, 1994.

ee291E Lecture 27: Controller Synthesis: Hybrid Systems

John Lygeros

May 3, 1999

1 Problem Formulation

Last two weeks we discussed controller synthesis for:

- Discrete systems: design characterized by a fixed point of a difference equation.
- Continuous systems: design characterized by fixed point of a partial differential equation.

In either case the solution is characterized as a fixed point of an equation, which we refer to as the Hamilton-Jacobi equation. Today, we bring the discrete and continuous parts together, and talk about hybrid systems. Our treatment follows [1] and [2]. See also [3].

Recall that the *plant* is modeled by an open hybrid automaton, $H = (Q, X, V, \text{Init}, f, I, E, G, R, \phi)$, where:

- Q is a finite collection of discrete state variables;
- X is a finite collection of continuous state variables;
- V is a finite collection of input variables. We assume $V = V_D \cup V_C$, where V_D contains discrete and V_C contains continuous variables.
- $\text{Init} \subseteq Q \times X$ is a set of initial states;
- $f : Q \times X \times V \rightarrow \mathbb{R}^n$ is an input dependent vector field;
- $I : Q \rightarrow 2^{X \times V}$ assigns to each $q \in Q$ an input dependent invariant set;
- $E \subset Q \times Q$ is a collection of discrete transitions;
- $G : E \rightarrow 2^{X \times V}$ assigns to each $e = (q, q') \in E$ a guard;
- $R : E \times X \times V \rightarrow 2^X$ assigns to each $e = (q, q') \in E$, $x \in X$ and $v \in V$ a reset relation; and,
- $\phi : Q \times X \rightarrow 2^V$ assigns to each state a set of admissible inputs.

Also recall that the set the input variables are partitioned into *controls* (that can be used to steer the system) and *disturbances* (whose values can not be controlled):

$$V = U \cup D$$

Assumption 1 *To avoid technical problems we assume that:*

1. *f is Lipschitz continuous in x and continuous in v .*

2. for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$, $\phi(q, x) = \mathbf{U} \times \mathbf{D} \neq \emptyset$.
3. for all $q \in \mathbf{Q}$ and for all $v \in \mathbf{V}$, $I(q)|_X$ is an open set.
4. for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$, and for all $u \in \mathbf{U}$ there exists $d \in \mathbf{D}$ such that:

$$[(x, u, d) \in I(q)] \vee [((x, u, d) \in G(q, q') \wedge (R(q, q', x, u, d) \neq \emptyset))]$$

Part 1 is standard, and is needed for existence of continuous evolution. Part 2 implies that acceptable control and disturbance inputs always exists, and that the choice of u can not influence the possible choices of d and vice versa. Part 3 implies that we need not worry about what happens on the boundary of the invariant set (otherwise we would have to assume transverse invariants, define the Out set, etc.). Finally, part 4 (together with part 3) implies that the controller can not block the system execution. Notice that this assumption is not symmetric for u and d . The reason is that, since we are dealing with safety specifications it will never be to the benefit of d to stop the execution.

As before we will try to establish the largest controlled invariant subset of a given set $F \subseteq \mathbf{Q} \times \mathbf{X}$, and design a controller that renders this set invariant. To avoid technical problems we assume that:

Assumption 2 F is a closed set.

We will again take an adversarial approach and treat the design as a game between u and d . Whenever possible we will give the advantage to d , in particular:

1. In the order of play (u will be the “leader” of the game).
2. When resolving non-determinism.

2 Definitions of Operators

Notice that the disturbance has two choices. It can:

1. Try to make the system “jump” outside F .
2. Try to “steer” the system outside F along continuous evolution.

The control also has two choices:

1. Try to “jump” to another state in F when the disturbance tries to steer out of F .
2. Try to “steer” the system and keep it in F along continuous evolution.

To characterize alternative 1 for the disturbance, introduce the *uncontrollable predecessor operator*, $\text{Pre}_d : 2^{\mathbf{Q} \times \mathbf{X}} \rightarrow 2^{\mathbf{Q} \times \mathbf{X}}$, which, given a set $K \subseteq \mathbf{Q} \times \mathbf{X}$ returns:

$$\text{Pre}_d(K) = K^c \cup \{(q, x) \in \mathbf{Q} \times \mathbf{X} : \forall u \in \mathbf{U} \exists d \in \mathbf{D}, q' \in \mathbf{Q} \text{ such that} \\ [(x, u, d) \in G(q, q')] \wedge [R(q, q', x, u, d) \cap K^c \neq \emptyset]\}$$

For a given K , $\text{Pre}_d(K)$ returns the set of states that are either in the complement of K , or whatever u does may find themselves in the complement of K by a discrete transition.

To characterize alternative 1 for the control, introduce the *controllable predecessor operator*, $\text{Pre}_u : 2^{\mathbf{Q} \times \mathbf{X}} \rightarrow 2^{\mathbf{Q} \times \mathbf{X}}$, which, given a set $K \subseteq \mathbf{Q} \times \mathbf{X}$ returns:

$$\text{Pre}_u(K) = K \cap \{(q, x) \in \mathbf{Q} \times \mathbf{X} : \exists u \in \mathbf{U} \text{ such that } \forall d \in \mathbf{D}, \\ [\exists q' \in \mathbf{Q} \text{ such that } (x, u, d) \in G(q, q')] \wedge \\ [(x, u, d) \notin I(q)] \wedge \\ [(x, u, d) \in G(q, q') \rightarrow R(q, q', x, u, d) \subseteq K]\}$$

For a given K , $\text{Pre}_u(K)$ returns the set of states that are already in K and there exists a control action that can force a transition that keeps the state in K .

Some simple facts about these two operators:

Proposition 1 For all $K \subseteq \mathbf{Q} \times \mathbf{X}$, $\text{Pre}_u(K) \subseteq K$, $\text{Pre}_d(K) \supseteq K^c$ and $\text{Pre}_u(K) \cap \text{Pre}_d(K) = \emptyset$

Remarks:

- The two operators are asymmetric.
- The order of the quantifiers is consistent with u being the leader in the game.
- Since all non-determinism is resolved in favor of d , u has to work harder:
 - it has to ensure a transition back into K exists (first condition in the definition of $\text{Pre}_u(K)$),
 - it has to be able to “force” the transition (no mention of $I(q)$ in the definition of $\text{Pre}_d(K)$),
 - it has to ensure all possible transitions stay in K (last condition in the definition of $\text{Pre}_u(K)$).

Finally, to characterize alternative 2 for bot u and d introduce the *reach-avoid operator*, $\text{Pre}_u : 2^{\mathbf{Q} \times \mathbf{X}} \times 2^{\mathbf{Q} \times \mathbf{X}} \rightarrow 2^{\mathbf{Q} \times \mathbf{X}}$, which, given two disjoint sets $K \subseteq \mathbf{Q} \times \mathbf{X}$ and $L \subseteq \mathbf{Q} \times \mathbf{X}$ returns:

$$\text{Reach}(K, L) = \{(q_0, x_0) \in \mathbf{Q} \times \mathbf{X} : \forall u \in \mathcal{U} \exists d \in \mathcal{D}, t \geq 0 \text{ such that} \\ [(q(t), x(t)) \in K] \wedge [\forall t' \in [0, t] ((q(t'), x(t')) \notin L)]\}$$

where $(q, x) : [0, t] \rightarrow \mathbf{Q} \times \mathbf{X}$ is a segment of continuous evolution with inputs u and d , i.e.:

$$\begin{aligned} (q(0), x(0)) &= (q_0, x_0) \text{ and } \forall t' \in [0, t] \\ q(t') &= q_0 \\ (x(t'), u(t'), d(t')) &\in I(q_0) \\ \dot{x}(t') &= f(q_0, x(t'), u(t'), d(t')) \end{aligned}$$

Given two disjoint sets K and L , the operator Reach returns the set of states for which whatever u does, d can chose an appropriate trajectory to bring the state of the system to K along continuous evolution, without going first through L .

Proposition 2 For all $K, L \subseteq \mathbf{Q} \times \mathbf{X}$ with $K \cap L = \emptyset$, $K \subseteq \text{Reach}(K, L) \subseteq L^c$.

Notice that the definition of Reach is somewhat informal, since:

- u is implicitly assumed to be a feedback strategy (in fact, without loss of generality a *memoryless* feedback strategy, see Lecture 21).
- u and d are allowed to be piecewise continuous (recall the justification for this given in Lecture 26).

3 Basic Algorithm

Using the above definitions, the following algorithm can now be formulated for computing the largest controlled invariant subset of a given set F .

Algorithm 1 (Controlled Invariant Set)

Initialization:

$$W^0 = F, W^1 = \emptyset, i = 0$$

while $W^i \neq W^{i+1}$ **do**

begin

$$W^{i-1} = W^i \setminus \text{Reach}(\text{Pre}_d(W^i), \text{Pre}_u(W^i))$$

$$i = i - 1$$

end

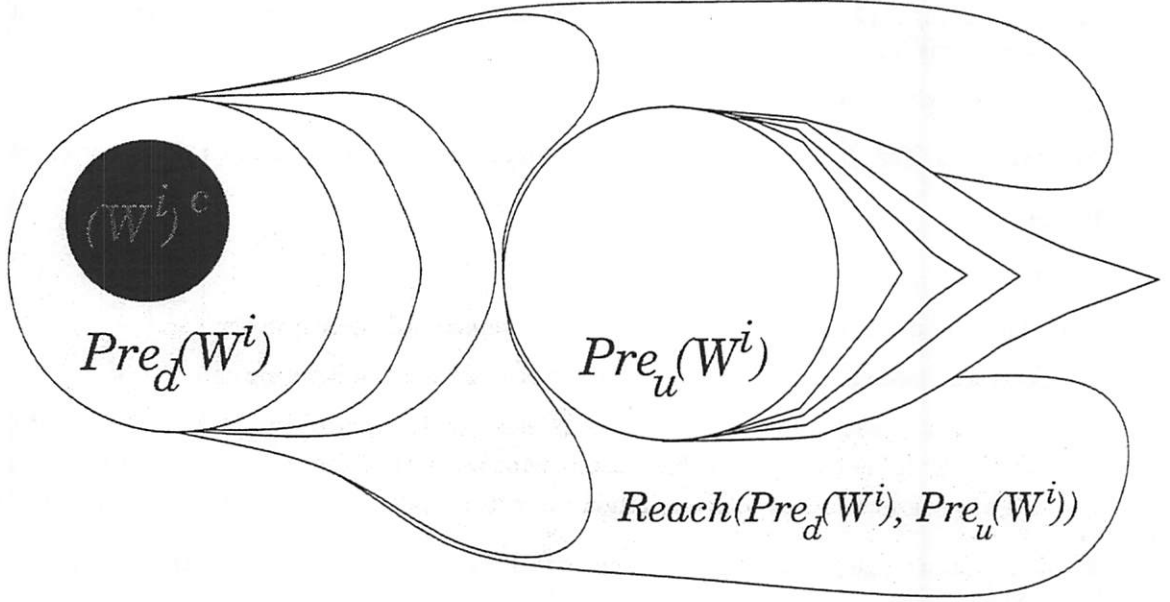


Figure 1: One step of the algorithm.

Pictorially, the operations involved in one step of the algorithm is illustrated in Figure 1.

Proposition 3 *If the algorithm terminates in a finite number of steps, the fixed point W^* is the maximal controlled invariant subset of F .*

Proof: Clearly, $W^* \subseteq \dots \subseteq W^{i-1} \subseteq W^i \subseteq \dots \subseteq F$. Assume that the algorithm terminates in a finite number of steps. Then:

$$\begin{aligned}
 W^* &= W^* \setminus \text{Reach}(\text{Pre}_d(W^*), \text{Pre}_u(W^*)) \\
 \Leftrightarrow W^* &\cap \text{Reach}(\text{Pre}_d(W^*), \text{Pre}_u(W^*)) \\
 \Leftrightarrow \text{Reach}(\text{Pre}_d(W^*), \text{Pre}_u(W^*)) &\subseteq (W^*)^c
 \end{aligned}$$

But:

$$\begin{aligned}
 (W^*)^c &\subseteq \text{Pre}_d(W^*) \subseteq \text{Reach}(\text{Pre}_d(W^*), \text{Pre}_u(W^*)) \subseteq (W^*)^c \\
 \Leftrightarrow \text{Reach}(\text{Pre}_d(W^*), \text{Pre}_u(W^*)) &= \text{Pre}_d(W^*) = (W^*)^c
 \end{aligned}$$

Consider an arbitrary $(q_0, x_0) \in W^*$. Then:

$$(q_0, x_0) \notin \text{Reach}(\text{Pre}_d(W^*), \text{Pre}_u(W^*))$$

Taking the negation of the expression for Reach this becomes:

$$\exists u \in \mathcal{U} \text{ such that } \forall d \in \mathcal{D}, \forall t \geq 0 \ [((q(t), x(t)) \notin \text{Pre}_d(W^*)) \vee [\exists t' \in [0, t] ((q(t'), x(t')) \in \text{Pre}_u(W^*))]]$$

Replacing $\text{Pre}_d(W^*)$ by $(W^*)^c$ and substituting the definition of $\text{Pre}_u(W^*)$:

$$\begin{aligned}
 \exists u \in \mathcal{U} \text{ such that } \forall d \in \mathcal{D}, \forall t \geq 0 \ [& ((q(t), x(t)) \in W^*) \vee \\
 & [\exists t' \in [0, t] \text{ such that } \exists u(t') \in \mathcal{U} \text{ such that } \forall d(t') \in \mathcal{D}, \\
 & (\exists q' \in \mathcal{Q} \text{ such that } (x(t'), u(t'), d(t')) \in G(q(t'), q')) \wedge \\
 & ((x(t'), u(t'), d(t')) \notin I(q(t')) \wedge \\
 & ((x(t'), u(t'), d(t')) \in G(q(t'), q') \rightarrow R(q(t'), q', x(t'), u(t'), d(t')) \subseteq W^*)]]
 \end{aligned}$$

In other works, if $(q_0, x_0) \in W^*$ either u can keep the system forever in W^* without any discrete transitions taking place, or it can drive the system so that $(q(\tau_1), x(\tau_1)) \in W^*$ (the first discrete transition the state is again in W^*). Controlled invariance of W^* follows by induction.

To show maximality, consider an arbitrary $(q_0, x_0) \in (W^*)^c$. Then, since the algorithm terminated in a finite number of steps, there exists i such that:

$$\begin{aligned} & (q_0, x_0) \in W^i \setminus W^{i-1} \\ \Leftrightarrow & (q_0, x_0) \in \text{Reach}(\text{Pre}_d(W^i), \text{Pre}_u(W^i)) \\ \Leftrightarrow & \forall u \in \mathcal{U} \exists d \in \mathcal{D}, t \geq 0 \text{ such that } [(q(t), x(t)) \in \text{Pre}_d(W^i)] \wedge [\forall t' \in [0, t], ((q(t'), x(t')) \notin \text{Pre}_u(W^i))] \end{aligned}$$

Substituting the definition of Pre_d leads to:

$$\begin{aligned} \forall u \in \mathcal{U} \exists d \in \mathcal{D}, \exists t \geq 0 \text{ such that } & [\forall t' \in [0, t] ((q(t'), x(t')) \notin \text{Pre}_u(W^i)) \wedge \\ & [((q(t), x(t)) \in (W^i)^c) \vee \\ & (\forall u(t) \in \mathcal{U} \exists d(t) \in \mathcal{D}, q' \in \mathcal{Q} \text{ such that } [(x(t), u(t), d(t)) \in G(q(t), q')]) \wedge \\ & [R(q(t), q', x(t), u(t), d(t)) \cap (W^i)^c \neq \emptyset])] \end{aligned}$$

Since $W^j \subseteq W^i$, for all $j \leq i$, the above shows that in finite time the state may end up either in F^c or in W^j for some $j > i$ whatever the control does (at best it ends up in W^{i+1}). Therefore, by induction, the state leaves F in finite time. ■

References

- [1] C. Tomlin, J. Lygeros, and S. Sastry, “Synthesizing controllers for nonlinear hybrid systems”, in *Hybrid Systems: Computation and Control*, S. Sastry and T.A. Henzinger, Eds., number 1386 in LNCS, pp. 360–373. Springer Verlag, 1998.
- [2] John Lygeros, Claire Tomlin, and Shankar Sastry, “Controllers for reachability specifications for hybrid systems”, *Automatica*, pp. 349–370, March 1999.
- [3] Claire Tomlin, *Hybrid Systems with Application to Air Traffic Management*, PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1998.

ee291E Lecture 28: Controller Synthesis for Hybrid Systems: Computation

John Lygeros

May 5, 1999

1 Controller Synthesis Algorithm Review

Last time an algorithm to compute the largest controlled invariant set contained in a given set F was introduced:

Algorithm 1 (Controlled Invariant Set)

Initialization:

$W^0 = F, W^1 = \emptyset, i = 0$

while $W^i \neq W^{i+1}$ **do**

begin

$W^{i-1} = W^i \setminus \text{Reach}(\text{Pre}_d(W^i), \text{Pre}_u(W^i))$

$i = i - 1$

end

where the index decreases as a reminder that the algorithm involves a predecessor operation. Recall that $\text{Pre}_d(W^i)$ is the set of states that are either already in the complement of W^i , or may find themselves in the complement of W^i through a discrete transition, whatever u does through an appropriate choice of d . $\text{Pre}_u(W^i)$, on the other hand, is the set of states that are already in W^i and there exists a u that can force a transition to another state in W^i , whatever d does. Finally, $\text{Reach}(\text{Pre}_d(W^i), \text{Pre}_u(W^i))$ is the set of states that whatever u does there exists an appropriate response by d that drives the system to $\text{Pre}_d(W^i)$ without going through $\text{Pre}_u(W^i)$ along continuous evolution.

2 Computation

This is a pseudo-algorithm, in the sense that all the operators used were defined axiomatically. Therefore, even though conceptually the algorithm will produce the largest controlled invariant set, it can not be implemented as it is in practice. To effectively implement it and guarantee termination we need to:

1. be able to store arbitrary sets of states and perform operations on them.
2. compute Pre_u and Pre_d at each step.
3. compute Reach at each step.
4. ensure that only a finite number of steps are needed.

Finite sets can be stored and manipulated by enumeration. The situation is not as simple for continuous quantities however. Here we will assume that sets of states are stored as level sets of functions. This representation is convenient conceptually, but is not necessarily effective computationally, as it shifts the problem

from storing sets to storing functions! For certain classes of sets/functions computation is in fact possible (for example, for polygons, ellipsoids, polynomials, functions encoded by neural networks, etc.). In this lecture we follow [1].

The computation of Pre_u and Pre_d is straightforward conceptually, as it involves inverting the discrete transition relation. The way to implement it computationally depends on the representation one uses to store and manipulate sets, and the sets that encode the discrete transitions (invariants, guards, reset relation). For certain classes (e.g. linear maps, polygonal sets, etc.) the computation may be feasible using quantifier elimination or optimal control tools.

Obtaining a representation of Reach is not only computationally involved, but also conceptually. The standard optimal control and game theory concepts do not deal with this type of problem, where a certain set must be reached before another. Since q remains constant along continuous evolution, the computation of Reach can be performed separately for each value of $q \in \mathbf{Q}$. The computation will involve a modification of the Hamilton-Jacobi partial differential equation. Freeze the value of q and consider two sets $K \subseteq \mathbf{X}$ and $L \subseteq \mathbf{X}$. Assume that:

Assumption 1 Assume L is closed, K is open and there exist continuously differentiable functions $l_L : \mathbf{X} \rightarrow [0, 1]$ and $l_K : \mathbf{X} \rightarrow [0, 1]$ such that $l_K(x) = 0$ for $x \in L$, $l_K > 0$ for $x \notin L$ and $l_K(x) = 1$ for $x \in \overline{K}$, whereas $l_L(x) = 0$ for $x \in \overline{K}$, $l_L > 0$ for $x \notin \overline{K}$ and $l_L(x) = 1$ for $x \in L$.

This assumption is satisfied if, for example, the sets have smooth boundaries, non-empty interiors and their closures are disjoint.

Consider the value function $J : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$ governed by the Hamilton-Jacobi equation:

$$\begin{aligned} -\frac{\partial J(x,t)}{\partial t} &= \begin{cases} \max \left\{ 0, H^*(x, \frac{\partial J(x,t)}{\partial x}^T) \right\} & \text{if } J(x,t) \geq 1 \\ H^*(x, \frac{\partial J(x,t)}{\partial x}^T) & \text{if } -1 < J(x,t) < 1 \\ \min \left\{ 0, H^*(x, \frac{\partial J(x,t)}{\partial x}^T) \right\} & \text{if } J(x,t) \leq -1 \end{cases} \\ H^*(x,p) &= \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} p^T f(q, x, u, d) \\ J(x, 0) &= l_L(x) - l_K(x) \end{aligned} \quad (1)$$

Notice that:

Proposition 1 If for some $x \in \mathbf{X}$, $J(x,t) \leq -1$ (or $J(x,t) \geq 1$) for some $t \leq 0$ then $J(x,t') \leq -1$ (or $J(x,t') \geq 1$) for all $t' \leq t$.

The situation $J(x,t) \leq -1$ corresponds to points for which whatever u does, there exists a response for d such that the state reaches K by time $|t|$, without going through L for any $0 \leq s < |t|$. The situation $J(x,t) \geq 1$ corresponds to points for which there exists a choice for u such that whatever d does the state reaches L by time $|t|$ without going through K for any $0 \leq s < |t|$. Finally, the situation $J(x,t) \in (-1, 1)$ correspond to states that are still “undecided” at time t : they may end up in K or in L by some time $|s| > |t|$ or they may never reach either set. Notice that the situation is not quite symmetric because of the order of the quantification. As usual we are looking for a stationary solution to this PDE.

Proposition 2 If a continuously differentiable solution to the above equations exists that converges to a continuously differentiable $J^* : \mathbb{R}^n \rightarrow \mathbb{R}$ as $t \rightarrow -\infty$, then $\text{Reach}(K, L) = \{x \in \mathbb{R}^n \mid J^*(x) < -1\}$.

In the limit, the set where $J^*(x) < -1$ consists of states that can ultimately be driven to K without first going through L , the set where $J^*(x) \geq 1$ consists of states that can ultimately be driven to L without first going through K , while the set of states where $J^*(x) \in [-1, 1)$ contains states that can not be forced to reach neither K nor L .

An alternative characterization of $\text{Reach}(K, L)$ involves two coupled partial differential equations. Assume continuously differentiable function $l_K : \mathbf{X} \rightarrow \mathbb{R}$ and $l_L : \mathbf{X} \rightarrow \mathbb{R}$ exists such that $K = \{x \in \mathbf{X} \mid l_K(x) < 0\}$ and

$L = \{x \in \mathbf{X}_C \mid l_L(x) \leq 0\}$. Consider the following system of coupled Hamilton-Jacobi equations:

$$-\frac{\partial J_K(x, t)}{\partial t} = \begin{cases} H_K^*(x, \frac{\partial J_K(x, t)}{\partial x}) & \text{for } \{x \in X \mid J_K(x, t) > 0\} \\ \min\{0, H_K^*(x, \frac{\partial J_K(x, t)}{\partial x})\} & \text{for } \{x \in X \mid J_K(x, t) \leq 0\} \end{cases} \quad (2)$$

and

$$-\frac{\partial J_L(x, t)}{\partial t} = \begin{cases} H_L^*(x, \frac{\partial J_L(x, t)}{\partial x}) & \text{for } \{x \in X \mid J_L(x, t) > 0\} \\ \min\{0, H_L^*(x, \frac{\partial J_L(x, t)}{\partial x})\} & \text{for } \{x \in X \mid J_L(x, t) \leq 0\} \end{cases} \quad (3)$$

where $J_K(x, t) = l_K(x(0))$ and $J_L(x, t) = l_L(x(0))$, and

$$H_K^*(x, \frac{\partial J_K}{\partial x}) = \begin{cases} 0 & \text{for } \{x \in X \mid J_L(x, t) \leq 0\} \\ \max_{u \in U} \min_{d \in D} \frac{\partial J_K}{\partial x} f(x, u, d) & \text{otherwise} \end{cases} \quad (4)$$

$$H_L^*(x, \frac{\partial J_L}{\partial x}) = \begin{cases} 0 & \text{for } \{x \in X \mid J_K(x, t) \leq 0\} \\ \min_{u \in U} \max_{d \in D} \frac{\partial J_L}{\partial x} f(x, u, d) & \text{otherwise} \end{cases} \quad (5)$$

Equation (2) describes the evolution of the set K under the Hamiltonian H_K^* . This is the solution to the “ $\max_u \min_d$ ” game for reachability in purely continuous systems, with the modification that $H_K^* = 0$ in $\{x \in \mathbf{X}_C \mid J_L(x, t) \leq 0\}$. This ensures that the evolution of $J_K(x, t)$ is frozen once this set is reached. Similarly, equation (3) describes the evolution of the set L under the Hamiltonian H_L^* . Here a “ $\min_u \max_d$ ” is used, since it is assumed that the control tries to push the system into L , to escape from K . $H_L^* = 0$ in $\{x \in \mathbf{X}_C \mid J_K(x, t) \leq 0\}$ to ensure that the evolution of $J_L(x, t)$ is frozen once this set is reached. One can show that the resulting set $\{x \in \mathbf{X}_C \mid J_K(x, t) < 0\}$ contains neither L nor states for which there is a control which drives the system into L ; and the set $\{x \in \mathbf{X}_C \mid J_L(x, t) < 0\}$ contains neither K nor states for which there is a disturbance which drives the system into K .

Proposition 3 Assume that $J_K(x, t)$ ($J_L(x, t)$ respectively) satisfies the Hamilton-Jacobi equation (2) ((3) respectively), and that it converges as $t \rightarrow -\infty$ to a function $J_K^*(x)$ ($J_L^*(x)$ respectively). Then, $\text{Reach}(K, L) = \{x \in \mathbf{X}_C \mid J_K^*(x) < 0\}$.

As for the first representation of Reach , comments relating the values of J_K^* and J_L^* and the “fate” of the corresponding states can be made.

Remarks:

- The first representation is more compact. It involves solving a PDE in $n + 1$ dimensions, as opposed to the $2n + 1$ dimensions needed for the second representation.
- The second representation is more general. For example, it does not require the closures of K and L to be disjoint. This often turns out to be needed in applications.
- In either case, we are still faced with the problem of finding a solution to PDEs. Technical problems arise since a solution in the conventional sense does not always exist or may not be unique. For example, weaker solutions concepts (e.g. viscosity solutions) may be needed if the value function turns out not to be smooth.
- Even if we decide on an appropriate solution concept, obtaining the solution computationally may still be a problem. Computation tools that may be applicable include:
 - finite element methods
 - method of characteristics
 - level set methods (these were studied in detail in the doctoral dissertation of Tomlin [2]).
 - approximate solution using basis functions
 - viability kernel computations

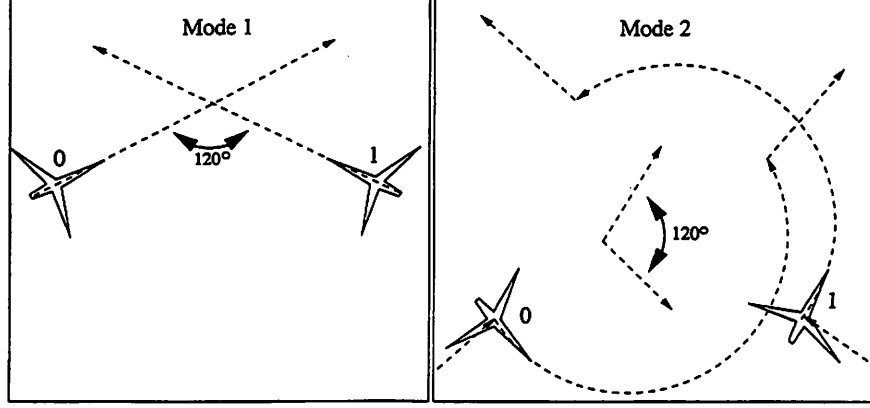


Figure 1: The two modes of operation.

Each of these techniques has computational advantages and disadvantages and may be applicable in some problems but not others. They all suffer from the “curse of dimensionality”: they typically run out of steam beyond \mathbb{R}^3 .

- Problems may arise even after a numerical solution is computed:
 - will the numerical approximation be contained in the controlled invariant set?
 - will the numerical approximation be controlled invariant itself?
 - can a controller that renders the numerical approximation controlled invariant be synthesized?
 - or at least, can a controller that renders the largest controlled invariant set invariant be synthesized using the approximation?

3 Example: Aircraft Conflict Resolution

Consider from the dissertation of C. Tomlin [2], a collision avoidance maneuver for a pair of aircraft consisting of two modes of operation: a *cruise* mode in which both aircraft follow a straight path, and an *avoid* mode in which both aircraft follow a circular arc path. When the maneuver is initiated, each aircraft turns 90° to its right and follows a half circle. Once the half circle is complete, each aircraft returns to its original heading and continues on its straight path (Figure 1).

In each mode, the continuous dynamics may be expressed in terms of the *relative motion* of the two aircraft (equivalent to fixing the origin of the relative frame on aircraft 0 and studying the motion of aircraft 1 with respect to aircraft 0):

$$\begin{aligned}
 \dot{x}_r &= -v_0 + v_1 \cos \psi_r + \omega_0 y_r \\
 \dot{y}_r &= v_1 \sin \psi_r - \omega_0 x_r \\
 \dot{\psi}_r &= \omega_1 - \omega_0
 \end{aligned} \tag{6}$$

in which $(x_r, y_r, \psi_r) \in \mathbb{R}^2 \times [-\pi, \pi]$ is the relative position and orientation of aircraft 1 with respect to aircraft 0, and v_i and ω_i are the linear and angular velocities of each aircraft. In the cruise mode $\omega_i = 0$ for $i = 0, 1$ and in the avoid mode $\omega_i = 1$ for $i = 0, 1$. The control is the linear velocity of aircraft 0, $u = v_0 \in \mathbf{U}$, and the disturbance is the linear velocity of aircraft 1, $d = v_1 \in \mathbf{D}$, where \mathbf{U} and \mathbf{D} denote the range of possible linear velocities of each aircraft. Here we restrict our attention to the case where both \mathbf{U} and \mathbf{D} are singletons (i.e. the aircraft maintain the same speed).

The discrete state takes on three possible values, $\mathbf{Q} = \{q_1, q_2, q_3\}$. q_1 corresponds to cruising before the avoid maneuver, q_2 corresponds to the avoid mode and q_3 corresponds to cruising after the avoid maneuver has been

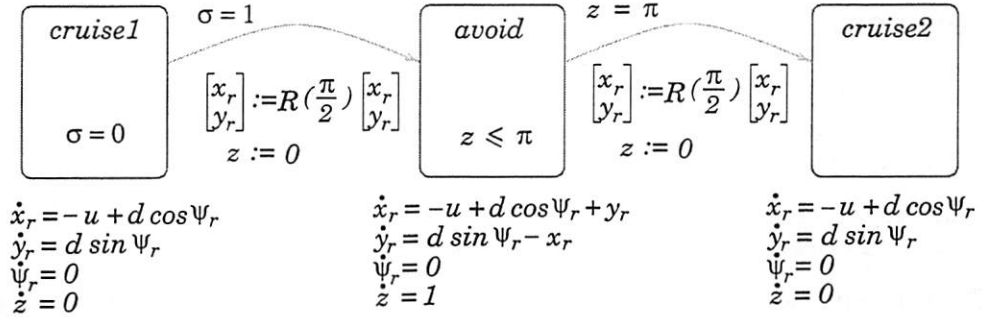


Figure 2: Hybrid dynamics of the two aircraft example

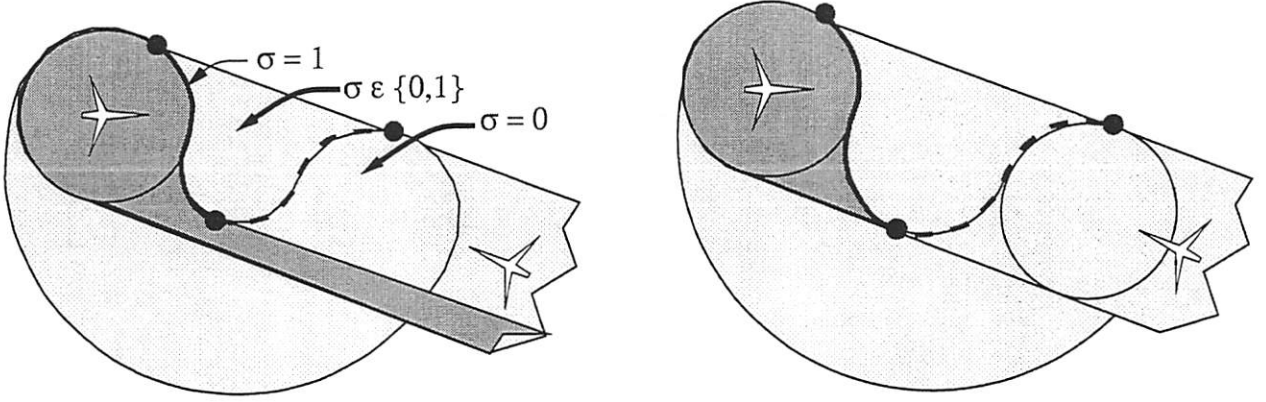


Figure 3: Enabling and forcing boundaries for σ_0 and the effect of increasing the radius of the turn.

completed. There are two transitions. The first is controllable by a discrete input variable, σ , and corresponds to the initiation of the avoid maneuver. The second transition, corresponding to the completion of the avoid maneuver, is required to take place exactly when the aircraft have completed a half circle, and is therefore treated as uncontrollable. The continuous state space is augmented with a timer $z \in \mathbb{R}$ to force this transition. We set $X = \{x_r, y_r, \psi_r, z\}$. The dynamics of the maneuver can easily be encoded by a hybrid automaton, shown pictorially in Figure 2. The maneuver is safe if the aircraft remain at least 5 nautical miles apart throughout, that is:

$$F = \{q_1, q_2, q_3\} \times \{x \mid x_r^2 + y_r^2 \geq 25\}$$

W^* can be computed following the above algorithm. The computation Reach is simplified by the fact that the continuous inputs are assumed to be constant, while the computation of Pre_u and Pre_d is simplified by the fact that there are no discrete disturbance inputs. The resulting controller for σ is illustrated in Figure 3(a). The transition $\sigma = 0$ until the continuous state reaches the dashed line; at this point the transition is enabled (σ is allowed to be either 0 or 1), and the transition to state q_2 may occur. The transition is forced to occur (by setting $\sigma = 1$) when the state reaches the solid boundary of W^* . Note that there are states which are not rendered safe by the maneuver. Indeed, in q_1 , if the initial state is in the dark region, then the aircraft are doomed to collide. Figure 3(b) displays the result of increasing the radius of the turn in q_2 . Notice that the set W^* increases as the turning radius increases. This implies that the maneuver renders a larger subset of the state space safe. Figure 3(b) shows the critical value of the turning radius, for which the maneuver is guaranteed to be safe, provided the conflict is detected early enough.

For more details on the application of hybrid systems to multi-aircraft conflict resolution see [3].

References

- [1] C. Tomlin, J. Lygeros, and S. Sastry, "Computing controllers for nonlinear hybrid systems", in *Hybrid Systems: Computation and Control*, F. W. Vaandrager and J. H. Van Schuppen, Eds., number 1569 in LNCS, pp. 238–255. Springer Verlag, 1999.
- [2] Claire Tomlin, *Hybrid Systems with Application to Air Traffic Management*, PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1998.
- [3] Claire Tomlin, George Pappas, and Shankar Sastry, "Conflict resolution in multi-agent systems: A case study in air traffic control", *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509–521, 1998.

John Lygeros, Spring 1999
Dept. of EECS, U.C. Berkeley

Problem 1: Let $x_0, x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and u a piecewise continuous function of time. Consider the matrix exponential function defined by its Taylor series expansion $e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots$, where I is the identity $n \times n$ matrix.

(a) Consider the autonomous, linear dynamical system:

$$\dot{x}(t) = Ax(t), \quad x(0) = x_0 \quad (1)$$

For an arbitrary $T > 0$ show that:

$$x(t) = e^{At}x_0 \quad (2)$$

is an execution of (1) for $t \in [0, T]$. Can there be another execution, x' , with $x'(t) \neq x(t)$ for some $t \in [0, T]$?

(b) Consider the linear control system:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0 \quad (3)$$

For an arbitrary $T > 0$ show that:

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \quad (4)$$

is an execution of (3) for $t \in [0, T]$. Can there be another execution, x' , with $x'(t) \neq x(t)$ for some $t \in [0, T]$? (Hint: Use the Leibniz rule.)

Problem 2: Consider the finite automaton, $M = (\mathbf{Q}, \Sigma, \Delta, q_0, F)$, with $\mathbf{Q} = \{p_0, p_1, p_2\}$, $\Sigma = \{a, b\}$, $\Delta = \{(p_0, a, p_1), (p_1, b, p_0), (p_1, b, p_2), (p_2, a, p_0)\}$, $q_0 = \{p_0\}$, $F = \{p_0\}$. Construct an equivalent deterministic finite automaton, M' . Your answer should include a justification of the construction, not just a picture!

(Hint: The fact that if M has $|Q|$ states, M' will in general have $2^{|Q|}$ states suggests a relation between states of M' and sets of states of M .)

Problem 3: Consider the bouncing ball system of Figure 1, where $x_1, x_2 \in \mathbb{R}$ and $c \in [0, 1]$. Assume that initially $x_1 \geq 0$.

- Write down a model of the system in the hybrid automaton modeling formalism.
- Show that the bouncing ball automaton has transverse invariants.
- Show that the bouncing ball automaton is non-blocking.
- Show that the bouncing ball automaton is deterministic.
- Show that, if $c \in [0, 1]$, all execution of the bouncing ball automaton are Zeno.

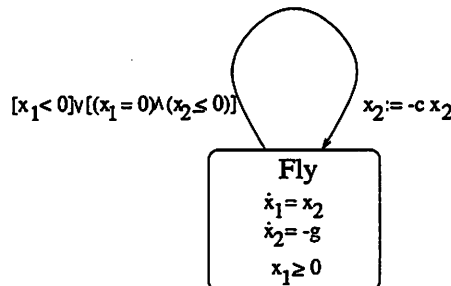


Figure 1: Bouncing ball

Problem 4: Consider a hybrid automaton, H , such that the set $\cup_{q \in \mathbf{Q}}(q, I(q))$ is open.

(a) Show that if:

1. for all $q \in \mathbf{Q}$, $I(q)^c \subseteq \cup_{(q, q') \in E} G(q, q')$; and
2. for all $(q, q') \in E$ and for all $x \in G(q, q')$, $R((q, q'), x) \neq \emptyset$,

then H is non-blocking.

(b) Show that if:

1. for all $q \in \mathbf{Q}$, $I(q)^c \supseteq \cup_{(q, q') \in E} G(q, q')$;
2. for all $(q, q'), (q, q'') \in E$ with $q' \neq q''$, $G(q, q') \cap G(q, q'') = \emptyset$; and
3. for all $(q, q') \in E$ and for all $x \in G(q, q')$, $|R((q, q'), x)| \leq 1$,

then H is deterministic.

(c) Deduce an existence and uniqueness theorem for hybrid automata with open invariant sets.

John Lygeros, Spring 1999
Dept. of EECS, U.C. Berkeley

Problem 1: Consider two compatible hybrid automata, H_1 and H_2 , and let $H = H_1 \parallel H_2$ denote their composition. Show that $\mathcal{H} = \{\chi \in \text{Hyb}(\text{Var}(H)) : \chi|_{\text{Var}(H_i)} \in \mathcal{H}_i, \text{ for } i = 1, 2\}$.

Problem 2: Show that a sequence property (W, P) is both a safety and a liveness property if and only if $P(\chi) = \text{True}$ for all $\chi \in \text{Hyb}(W)$.

Problem 3: Consider a sequence property, (W, P) , such that $\{\chi \in \text{Hyb}(W) : P(\chi) = \text{True}\} \neq \emptyset$. Show that there exist a safety property (W, P_1) and a liveness property (W, P_2) such that $P(\chi) = \text{True}$ if and only if $P_1(\chi) = \text{True}$ and $P_2(\chi) = \text{True}$.

Problem 4: Consider a hybrid automaton, $H = (Q, X, \text{Init}, f, I, E, G, R)$ with transverse invariants, and a set:

$$\text{Inv} = \{(q, x) \in Q \times X : s(q, x) \geq 0\}$$

where s is a function analytic in x . Let $\text{Inv}_q = \{x \in X : (q, x) \in \text{Inv}\}$. Assume that for all $(q, x) \in \text{Inv}$, $n_{(s,f)}(q, x) < \infty$ and let:

$$\text{Out}_{\text{Inv}} = \{(q, x) \in \text{Inv} : L_f^{n_{(s,f)}(q,x)} s(q, x) < 0\}$$

Show that if:

1. $\text{Init} \subseteq \text{Inv}$
2. For all $(q, q') \in E$ and $(q, x) \in \text{Reach}(H)$, $x \in G(q, q') \cap \text{Inv}_q \Rightarrow R(q, q', x) \subseteq \text{Inv}_{q'}$.
3. $(q, x) \in \text{Out}_{\text{Inv}} \cap \text{Reach}(H) \Rightarrow x \in \text{Out}(q)$.

then H satisfies $(\text{Var}(H), \Box \text{Inv})$ (or equivalently $\text{Reach}(H) \subseteq \text{Inv}$).

Problem 5: Let H be the water tank automaton of Lecture 5. Consider:

$$F = \{x \in \mathbb{R}^2 : x_1 \geq r_1 \wedge x_2 \geq r_2\}$$

Show that H satisfies $(X, \Box F)$. What does this tell you about the limitations of deductive proofs?

John Lygeros, Spring 1999
Dept. of EECS, U.C. Berkeley

Definition 1 (Singular Automaton) *A compact, singular, initialized, rectangular automaton (singular automaton for short!) is a rectangular automaton such that:*

1. **Compact:** for all $e \in E$, $G(e)$ and $R(e, x)$ are compact, and for all $q \in Q$, $\text{Init}(q)$ and $F(q)$ are compact, and $I(q) = X$.
2. **Singular:** for all $q \in Q$, $F(q) = F_1(q) \times \dots \times F_n(q)$ is a singleton, and for all $e \in E$, $R(e, x)$ is a singleton.
3. **Initialized:** for all $e = (q, q') \in E$, if $F_i(q) \neq F_i(q')$ then $R_i(e, x) \neq \{x_i\}$.

Definition 2 (Stopwatch Automaton) *A stopwatch automaton is a singular automaton such that for all $q \in Q$, $F_i(q) \in \{0, 1\}$.*

Definition 3 (Generalized Timed Automaton) *A generalized timed automaton is a stopwatch automaton such that for all $q \in Q$, $F_i(q) = 1$.*

Problem 1: Show that every singular automaton is bisimilar to an initialized stopwatch automaton. (Hint: as a bisimulation relation use a time scaling, as in the proof of Proposition 1, Lecture 14).

Problem 2: Show that every initialized stopwatch automaton is bisimilar to a generalized timed automaton. (Hint: introduce additional discrete states to store the values of stopwatches when they stop.) Given that generalized timed automata are bisimilar to finite automata (the bisimulation relation used for timed automata still works!), what can you conclude about the reachability problem for singular and stopwatch automata?

Problem 3: Consider the linear system:

$$\dot{x} = \begin{bmatrix} \frac{2}{3} & 0 \\ 0 & -1 \end{bmatrix} x$$

Let $Y = \{(y_1, y_2) \in \mathbb{R}^2 : y_1 = 4 \wedge y_2 = 3\}$

1. Characterize $\text{Pre}_\tau(Y)$ in terms of a first order formula in $\text{OF}_{\text{exp}}(\mathbb{R}) = \{\mathbb{R}, +, -, \cdot, <, 0, 1, \exp\}$.
2. Going through the procedure outlined in the notes, show that this formula can also be written in quantifier free form.

Problem 4: Consider the linear system:

$$\dot{x} = \begin{bmatrix} 0 & \frac{2}{3} \\ -\frac{2}{3} & 0 \end{bmatrix} x$$

Let again $Y = \{(y_1, y_2) \in \mathbb{R}^2 : y_1 = 4 \wedge y_2 = 3\}$

1. Characterize $\text{Pre}_\tau(Y)$ in terms of a first order formula in $\text{OF}_{\text{an}}(\mathbb{R}) = \{\mathbb{R}, +, -, \cdot, <, 0, 1, \{\hat{f}\}\}$. (Recall that each \hat{f} represents an analytic function, $\hat{f} : \mathbb{R} \rightarrow \mathbb{R}$ with its domain restricted to a compact set).
2. Going through the procedure outlined in the notes, show that this formula can also be written in quantifier free form.

ee291E, Spring 1999

Final Project Presentations

Monday, May 10, 1999, Room 241 Cory

- 3:10-3:25: Carlo Cloet, "Modeling and control of a copier paper-path".
- 3:25-3:40: Jianghai Hu, "Some attempt to stochastic hybrid system".
- 3:40-3:55: Jun Zhang, "Properties of Zeno Hybrid Automaton"
- 3:55-4:10: Laura Munoz, "Modeling a Vehicle String".
- 4:10-4:15: Break
- 4:15-4:30: John Koo & Bruno Sinopoli, "Flight Envelope Protection Controller Synthesis based on Finite Element Method"
- 4:30-4:45: Pedro Arroyo, "Bipedal Gait Modeling Using the Hybrid Formalism"
- 4:45-5:00: Motoyoshi Ozaki, "Game theoretic approach to drilling control"
- 5:00-5:15: Gabriel Gomes, "MLD systems: modeling and optimal control"

Wednesday, May 12, 1999, Room 241 Cory

- 3:10-3:25: Shawn Schaffert & Rene Vidal, "Maximal controlled invariant set and least restrictive controller of discrete-time linear systems"
- 3:25-3:40: Claudio Pinello, "Stability issues for the Fast Positive Force Transient Control"
- 3:40-3:55: John Musacchio, "Stability of hybrid systems with piecewise linear differential inclusions"
- 3:55-4:10: Hyounjin Kim & Shahid Rashid, "Pursuit-Evasion Game w/ Multiple Evaders and Pursuers"
- 4:10-4:15: Break
- 4:15-4:30: Ben Horowitz, "Rectangular Hybrid Games"
- 4:30-4:45: Adam Sweet, "Implementation of hybrid phenomena in the HCC simulation language"
- 4:45-5:00: Arnab Nilim, "Reachability Calculation for the Hybrid System"
- 5:00-5:15: Omid Shakernia, "Controller Synthesis for vision guided landing of a UAV"

Back to the home page of ee291E

Bipedal Gait Modeling Using the Hybrid Formalism

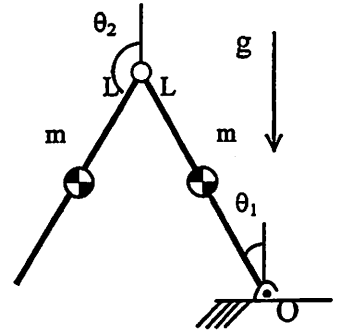
Introduction

The work detailed in this project concerns the application of the Hybrid Systems formalism in order to generate a realistic model of bipedal gait. Such models may be of utility in applications such as the design, control and analysis of autonomous bipedal robots. Potentially, these models may also find use in better understanding the mechanisms by which humans and other bipedal animals maintain stable limit cycles during the progression of their gait. To date, most of the biomechanical depiction of bipedal gait consists of disparate simple models used to attempt to verify empirical results. It is hoped that the development of a detailed physical model of controlled gait may shed some light on these issues. Also, given the potential inherent synergy between the knowledge of human walking and the study of artificial bipedal robots, this opens up the possibility of extending the research to encompass human/robotic collaborative systems. These are systems where both a robot worn about the legs and a human wearer may cooperate to achieve a specified goal.

Part of the motivation for selecting the hybrid formalism to express our gait model was the intuitive observation that the walking cycle lends itself to a hybrid model. If one considers the legs as inverted pendula, then the model of the bipedal walking apparatus may be expressed as a nonlinear continuous model with impulsive events occurring whenever a leg impacts the ground.

Hybrid Model of Bipedal Gait

A simple physical model of a pair of legs is shown at right. Essentially, each leg is modeled as a link in a double inverted pendulum. One leg is in stance, meaning it is attached to the ground, whereas the other leg is free to swing. The mechanism has two degrees of freedom, in other words, its orientation may be expressed using the two generalized coordinates θ_1 (representing the orientation of the ankle of the stance leg) and θ_2 (representing the orientation of the swing leg). In order to make controlling the device easier, we assume that the model is fully actuated. Essentially we assume we are able to apply control torques both at the ankle of the stance leg and at the hip of the device (the joint between the two legs). This is done to avoid having uncontrollable subspaces and to make the controller design easier.



The main reason such a simple physical model of the legs was used was the fact that it was hoped that the model may be used to shed physical insight into the conditions under which stable walking is achieved. Given the fact that the model only fourth order one would be able to graph the state trajectories of each leg. The equations governing the continuous dynamics of the system are:

$$M\ddot{x} + C(\dot{x}, x) + N(x) = u, \text{ where } u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, x = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}, \text{ and}$$

$$M = \begin{bmatrix} 5ml^2 & 2ml^2 \cos(\theta_1 - \theta_2) \\ 2ml^2 \cos(\theta_1 - \theta_2) & ml^2 \end{bmatrix} \quad C = \begin{bmatrix} ml^2 \sin(\theta_1 - \theta_2)(\dot{\theta}_1 \dot{\theta}_2 - \dot{\theta}_2^2) \\ ml^2 \sin(\theta_1 - \theta_2)(3\dot{\theta}_1 \dot{\theta}_2 - \dot{\theta}_1^2) \end{bmatrix}$$

$$N = g \begin{bmatrix} -3ml \sin \theta_1 \\ -ml \sin \theta_2 \end{bmatrix}$$

$$\begin{aligned}
x &= (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in X = \mathbb{R}^4 \\
Q &= \{q\}, q \in Q = \{\text{left_stance}, \text{right_stance}\} \\
\text{Init} &\subseteq Q \times X
\end{aligned}$$

The hybrid automaton consisted of a single discrete variable which corresponded to which leg was on the ground. This variable could take any of the two state valuations *left_stance* and *right_stance*. Note that there is no state which represents the condition when both feet are on the ground. In essence, we have collapsed this portion of the cycle to an instantaneous event. This was an unfortunate consequence of the simplicity of our physical model. In comparison with empirical data of a typical walk cycle, this is an oversimplification, where it has been routinely shown that the double support phase accounts for about 10% of a typical walk cycle.

One may also wonder whether it would be possible to simplify the discrete portion of the model even further, and take advantage of the symmetry between the left and right legs such that one would only have to model a single step in the cycle. It was desired to keep the model a bit general here due to the fact that Goswami, studying the stability of limit cycles for passive mechanisms of this type notes: "One curious thing happens for slightly larger slopes. There is a bifurcation of the solution and the robot exhibits a limit cycle which repeats itself every two cycles. Physically, this means that any two successive steps of the robot are not identical. A local stability analysis shows that the gait is stable." [1] Hence, in order to possibly encompass phenomena of this nature, it was decided to differentiate between the successive cycles of each leg.

Guards

The guards for this system represent collisions of the feet with the ground. Due to the fact that the physical model does not account for the lifting of the swing leg so as not to contact the ground, the condition for collision was based on the angle of the swing leg. Hence, the swing leg ignores the fact that it has penetrated the ground until it reaches a certain angle, where the model registers a collision.

$$G(q, q) = (\theta_2 = \theta_d)$$

Resets

The resets for the system remap the system state after the discrete transitions. The resets for our model of gait were physically motivated. In this case, they were derived from models of rigid body collisions for planar kinematic chains. As in the bouncing ball automaton studied early in the course, after every discrete transition the model retains memory of its position but loses memory of its velocity.

$$R(q, q, x) = (\theta_1, \theta_2, \omega_1^+(x), \omega_2^+(x))$$

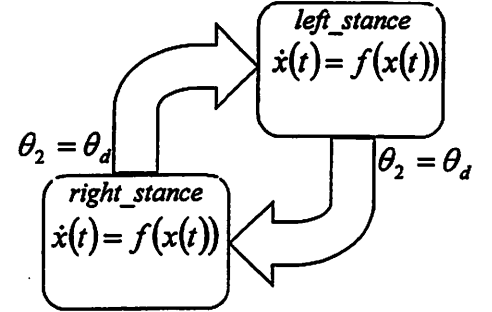
Unlike the bouncing ball automaton, however, deriving the collision response dynamics for a planar kinematic chain of rigid bodies is not a trivial exercise. The collision models were thus taken from a paper by Hurmuzlu et al. In this paper Hurmuzlu et al. model kinematic chains impacting external surfaces with one end of the chain in contact with the surface. In the paper, it is assumed that there is no vertical motion at the resting end. The model also incorporates a coefficient of restitution, which models the energy lost by the system from the collision. This is an important consideration in determining stability using the Lyapunov methods for hybrid systems, as one must not only examine the behavior of the Lyapunov function during continuous evolution, but as the system jumps from one discrete state to another as well.

An example of the velocity remapping functions derived for a perfectly elastic collision of a two link planar mechanism are as follows:

$$\omega_1^+(x) = \frac{[\omega_1 - 4\omega_1 \cos(2\theta_1 - 2\theta_2) + 2\omega_2 \cos(\theta_1 - \theta_2)]}{-3 + 2 \cos(2\theta_1 - 2\theta_2)}$$

$$\omega_2^+(x) = \frac{[-2\omega_1 \cos(2\theta_1 - 2\theta_2) + \omega_2]}{-3 + 2 \cos(2\theta_1 - 2\theta_2)}$$

A graphical depiction of the hybrid automaton is shown at right.



Gait Controller Design

Now that we have our physical model of the system it was necessary to imbue the system with the intelligence necessary to enable it to walk. We approach the problem by separately specifying the objectives for both the stance and the swing leg.

One conceivable simple model to designate the behavior of the swing leg is to have it mirror the angle of the stance leg. This condition can be expressed as $\theta_1 = \theta_2 - 180$. Physically this corresponds to the torque at the hip attempting to keep the triangle formed by the two legs isosceles. One may visualize the system walking as the two legs closing like a scissor until they perfectly overlap when they are vertical; then they open up again until the swing leg collides.

The desired behavior of the stance leg is to pivot over on the ankle to the point where the swing leg can collide with the ground for the next step. Hence, since we have specified our guards to detect a collision when $\theta_2 = \theta_d$, and given the fact that the swing leg mirrors the stance leg, we may write the desired output of our stance leg in terms of the collision angle of the swing leg.

A controller to accomplish this may be designed by specifying these conditions as outputs of the controller and having the controller attempt to drive these outputs to zero. Given this, the output equations are expressed as:

$$y_1 = \theta_2 - \theta_1 - 180$$

$$y_2 = \theta_1 - \theta_d$$

A feedback linearizing controller was then designed in order to specify the closed loop continuous dynamics of the system.

Assessment of Stability of Limit Cycles

Now that we have designed a controlled walking apparatus, it is desired to analyze certain properties of the system, most notably the stability of the gait. Different techniques may be employed to accomplish this- two are detailed in this paper.

Stability Assessment Using the Methods of Lyapunov

These methods follow from the work of Hui Ye on stability theory for hybrid dynamical systems [3]. Basically, their criterion for stability is that a Lyapunov function V is required to be monotonically decreasing everywhere except at the instants where the discrete transitions occur. At every such instantaneous transition the Lyapunov function V is only allowed to decrease.

Stability Assessment Using the Method of Poincaré Sections

Due to the fact that stable walking exhibits a limit cycle behavior in the state space, one may also classify the stability of the system using a Poincare map. This has several advantages over the aforementioned Lyapunov methods. For one thing, it is easier to discuss concepts like orbital stability of a periodic orbit of a differential equation. Rather than having to extend the Lyapunov theory of the stability of an equilibrium point to encompass limit cycles by examining the distance to the trajectory, one merely has to examine the stability of the point where the trajectory crosses the a submanifold of the state space.

What follows is a discussion of how to employ the method of Poincare sections to assess the stability of a discontinuous limit cycle.

Employing the method of Poincare sections to assess the stability of a limit cycle with a discontinuity

Due to the fact that direct application of the method of Poincare requires that $f(x)$ be continuous on X , we are only able to currently determine the stability of a walking model with a single discrete state (*stance*). This corresponds to the walker having a symmetrical pattern between his left and right legs. Possible extensions to this work may be to investigate the application of Poincare sections to systems with discontinuous dynamics.

Essentially, one is able to apply the method of Poincare sections to a system with a single discontinuity in the vector field if the discontinuity occurs as a result of the state trajectory impacting an embedded submanifold S . We then may utilize the manifold S as our Poincare section for analysis. A quasi-rigorous explanation of the methodology follows. Much of the rigorous detail is omitted for the sake of brevity.

Consider a differential equation:

$$\dot{x}(t) = f(x(t)), \text{ with a solution } \varphi^f(t, x_0)$$

Here, $f(x(t))$ is the continuous vector field, not including the discontinuity through the manifold, and $\varphi^f(t, x_0)$ is the solution of the differential equation which extends through the surface S .

Define a function $H : X \rightarrow \mathbb{R}$, such that

$$S := \{x \in X | H(x) = 0\}$$

Define a map $\Delta : S \rightarrow X$

Write the system with impuse effects as

$$\Sigma := \begin{cases} \dot{x}(t) = f(x(t)) & x^-(t) \notin S \\ x^+(t) = \Delta(x^-(t)) & x^-(t) \in S \end{cases}, \text{ hence, the delta function is basically your reset relation.}$$

Define a *time to impact* function $T_I : X \rightarrow \mathbb{R} \cup \{\infty\}$, by

$$T_I(x_0) := \begin{cases} \inf \{t \geq 0 | \varphi^f(t, x_0) \in S\} & \text{if } \exists t \ni \varphi^f(t, x_0) \in S \\ \infty & \text{otherwise} \end{cases}$$

In other words, T_i is the function that takes any point in the state space that impacts the surface and tells you how long it will take to get there.

Given this, we can then define a set

$\tilde{X} := \{x \in X \mid 0 < T_i(x) < \infty \wedge L_f H(\phi'(T_i(x), x)) \neq 0\}$ of all continuous states of the system which will collide with the surface in finite time and, once there, do not stay on the surface.

Using this set, we are able to define another set

$$\tilde{S} := \Delta^{-1}(\tilde{X})$$

We then specify the so called Poincare Return Map, $P: \tilde{S} \rightarrow \tilde{S}$, which essentially maps points on the surface to points of their first return to the surface.

$$P(x) := \phi'(T_i(\Delta(x)), \Delta(x))$$

From the return map, we may then conclude:

- a) If O is a periodic orbit of the system, then there exists a point x_o on the surface that generates O .
- b) The orbit O is stable in the sense of Lyapunov if x_o is a stable equilibrium point of $x_{k+1} = P(x_k)$.
- c) The orbit O is asymptotically stable in the sense of Lyapunov if x_o is an asymptotically stable equilibrium point of $x_{k+1} = P(x_k)$.

Possible Future Work/Extensions to the Gait Model

Due to the fact that empirical results show that the double support phase generally accounts for approximately 10% of the human bipedal walk cycle, it is felt that the modeling of the double support phase as an instantaneous event is an oversimplification. It is envisioned that future modeling efforts should focus on extension of the model to encompass this state. In order to accomplish this, however, the physical leg model would have to be made more complex. In order for the model to exhibit nontrivial continuous dynamics during the double support phase one would need to add additional degrees of freedom to the model. This would take the form of either additional prismatic or rotary joints in the leg, representing the knees of the device.

Conclusion

Due to the fact that intuitively it would seem that the dynamics of human gait lend themselves to modeling using the hybrid formalism, an attempt was made to develop a model of bipedal walking using this framework. To accomplish this, a simple model of the continuous dynamics of a pair of walking legs was first developed. Secondly, it was necessary to develop the conditions under which the impulses would occur. Due to the fact that the swing leg in our model was kinematically incapable of lifting itself to avoid collision with the ground, an angle criterion was used for inferring collision of the swing foot. Reset relations were derived from rigid body collision response models for planar kinematic chains.

Once the physics of the model was specified, a controller was designed which would enable the mechanism to walk. This was done by separately specifying objectives for both the swing and the stance leg. These objectives were encoded in terms of the desired outputs of a controller, and a feedback linearized controller was designed to regulate the system.

Lastly, some methods for assessing the stability of such a system were introduced. More specifically, both Lyapunov and Poincare methods of determining stability for hybrid systems were investigated.

References

- [1] A. Goswami, B. Espiau, and A. Keramane. Limit cycles and their stability in passive bipedal gait. In *Proc. Of the IEEE International Conference on Robotics and Automation, Minneapolis, MN.*, pages 246-251, April 1996
- [2] Y. Hurmuzlu and Tai-Heng Chang. Rigid body collisions of a special class of planar kinematic chains. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, No. 5, pages 964-971, September/October 1992
- [3] Hui Ye, Anthony N. Michel, and Ling Hou. Stability theory for hybrid dynamical systems. *IEEE Transactions on Automatic Control*, Vol. 43., No. 4, April 1998.
- [4] J.W. Grizzle, Franck Plestan, and Gabriel Abba. Poincare's Method for Systems with Impulse Effects: Application to Mechanical Biped Locomotion.

Copier Paperpath Control

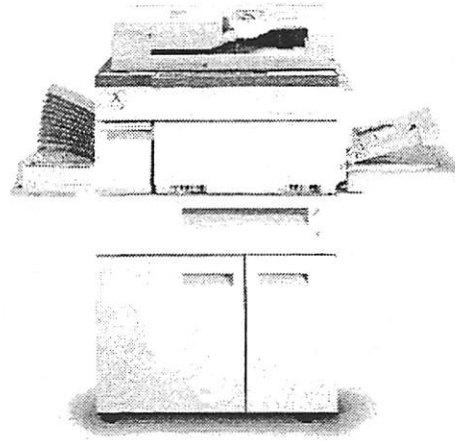
Carlo Cloet

EE 291E Presentation

May 1999



Berkeley
University of California



This class project is part of my current research on copier paperpath control.

The first part of the presentation gives the problem description, together with a high level overview of the current solution approach.

For the class project, I simulated the control strategy, using the tools Simulink and Stateflow. This is discussed in the second part of the presentation.

Problem Statement

Today's copiers operate mainly in *open loop*:

- Only small position error and alignment correction possible
- Conservative design
- Only limited range of media properties allowed
- Soft jams reduce productivity

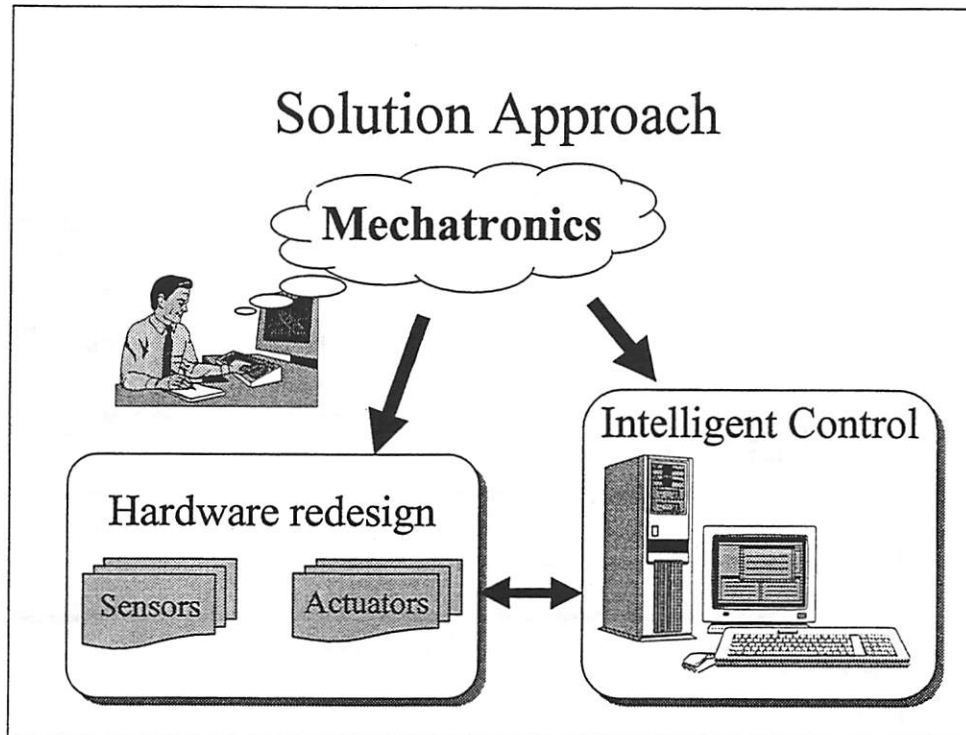
Color printing requires *high positioning accuracy*.

Enhanced features demand *more flexibility* in media handling.

How can we improve the existing technology?

Most copiers rely on standard sheet properties and certain ambient conditions. If these assumptions do not hold, the copier may perform badly. A typical example is a soft jam, which occurs when a sheet does not pass a sensor within a specified time range. This delay can be caused by a misfeed or a random disturbance along the paperpath. In that case, the machine will shut down and will ask the user to remove all sheets from the paperpath. No physical jam (= hard jam) had occurred, however.

Apart from color printing requirements, tomorrow's copiers should also be able to reliably handle recycled paper.

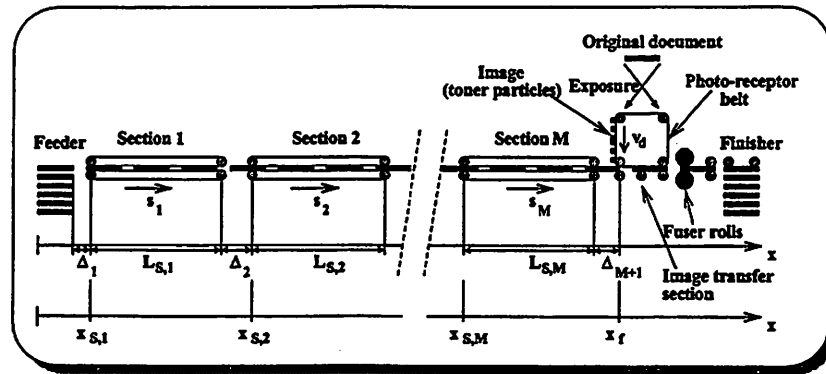


One possible approach to increase the robustness of the paperpath is by using a mechatronics mindset. Redesign the hardware by adding additional sensors and actuators. The resulting machine becomes more complex, so intelligent control is needed to manage the additional degrees of freedom and sensor information.

Solution Approach (cont.)

Hardware Redesign:

- 1-2 sections \longrightarrow multiple, *independent* sections



- Sensors: encoders, optical sensors, fax bars

The paperpath is redesigned by splitting it up in multiple sections. Each section is driven by its own motor.

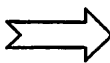
Additional optical sensors are used to detect the sheets along the paperpath.

The figure shows a typical paperpath layout. The feeder feeds sheets at regular intervals. These sheets are then transported to the image transfer section, where the image (charged toner particles) is transferred onto the sheet. Fuser rolls bake the toner onto the sheet. The finisher staples, sorts, etc...

Solution Approach (cont.)

Intelligent control to:

- match position and velocity of sheet with image at image transfer time, despite varying feeder times and disturbances along the paperpath
- avoid collisions

Combined approach 

- Increased *robustness* by allowing wider range of media under different operating conditions
- Increased *throughput*

The goal of the intelligent control strategy is to ensure timely arrival of all sheets at the image transfer section.

The sheet position and velocity must match those of its image during image transfer.

Only then will the image be transferred nicely onto the correct location on the sheet.

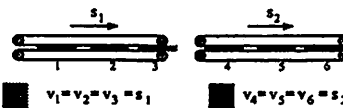
System Model: Constraints

Constraints imposed by system

- Photoreceptor belt velocity
 v_d is constant



- Section vs. sheet velocity



Constraints imposed on controller

- Synchronization during transfer



- No collision



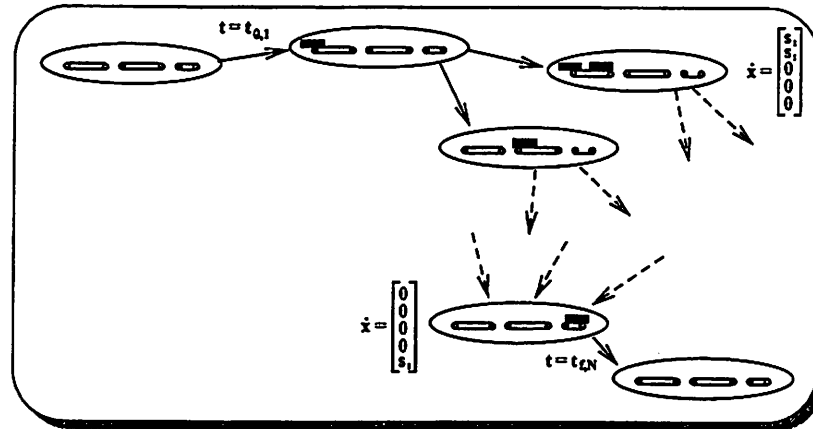
From top left, counterclockwise.

- The photoreceptor belt velocity is kept constant due to some xerographic (laser) and mechanical (inertia) reasons.
- Sheets in the same section run at equal velocity (see later).
- Two sections must synchronize during sheet transfer to prevent buckle buildup or damaging the sheet.
- Collisions are of course not an option.

System Model: Dynamics

Hybrid system

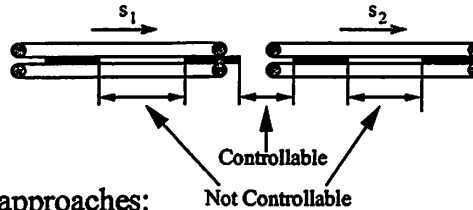
Mapping from input to state evolution changes discretely.



Let's ignore the actuator dynamics and consider the mapping from section velocities (input) to sheet velocities (state). When sheet 1 is in section 1, its velocity is determined by section 1. However, as it enters section 2, its velocity is suddenly determined by section 2 (we assume both sections synchronize during sheet transfer, as dictated by the constraints). Clearly, the mapping from input to state evolution changes discretely between different continuous dynamics or also, the system is a hybrid system.

Position versus Spacing Control

Opportunities to control are *limited*:



Two main approaches:

InterSheet Spacing Control (ISSC)

- Try to keep sheets at a desired relative spacing
- Assumes downstream sheets arrive with small or no error
- String stability issues

Absolute Reference Tracking Control (ARTC)

- Track absolute reference position
- Difficult to generate reference trajectories that satisfy constraints.
- Control of one sheet may introduce error for other sheets in the section.

Sheets that are in the same section, cannot be independently controlled. An intersheet spacing can only be changed when both sheets completely reside in different sections. Note that this

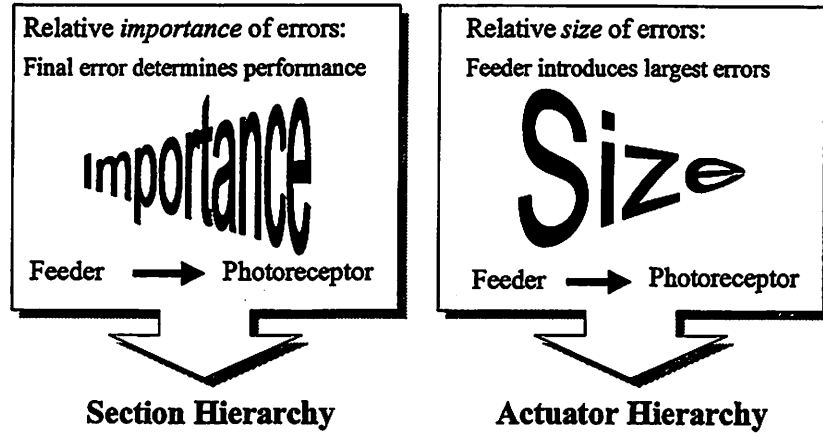
is the reason why the paperpath was split up into different sections.

ISSC tries to control the intersheet spacings. Their reference value corresponds to the spacing between images on the photoreceptor belt.

ARTC tries to control sheets by using an absolute reference trajectory for every sheet.

The approach used in this research is mainly ISSC, with ARTC for the first sheet in the copyjob and when a sheet is the leading sheet in the last section (to avoid systematic error build-up).

Position/Spacing Error Properties



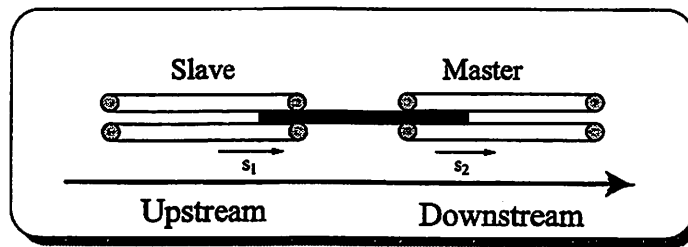
The system performance, and therefore control goal, is mainly determined by the position error of the sheet being delivered to the image transfer section. This error represents the final position error of the image on the sheet. Errors further upstream are less critical. Therefore, one assigns a larger weight to downstream errors.

This leads to section hierarchy (see next slide).

Experiments have shown that the largest errors are introduced when a sheet is being fed into the paperpath. Along the paperpath, a sheet will experience only minor disturbances due to slip and sheet bending effects. Therefore, under closed loop control, errors should typically decrease as the sheets approach the image transfer section. In the case that the disturbances along the paperpath would be of comparable size to the feeder errors, the average error size would be more or less constant and only decrease towards the very end. This will be further addressed when discussing actuator hierarchy.

Section Hierarchy

Since the error of its leading sheet is more important, the downstream section dictates the velocity that two sections will run at *during sheet transfer*. The upstream section becomes the **SLAVE** of the downstream section.



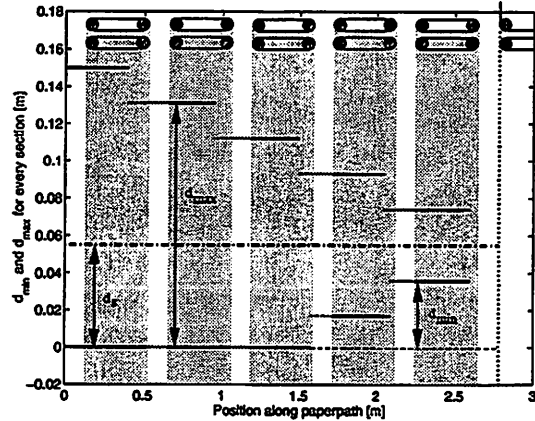
When a sheet is about to be transferred from one section to the next, both sections must start to synchronize. The common velocity is dictated by the downstream section, as the error of its leading sheet is more important. This follows from the fact the section is located closer to the image transfer section.

Actuator Hierarchy

Upstream sections are assigned *larger* acceleration and velocity limits.
This guarantees decreasing errors towards image transfer section.

Worst case reachability analysis (*dynamic game*) corresponds to expected size of errors.

Size \leftrightarrow

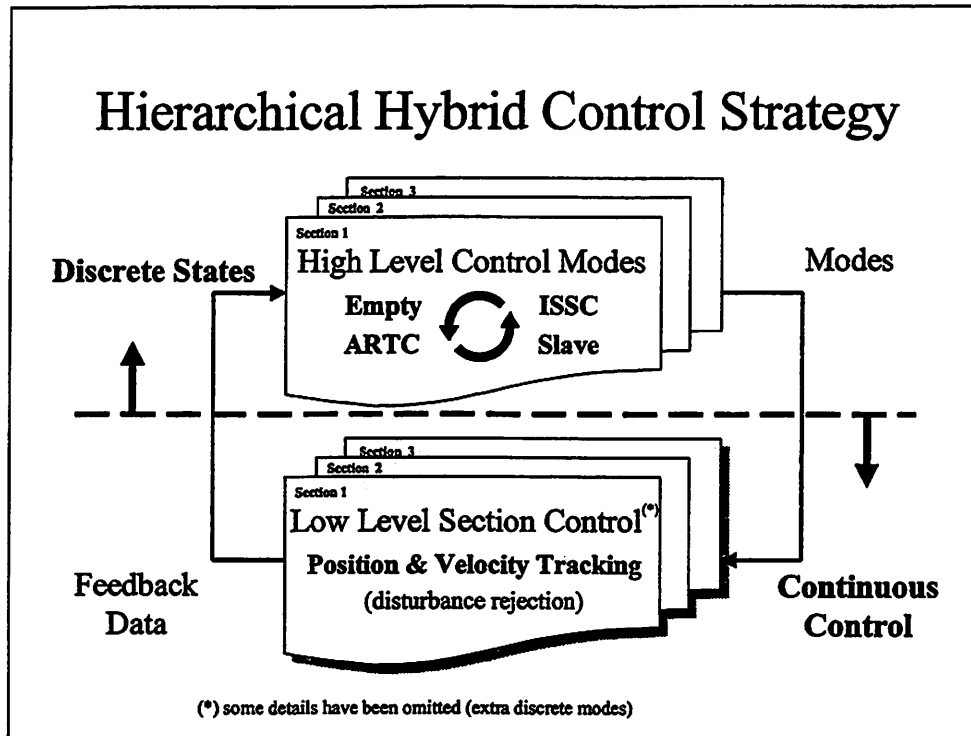


We assign larger acceleration and velocity limits to upstream sections. Assume double integrator dynamics (current controlled DC motor). A backwards reachability analysis under worst case conditions allows us to calculate the allowable spacing error in every section that can definitely be reduced to zero before the sheet in that section will reach the image transfer section.

The analysis is based on a game theoretic approach, where the downstream section is considered the adversary (it will do the worst possible action to try to increase the spacing error). The results are conservative, as the worst case scenario is unlikely to happen, but they guarantee performance independent of the action of the disturbance (downstream section). Note that we have ignored the additional (assumed small) disturbances due to slip etc.

The endresult corresponds to the expected size of the errors along the paperpath, which justifies the approach. As mentioned before, if disturbances could introduce equally large errors across the paperpath, varying the actuator capabilities as detailed above, would not be a good choice.

Note that this strategy guarantees that the spacing error will decrease when transferring a sheet from one section to the next, independent of the action of the downstream section, which is a performance guarantee.

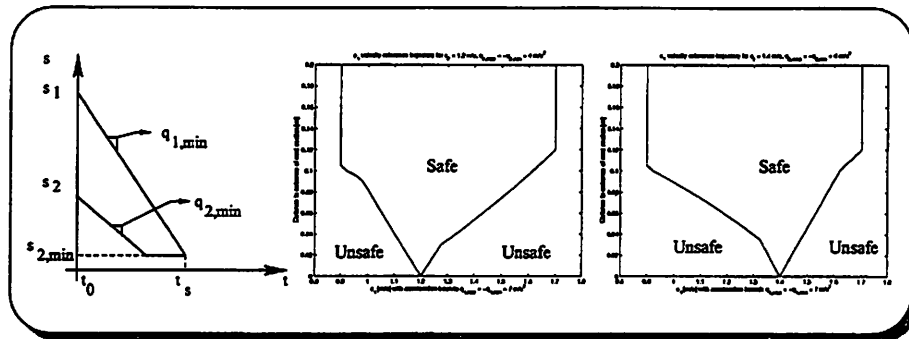


The proposed controller structure is fairly similar to the PATH project (AHS) architecture. A finite automaton steps through several high level control modes. Every high level mode corresponds to and generates setpoints for a lower level control loop (PD, PID, etc).

This corresponds to abstracting the control tasks as you move up the hierarchy.

Note that there is an extra discrete mode, which is not shown in this diagram. See the discussion on dynamic funnel tracking.

Dynamic funnel tracking



Funnel shaped limitation on section velocity guarantees section synchronization during sheet transfer.

As a sheet approaches the next section, the velocity difference between both sections must become smaller. This is due to the fact that both sections must synchronize during sheet transfer and accelerations are limited. Remember that we assume double integrator models for all sections.

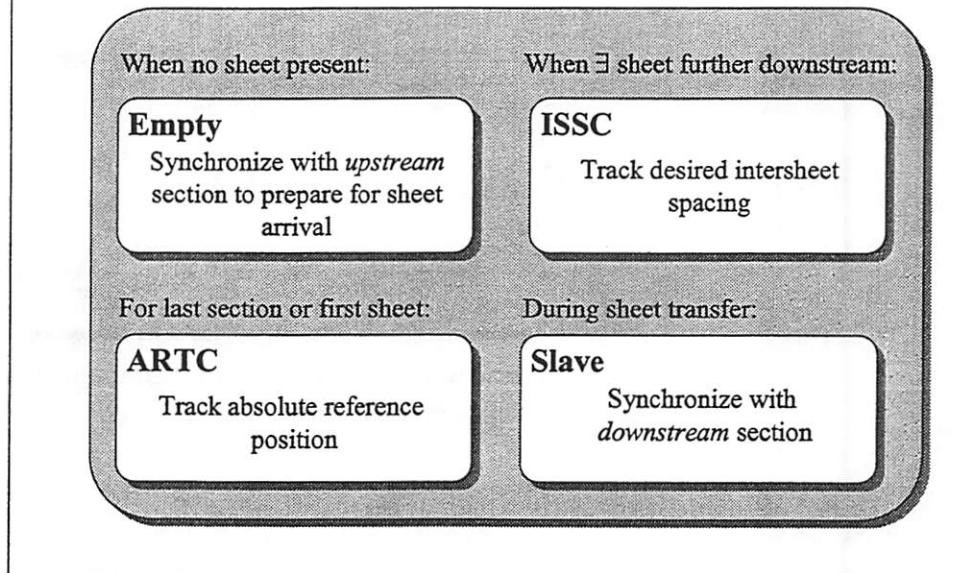
One way to approach this problem is to again assume a worst case scenario. For a given distance of the leading sheet to the entrance of the next section and the velocity of that section, one can calculate the allowable velocity bounds of the section driving the sheet.

The bounds ensure synchronization even if the downstream section starts to accelerate/decelerate at maximum pace. The derivation uses the saturation levels imposed by the actuator hierarchy. The endresult is a funnel shaped bound on velocity.

The extra mode mentioned in the previous slide, corresponds to a "funnel tracking" mode, which overwrites the control action to ensure that a section does not leave the velocity funnel. By doing so, two sections will always synchronize during sheet transfer, independent of the control action for the downstream section.

Note that the tip of the funnel equals the velocity of the downstream section.

High Level Modes



The high level modes are fairly intuitive. Note that ARTC is needed for the first sheet. Since there is no sheet in front, ISSC is not possible. Also, for the last section, we want to avoid a build-up of systematic errors, so the last section is controlled with ARTC with reference signal equal to the position of the image on the photoreceptor belt, projected onto the paperpath.

Note that the empty mode makes it easier for the upstream section to satisfy the funnel constraint.

Simulation Environment

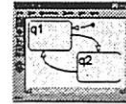
Simulink® for *continuous state evolution*

- Standard block diagram notation
- Controllers represented by enabled subsystems



Stateflow® for *discrete control logic*

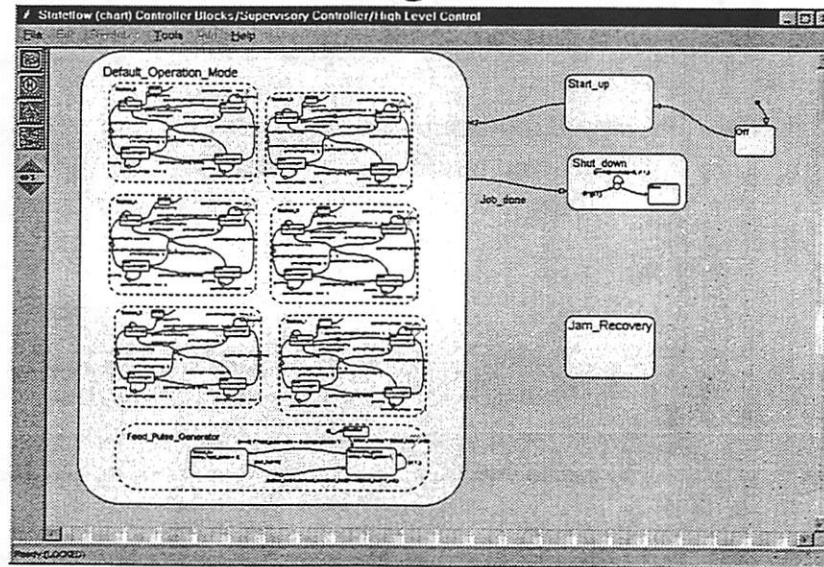
- Extension to Simulink
- Allows to graphically model finite automata
- Automaton outputs trigger enabled subsystems



The actual project work consisted of simulating the proposed control strategy, using Simulink and Stateflow.

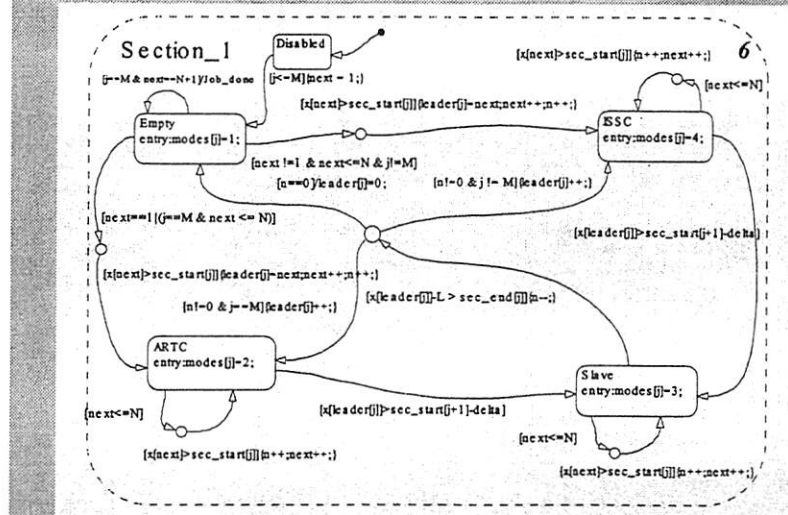
The control strategy was mainly developed during fall 98 and early spring 99. After that, it took about 2 months to read through all the Matlab manuals, set up the simulation structure and actually code it up. Apart from stateflow, the simulation uses C-MEX s-functions and the real-time workshop (RTW).

Stateflow design environment



This slide shows a typical Stateflow design window. It illustrates the high level controller for a paperpath with up to 6 sections. States, transitions, default transitions are created by simple click and drag operations. Note the state hierarchy.

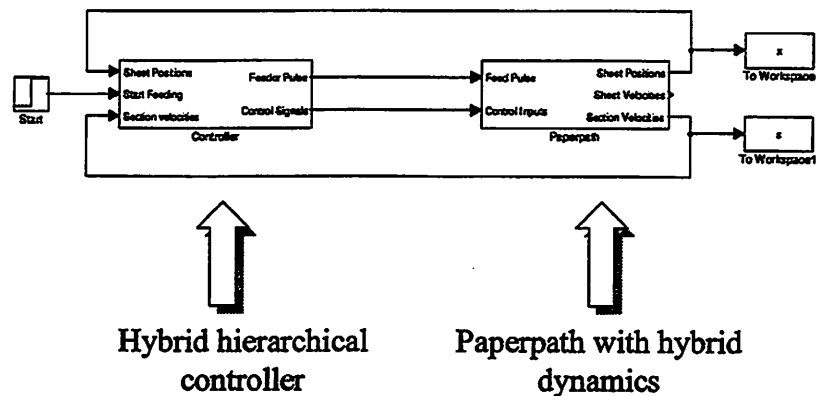
Control logic in Stateflow for single section:



This finite automaton is a zoomed in version of the previous slide. Stateflow allows to define states, transitions, events, guards, state entry actions and much more. It is very versatile. Apart from the graphical interface, one can also define local variables and input and output data/events to interact with other simulink blocks.

In the example above, $modes[j]$ corresponds to the j -th entry of the output vector "modes". Setting this to one, will enable mode j of that section by enabling an enabled subsystem in the simulink part of the simulation. This will be shown in the actual block diagrams.

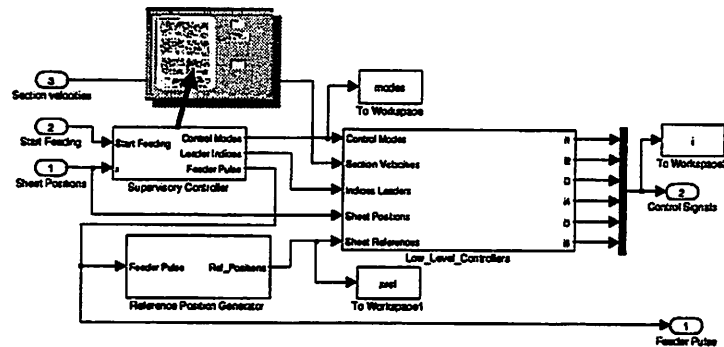
Copier Paperpath Block Diagram



Similar to stateflow, Simulink allows to hierarchically model dynamic blocks. This slide shows the highest level in the simulation. It consists of the "computer part" (the controller) and the "physical part" (the sections, feeder, sheets and image transfer section).

Note that perfect state knowledge is assumed for now.

Inside the *controller* block

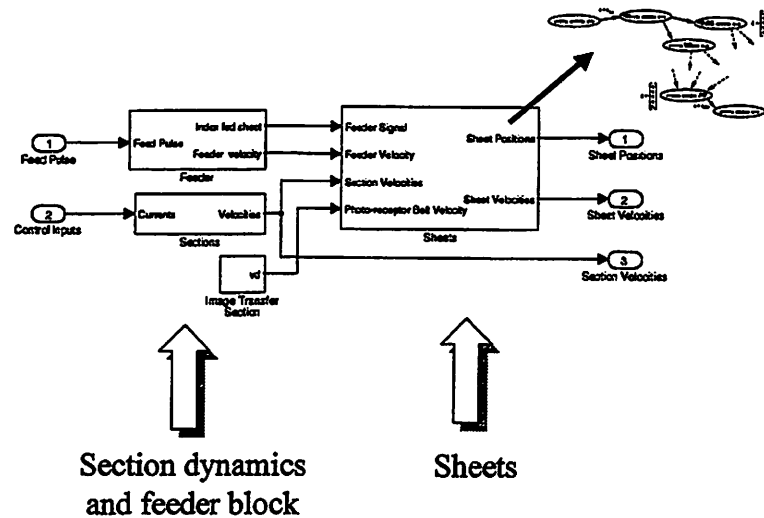


Supervisory Controller
+ Reference Generator

Low level Controllers

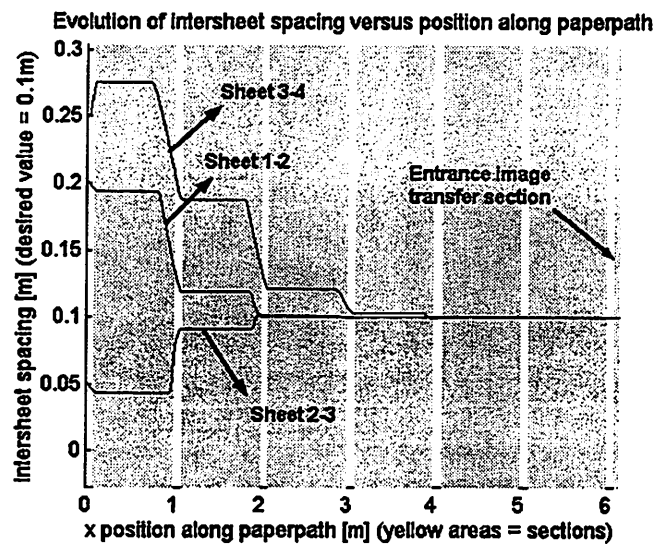
The supervisory controller block contains the stateflow diagram shown before. Note that the control modes output of the block is what steers the lower level controller blocks. The lower level controller contains velocity and position control loops and the funnel tracking control logic.

Inside the *paperpath* block



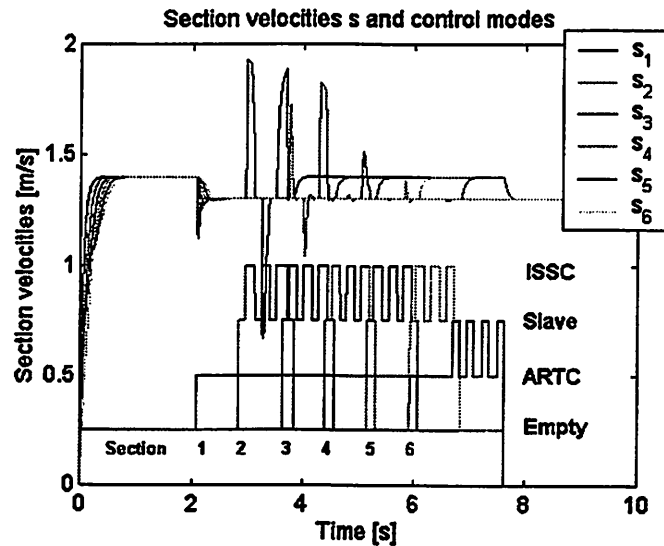
The sheets block contains another stateflow diagram that models the hybrid mapping from section to sheet velocities, as discussed before under system modeling.

Some simulation results



The plot shows the evolution of the intersheet spacings versus position along the paperpath for a copyjob consisting of 4 sheets. The desired intersheet spacing is 0.1 m. The first sheet is fed on time. Sheet 2 is fed late and its initial spacing w.r.t. sheet 1 is 0.2 m. Sheet 3 is fed too early (.05 m error) and sheet 4 is again fed too late. The simulation shows how all errors gradually reduce to zero. Note that intersheet spacing errors remain constant inside a section until the sheet becomes the first sheet of that section.

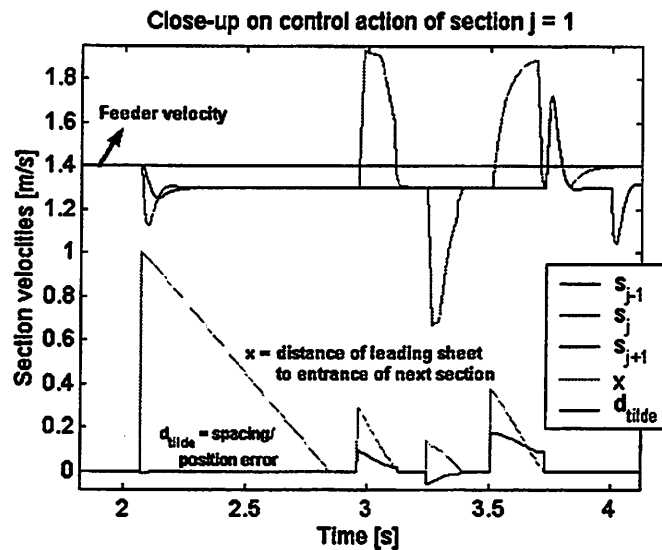
Simulation Results (cont.)



This slide gives an overview of the section velocities and section control modes for the copyjob scenario of the previous slide. All sections initially start up in the empty mode. The first sheet is fed at 2 seconds. Section 1 switches to ARTC mode (1st sheet). When section 1 delivers the sheet to section 2, it switches to slave mode and section 2 simultaneously switches to ARTC mode. Once sheet 1 leaves section 1, section 1 switches to ISSC mode, since sheet 2 is already inside the section and now becomes the leading sheet. When sheet 2 is transferred to section 2, section 1 switches again to slave mode, and so on.

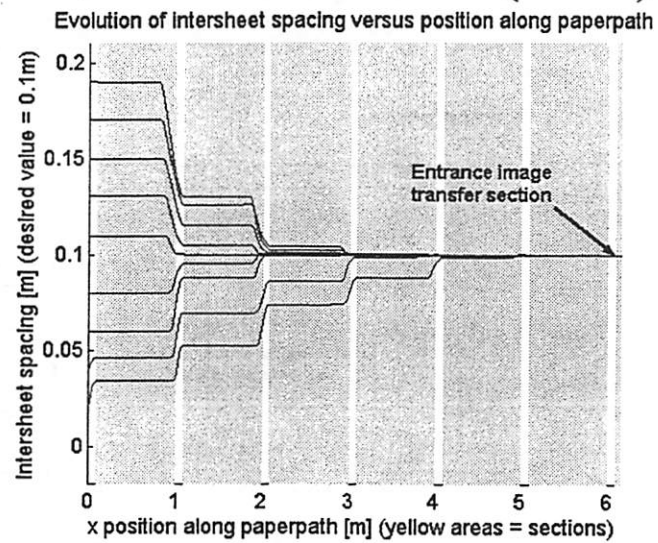
The velocity profiles show the control action to correct the spacing errors. Note that all velocities synchronize during sheet transfer.

Simulation Results (cont.)



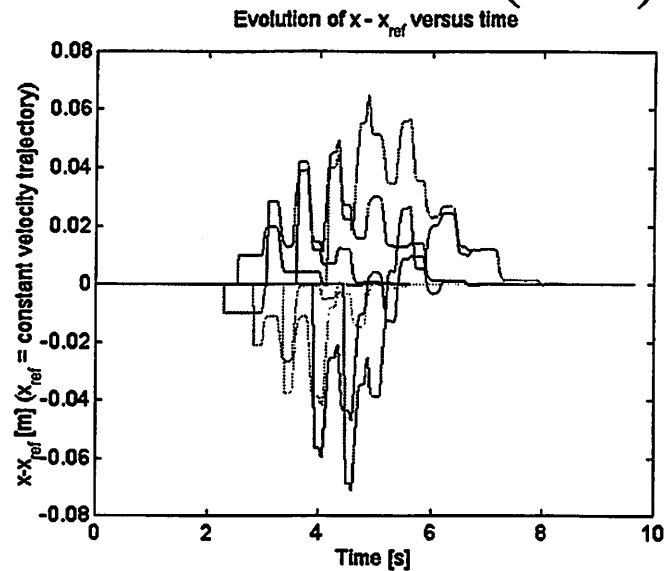
This is a close-up on the control action for section 1. The first sheet reaches the entrance of section 1 a little after $t=2$ seconds. x denotes the distance of the leading sheet in section 1 to the entrance of section 2. The four triangular shaped x profiles correspond to the 4 sheets in the copy job, since each of them is at some point the leading sheet in section 1. The bottom plot shows the evolution of the spacing error. Notice how the slope of the spacing error becomes zero when x approaches zero, due to the velocity synchronization (funnel tracking mode). The velocities of section 1 and 2 indeed synchronize when x becomes zero as shown by the plots on the top.

Simulation Results (cont.)



The control algorithm handles any copyjob complexity. The initial conditions and machine parameters determine whether all sheets make it or not.

Simulation Results (cont.)



This plot shows the deviation of the sheet positions from a straight line reference trajectory for the random copyjob shown on the previous slide. Due to the intersheet spacing control, all sheet remain at a safe distance from each other, so collisions are not likely to occur.

Future Work

- Implementation on experimental setup
- Introduce cooperation among sections
- Add observer and sensor integration
- Improve dynamical model (currently $1/s^2$)
- String stability issues may surface
- Jam clearance

* An experimental setup, consisting of a paperpath loop with three sections, will be used to evaluate the control strategy in practice.

* The algorithm does not allow sections to cooperate. This means that a section only considers the error of its leading sheet and does not worry about how upstream sections are doing. A variation with cooperation has been developed and submitted to CDC 99.

* The current simulation results were based on perfect state knowledge. In practice, we will have to rely on an observer.

* The double integrator model should be extended to include the dynamics introduced by timing belts and friction.

* String stability issues could become an issue when using more complex models. This would have to be handled by the supervisory controller, who has access to all variables in the machine.

* A far-future goal is to introduce jam clearance and according job rescheduling in the paperpath. Jams could be cleared for example by driving a sheet backwards, making it exit and speeding up its upstream neighbors.

Acknowledgements

This work is a joint effort with

Martin Krucinski
Prof. Masayoshi Tomizuka
Prof. Roberto Horowitz

and is sponsored by

National Science Foundation
Xerox

Department of Electrical Engineering, UCB
Submitted to : Dr. John Lygeros
Final project report for EECS 291e

Modeling and control of mixed logical dynamical systems

Based on :

**“Control of systems integrating logic,
dynamics and constraints”**

**by Alberto Bemporad
Manfred Morari**

**Gabriel Gomes
5/99**

Overview

In the article by Bemporad and Morari on which I have based my final project, the authors describe the use of *mixed logical dynamical* models for systems whose description may include logical rules. These models make use of transformations of the logic into linear equalities and inequalities involving binary variables, which is a convenient structure for applying *mixed integer quadratic programming* techniques to find optimal trajectories.

In this report I review some of the main points in the article and give a numerical example in which the technique is applied to optimal control of a 'lunar lander'. As most of the article was described in the oral presentation, I place more emphasis here on the example.

Preliminaries

The types of systems of interest to the authors are those whose operation is influenced by interacting dynamics and logical rules; e.g. on/off switches, gear selectors, if then else rules, etc. Following the general hybrid systems modeling paradigm presented in this course, these systems may be modeled by appropriate definition of discrete states and guards. In this paper however, an alternative approach is investigated in which the transition logic is converted into a set of linear inequalities and the dynamic relations associated with each discrete state are reduced to a single equation.

The result is a mixed logical dynamical model (MLD) which is composed of linear dynamic and output equations and a set of mixed integer inequalities. In this form, existing and efficient numerical optimization procedures can be applied, which constitutes the main advantage of the technique. The optimization problem presented in the article is termed the mixed integer quadratic programming problem (MIQP), as it involves both continuous and integer variables and has a quadratic cost function subject to linear constraints. The authors comment that

the MIQP problem is widely investigated, and that there exist various algorithms for finding solutions. In general, these algorithms can be classified into four groups:

- *Cutting plane methods*: which make use of additional constraints or "cuts" to reduce the feasible domain.
- *Decompositional methods*: employing variable partitioning, duality and relaxation methods
- *Logic-based method*: using disjunctive constraints
- *Branch and Bound methods*: via construction of a binary tree and branch elimination. This method is generally considered the most succesful for solving mixed integer linear programs, and is the one used by lp_solve 2.3, which is the software used in the 'lunar lander' example.

The MLD modeling method, coupled with predictive control strategies, is a popular approach in process control applications, due to the mentioned numerical advantages and because it readily permits incorporation of soft constraints and heuristic rules. The authors identify 6 types of systems which are appropriate for MLD modeling:

- Linear hybrid systems
- Finite state machines
- Certain nonlinear dynamic systems
- Certain discrete event systems
- Constrained linear systems
- General linear systems

Concepts in propositional calculus

In section 2 of the article, some background on propositional calculus is given. A good source on this topic is H.P. Williams' "Model Building in Mathematical Programming" in which a couple chapters are dedicated to the use of logical variables in integer programming. Three main concpets are defined:

- *Literal*: A statement which is assigned a "truth value"; either true or false. e.g. " $x > 0$ " ; "temperature is hot"
- *Connective*: operators which take one or two literals and form a new literal with a truth value prescribed by a truth table.
- *Logical variable*: A binary variable ($\delta \in \{0,1\}$) related to a literal X , which takes the value 1 if X is true and 0 if X is false.

With the use of connectives, literals can be combined to form compound literals. As will be seen, compound literals are used to encode logic rules. Although no formal procedure of operator elimination is given, a few easily verifiable examples illustrate the translation of compound literals into linear inequalities.

$X_1 \vee X_2$	translates into	$\delta_1 + \delta_2 \geq 1$
$X_1 \wedge X_2$	translates into	$\delta_1 = 1, \delta_2 = 1$
$X_1 \rightarrow X_2$	translates into	$\delta_1 - \delta_2 \leq 0$
$X_1 \leftrightarrow X_2$	translates into	$\delta_1 - \delta_2 = 0$

This technique is now employed to translate statements which involve variables of the continuous dynamics. In general, statements may contain both continuous and logical variables and we would like to represent them in a similar manner as was previously shown. Again, the approach is illustrated by example:

$[f(x) \leq 0] \wedge [\delta = 1]$	translates into	$f(x) - \delta \leq -1 + m(1 - \delta)$
$[f(x) \leq 0] \vee [\delta = 1]$	translates into	$f(x) \leq M\delta$
$\sim [f(x) \leq 0]$	translates into	$f(x) \geq \varepsilon$
$[f(x) \leq 0] \rightarrow [\delta = 1]$	translates into	$f(x) \geq \varepsilon + (m - \varepsilon)\delta$
$[f(x) \leq 0] \leftrightarrow [\delta = 1]$	translates into	$\begin{cases} f(x) \leq M(1 - \delta) \\ f(x) \geq \varepsilon + (m - \varepsilon)\delta \end{cases}$

Here, the continuous variables (x) appear in literals of the form:

$$[f(x) \leq 0]$$

where $f(x)$ is a linear function and possesses over and underestimates: M and m . More general functions, such as nonlinear functions and functions of the form $f(x, \delta)$ are not mentioned.

Auxiliary variables

In order to preserve linearity, it is necessary to eliminate products of logical variables ($\delta_1 \delta_2$) and of logical and continuous variables ($f(x) \delta$). This is done with the aid of auxiliary variables which replace the product and are defined by literals as shown:

- Logic \times logic: $\delta_3 \equiv \delta_1 \delta_2$

$$[\delta_3 = 1] \leftrightarrow [\delta_1 = 1] \wedge [\delta_2 = 1]$$

$$\text{is equivalent to } \begin{cases} -\delta_1 + \delta_3 \leq 0 \\ -\delta_2 + \delta_3 \leq 0 \\ \delta_1 + \delta_2 - \delta_3 \leq 1 \end{cases}$$

- Continuous \times logic: $z \equiv \delta_1 f(x)$

$$[\delta_1 = 0] \rightarrow [z = 0] \wedge [\delta_1 = 1] \rightarrow [z = f(x)]$$

$$\text{is equivalent to } \begin{cases} z \leq M\delta_1 \\ z \geq m\delta_1 \\ z \leq f(x) - m(1 - \delta_1) \\ z \geq f(x) - M(1 - \delta_1) \end{cases}$$

Hence, the general form of a MLD system is:

$x(t+1) = A_t x(t) + B_{1t} u(t) + B_{2t} \delta(t) + B_{3t} z(t)$...dynamic equation
$y(t) = C_t x(t) + D_{1t} u(t) + D_{2t} \delta(t) + D_{3t} z(t)$...output equation
$E_{2t} \delta(t) + E_{3t} z(t) \leq E_{1t} u(t) + E_{4t} x(t) + E_{5t}$...inequalities

where t is a rational, $x(t)$ is the state vector, $y(t)$ is the output vector, $u(t)$ is an input vector, and $\delta(t)$ and $z(t)$ are auxiliary logical and continuous variables.

It is noted that, given $x(t)$ and $u(t)$, $\delta(t)$ and $z(t)$ may not be uniquely defined by the set of inequalities. When modeling a determinate system however, we would like $x(t+1)$ and $y(t)$ to be uniquely given by $x(t)$ and $u(t)$. This motivates the concept of a "well posed system", as one in which $x(t+1)$ and $y(t)$ are uniquely prescribed by $x(t)$ and $u(t)$. This definition implies that, for a well posed system, any indeterminate components of $\delta(t)$ and $z(t)$ bear no influence on $x(t+1)$ and $y(t)$.

Next, the authors develop a series of examples illustrating different systems which can be represented as MLDs. The examples are fairly detailed and include: systems with piece-wise linear dynamics, piece-wise linear outputs, discrete inputs, qualitative outputs, bilinear systems, and finite state machines.

In section 4 of the paper, the concept of stability is defined for MLD systems. A state and input are said to be an equilibrium pair (x_e, u_e) if

$$x(t, t_0, x_e, u_e) = x_e \quad \forall t \geq t_0$$

where the notation denotes the state at time t which has been applied the control u_e from an initial condition x_e at t_0 . A standard ϵ - δ definition of stability of the equilibrium pair is also given. The point to notice is that the auxiliary and logic variables (δ and z) do not appear in these definitions. It is therefore conceivable that, due to the nonlinear nature of the dependence of $\delta(t)$ and $z(t)$ on $x(t)$ and $u(t)$, that they may continue to oscillate as the state $x(t)$ converges. This behavior may become an important issue when selecting a cost for the

optimal control problem, since inclusion of such an auxiliary variable may overwhelm the cost upon convergence of other terms.

Application example: Modeling a Lunar Lander

The lunar lander is an example of a dynamical system, which only accepts inputs from a finite set of values. The lander is equipped with three thrusters, two horizontal and one vertical. These can either be 'on' or 'off' and it is assumed that both horizontal thrusters cannot be 'on' simultaneously. The lander is placed in a vertical gravitational field in which it is always free to move in any direction without crashing. The values of the momentum provided by the horizontal and vertical thrusts are denoted T_x and T_h respectively. The goal here is to apply the MLD modeling paradigm to represent the dynamics of the lander and to use this model to find optimal trajectories.

The discrete time dynamical model for this system is:

$$\begin{cases} V_x(t+1) = V_x(t) + u_1(t) \\ x(t+1) = x(t) + V_x(t) \\ V_h(t+1) = V_h(t) + u_2(t) - g \\ h(t+1) = h(t) + V_h(t) \end{cases} \quad \text{with} \quad \begin{cases} u_1(t) \in \{-T_x, 0, T_x\} \\ u_2(t) \in \{0, T_h\} \end{cases}$$

where the state $[V_x \times V_h \times h]$ is the horizontal velocity and position and the vertical velocity and position.

First, to denote the on/off condition of the thrusts, define logical variables: $\delta_1, \delta_2, \delta_3, \alpha_1$

$$\begin{array}{ll} [\delta_1=1] \rightarrow [z_1 = 0] & \dots \text{right thrust on} \\ [\delta_2=1] \rightarrow [z_1 = T_x] & \dots \text{thrusters off} \\ [\delta_3=1] \rightarrow [z_1 = 2T_x] & \dots \text{left thrust on} \\ [\alpha_1=1] \rightarrow [z_2 = T_h] & \dots \text{vertical thrust on} \\ [\alpha_1=0] \rightarrow [z_2 = 0] & \dots \text{vertical thrust off} \end{array}$$

Here I have used two different methods for encoding the same type of logic rule. For the horizontal thrusts, a logical variable was defined for each condition. The fact that only one of these is permitted at any given time requires an additional rule:

$$\oplus \delta_i$$

On the other hand, the two conditions for the vertical thrust are represented with a single logical variable and the exclusive or is removed. This reduces the number of variables in the system but with the loss of an equality constraint which may help the numerical algorithm. In general, the choice of logic representation may depend on the problem as well as on the numerical method.

Also, the values taken by z_1 range from 0 to $2T_x$ and not from $-T_x$ to T_x as u_1 . This is because the MILP package used to obtain solutions only considers positive values. The shift is accounted for in the new MLD equation.

Upon substitution of the auxiliary variables, the linear dynamic equation becomes:

$$\begin{cases} V_x(t+1) = V_x(t) + z_1(t) - T_x \\ x(t+1) = x(t) + V_x(t) \\ V_h(t+1) = V_h(t) + z_2(t) - g \\ h(t+1) = h(t) + V_h(t) \end{cases}$$

subject to the following linear inequalities:

$$\begin{aligned} z_1 &\leq (\delta_1 - 1) (-2T_x) \\ z_1 &\geq T_x + (\delta_2 - 1) (T_x) \\ z_1 &\leq T_x + (\delta_2 - 1) (-T_x) \\ z_1 &\geq 2T_x + (\delta_3 - 1) (2T_x) \\ z_2 &\geq T_h + (\alpha_1 - 1) (T_h) \\ z_2 &\leq T_h + (\alpha_1 - 1) (T_h) \end{aligned}$$

$$\delta_1 + \delta_2 + \delta_3 = 1$$

Optimal control of MLD's

The optimal control problem considered in this article is the mixed integer quadratic programming problem, stated as follows:

Given an initial condition $x(0)$ and a final time T , find the control sequence $u(t)$ which minimizes

$$J(u_o, x_o) = \sum_{t=0}^T \|u(t) - u_f\|_{Q_1}^2 + \|\delta(t) - \delta_f\|_{Q_2}^2 + \\ \|z(t) - z_f\|_{Q_3}^2 + \|x(t) - x_f\|_{Q_4}^2 + \|y(t) - y_f\|_{Q_5}^2$$

subject to $x(T) = x_f$ and a set of linear mixed integer inequality constraints.

In our application example, the objective of the optimal control problem is to transfer the lander from an initial to a final state (position and velocity) with minimum fuel consumption. The final time is specified (T) and the cost function is defined as:

$$J = \sum_{i=0}^{T-1} (i+1)(\delta_1(i) + \delta_3(i) + \alpha_1(i))$$

The inclusion of the weight $(i+1)$ is an attempt to make the lander reach its destination as quickly as possible by giving preference to earlier thrusts. This is in part in anticipation of the predictive control problem in which only the first few control commands will be applied. The fuel consumption cost used here is linear, which makes the problem much easier to solve than the general MIQP problem. Notice however that, since the only variables included in the cost are logical variables, the problem is in fact a MIQP.

The terminal state constraint can be expressed explicitly by:

$$\underline{X}(T) = A^T \underline{X}(0) + \sum_{i=0}^{T-1} A^i \{B_1 u(T-1-i) + B_2 \delta(T-1-i) + B_3 z(T-1-i)\}$$

$$\underline{X}(T) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T \underline{X}(0) + \sum_{i=0}^{T-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^i \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_1(T-1-i) \\ z_2(T-1-i) \end{bmatrix} \right\} + \begin{bmatrix} -T_x \\ 0 \\ -g \\ 0 \end{bmatrix}$$

The terms A^T and A^i have simple compact forms:

$$\underline{X}(T) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ T & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & T & 1 \end{bmatrix} \underline{X}(0) + \sum_{i=0}^{T-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ i & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & i & 1 \end{bmatrix} \begin{bmatrix} z_1(T-1-i) - T_x \\ 0 \\ z_2(T-1-i) - g \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} V_x(T) \\ x(T) \\ V_h(T) \\ h(T) \end{bmatrix} = \begin{bmatrix} V_x(0) \\ x(0) \\ V_h(0) \\ h(0) \end{bmatrix} + \sum \begin{bmatrix} z_1(T-1-i) - T_x \\ i(z_1(T-1-i) - T_x) \\ z_2(T-1-i) - g \\ i(z_2(T-1-i) - g) \end{bmatrix}$$

$$\begin{cases} V_x(T) - V_x(0) + TT_x = z_1(0) + \dots + z_1(T-1) \\ x(T) - x(0) - V_x(0)T + \frac{1}{2}T_x T(T-1) = (T-1)z_1(0) + (T-2)z_1(1) + \dots \\ V_h(T) - V_h(0) + TT_h = z_2(0) + \dots + z_2(T-1) \\ h(T) - h(0) - V_h(0)T + \frac{1}{2}T_h T(T-1) = (T-1)z_2(0) + (T-2)z_2(1) + \dots \end{cases}$$

These 4 equality constraints imply 4 necessary conditions for existence of a solution to the problem. For example, in the first equation, the terms on the right hand side are all multiples of T_x . On the left-hand side, TT_x is also a multiple of T_x . Therefore, it is required that $V_x(T) - V_x(0)$ be a multiple of T_x :

$\therefore \text{mod}(V_x(T) - V_x(0), T_x) = 0$ is a necessary existence condition.

Similarly,

$$\text{mod}(x(T) - x(0) - TV_x(0), T_x) = 0$$

$$\text{mod}(V_h(T) - V_h(0) + Tg, T_h) = 0$$

$$\text{mod}(h(T) - h(0) - TV_h(0), T_h) = 0$$

are necessary existence conditions.

These conditions however are not sufficient. They can be used to construct a map of reachable states from a given initial state. The missing condition to assure feasibility is that T be an appropriate value.

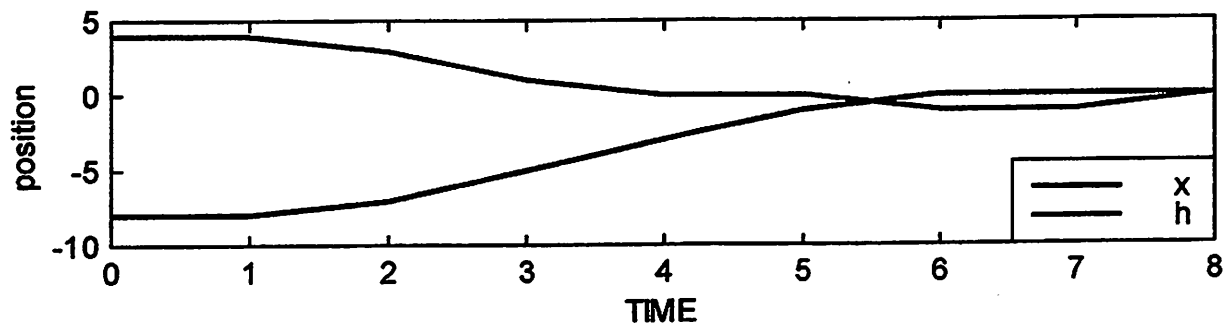
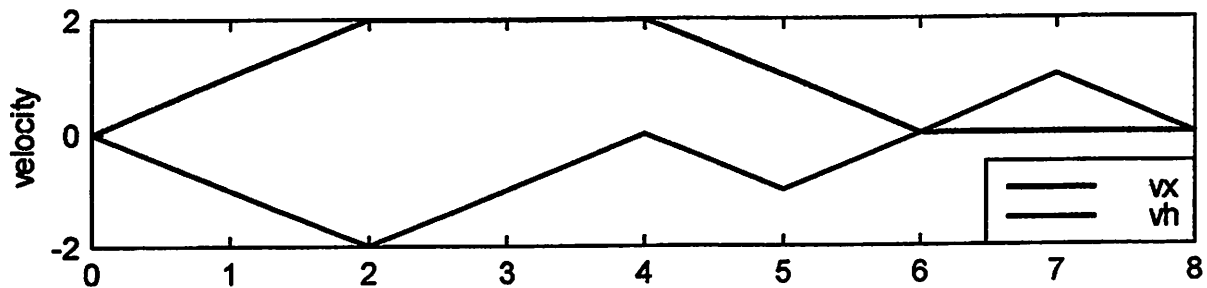
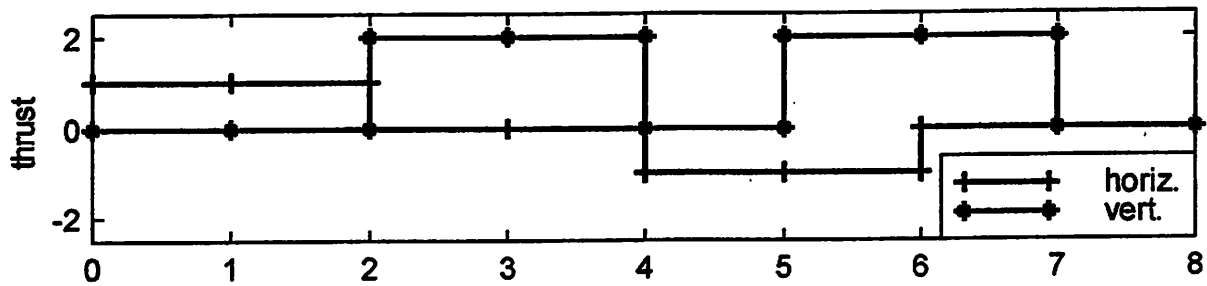
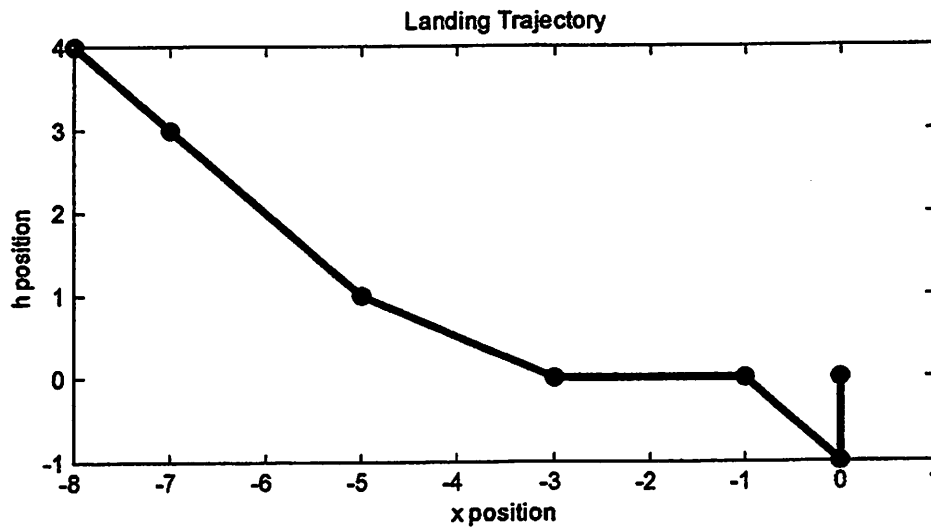
SOLUTION SOFTWARE

To solve the problem, I used `lp_solve 2.3` (Author: Michael Berkelaar) which is a publicly available linear mixed integer program solver. The software is C-based and takes input files written in the MPS format. The coding was done using MATLAB, with the aid of `lpmex` (Author: Jeffrey Kantor), which is a set of routines designed to interface MATLAB with `lp_solve` by constructing the MPS file, calling the solver and interpreting the output files. The code can be found in the appendix.

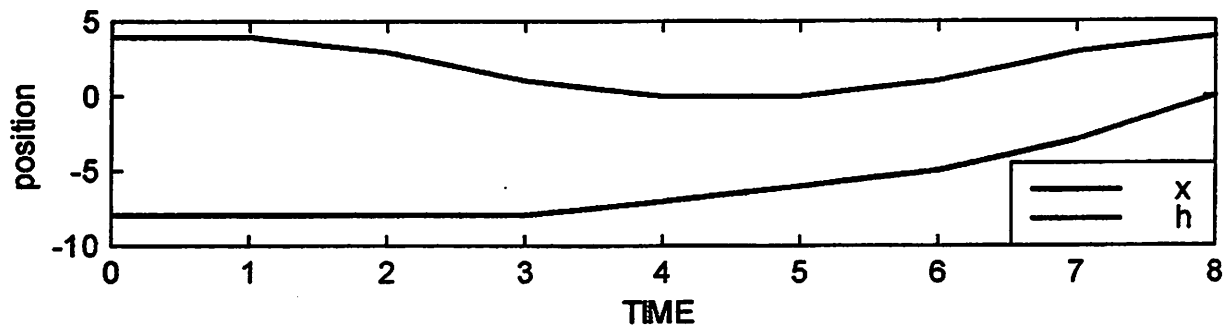
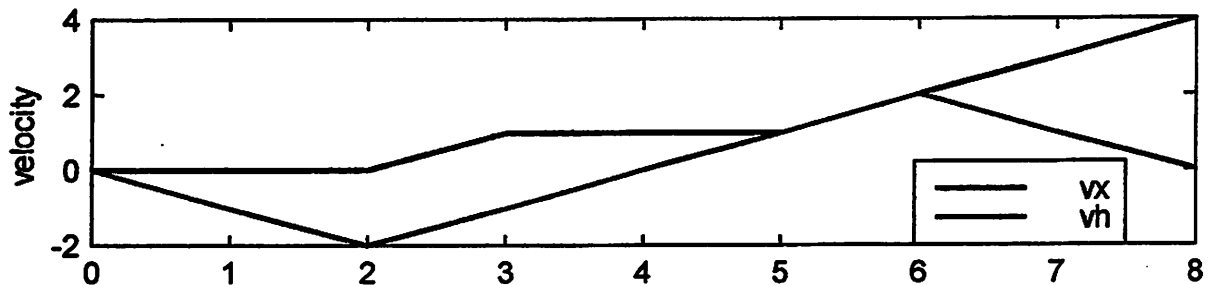
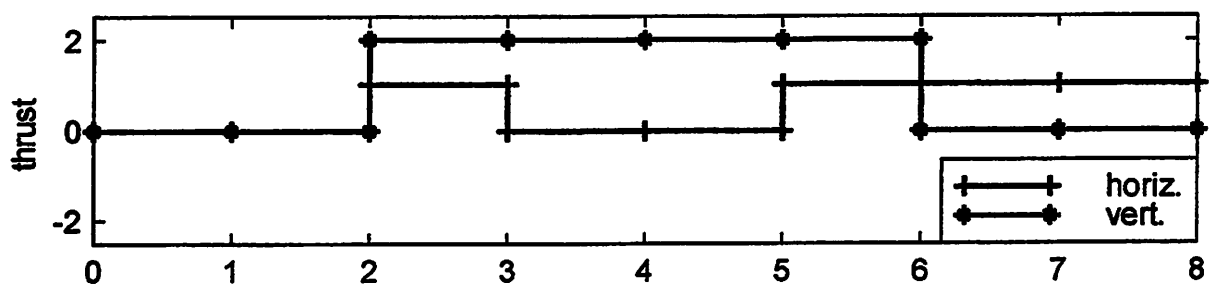
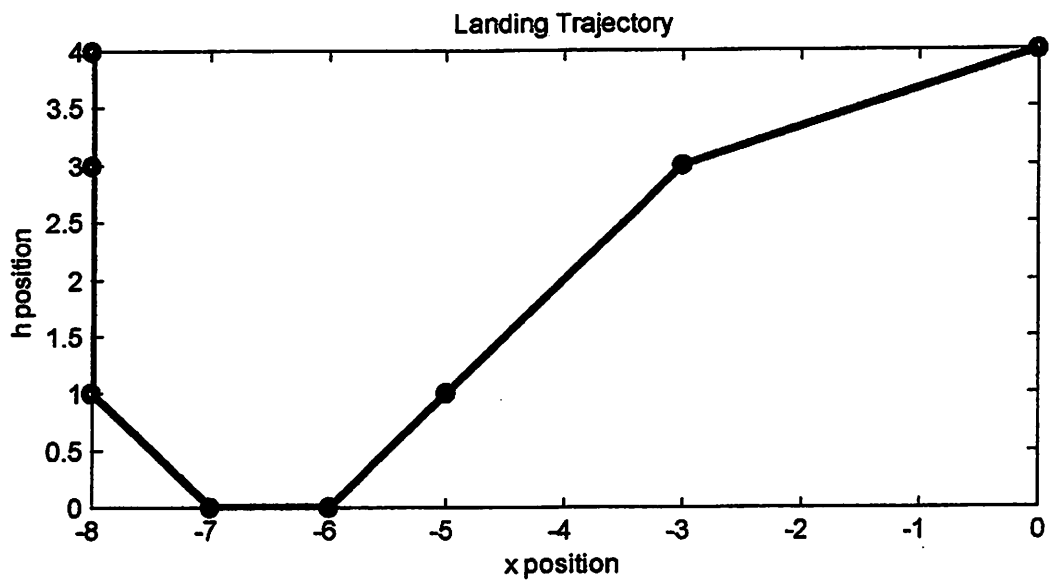
Solution example

Following are some example solutions to the optimal control problem:

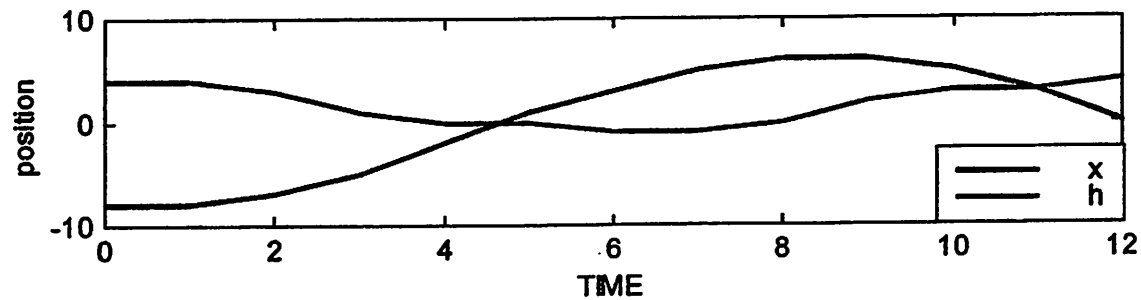
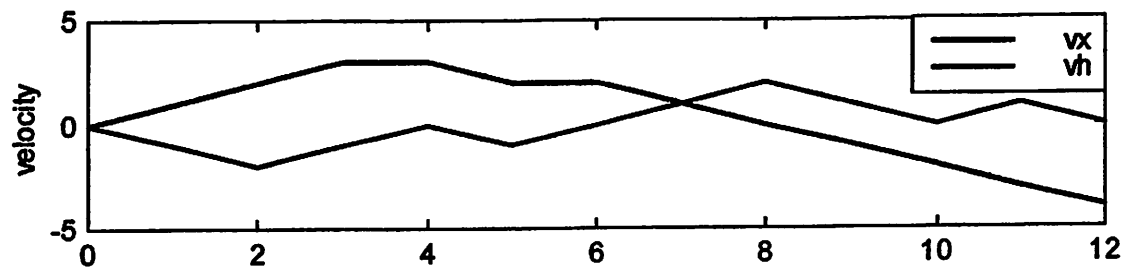
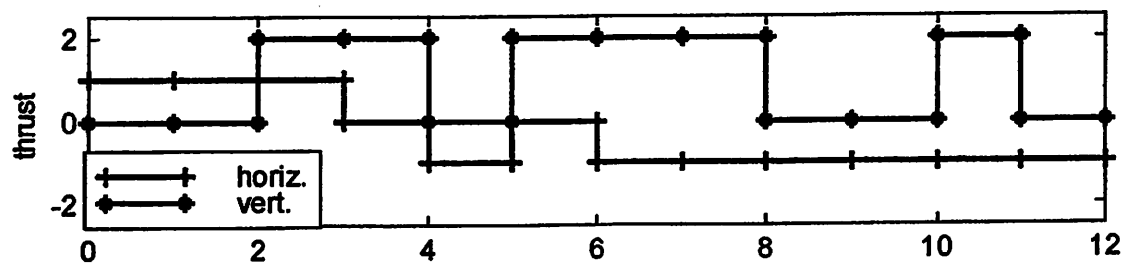
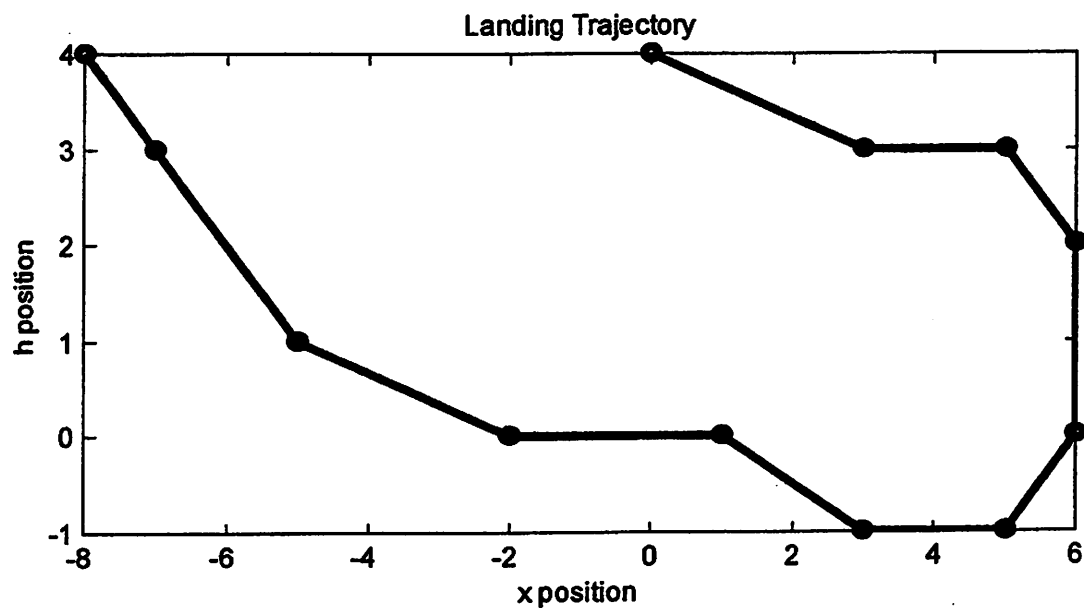
Example 1) $\underline{X}(0) = [0 \ -8 \ 0 \ 4]$ $\underline{X}(T) = [0 \ 0 \ 0 \ 0]$ $T = 8$



Example 2) $\underline{X}(0) = [0 \ -8 \ 0 \ 4]$ $\underline{X}(T) = [4 \ 0 \ 0 \ 4]$ $T = 8$



Example 3) $\underline{X}(0) = [0 \ -8 \ 0 \ 4]$ $\underline{X}(T) = [-4 \ 0 \ 0 \ 4]$ $T = 12$



Predictive control

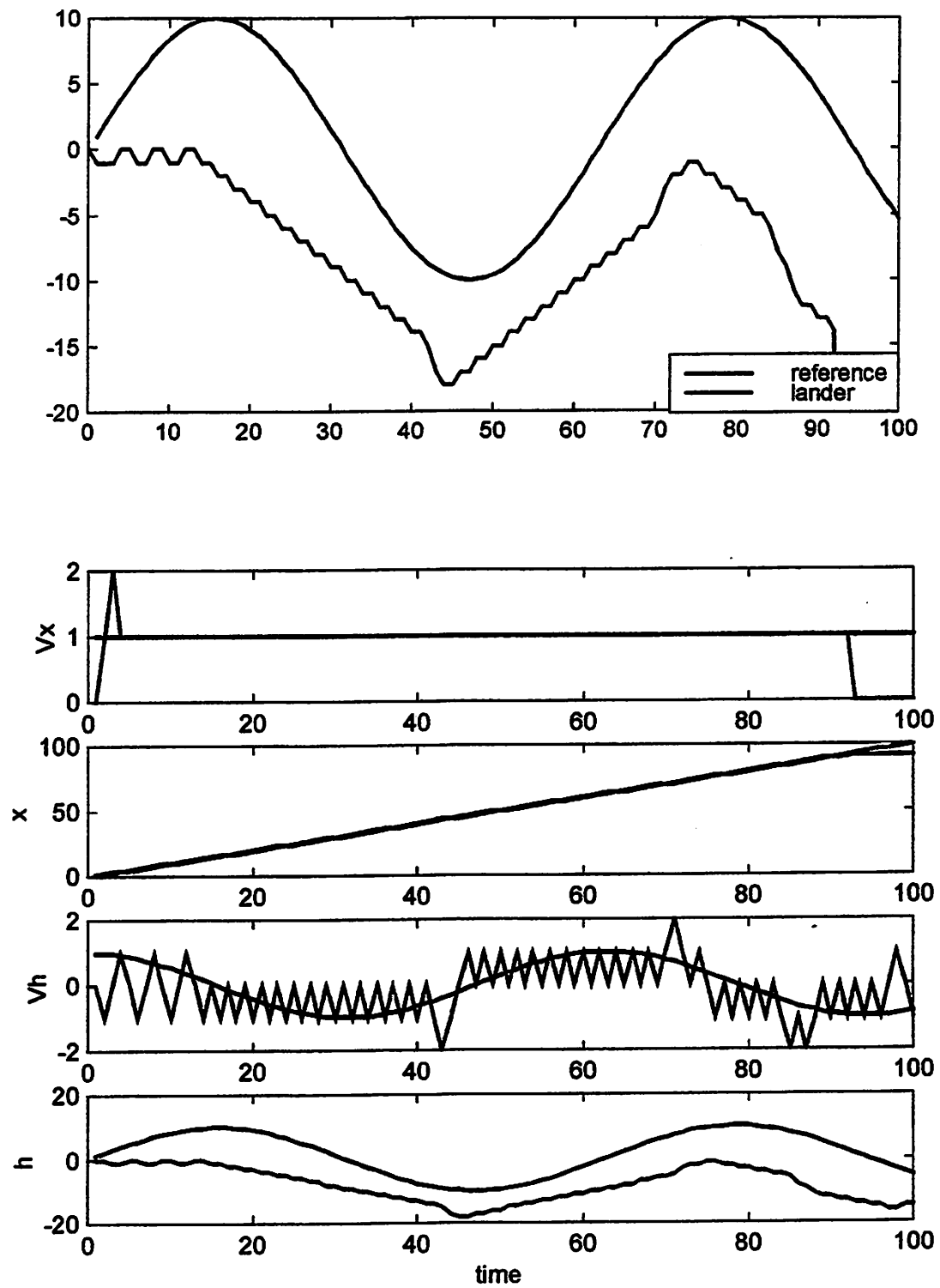
The next issue addressed in the paper is how MLD systems can be controlled to track a reference state. As was mentioned before, oscillation of auxiliary variables may preclude their use in infinite horizon optimizations. Also, it is desirable to include feedback of the actual state in the control law. Predictive control is suggested as a strategy which combines the optimization algorithm with state feedback, and thus provides a good solution to the tracking control problem.

In short, predictive control is a scheme in which optimal control commands are computed online over a specified time window. Only a portion of these input commands are actually applied (in the article the authors use only the first command), after which the process is repeated. This control law is referred to as *mixed integer predictive control* (MIPC) and is shown to converge when the equilibrium state is admissible.

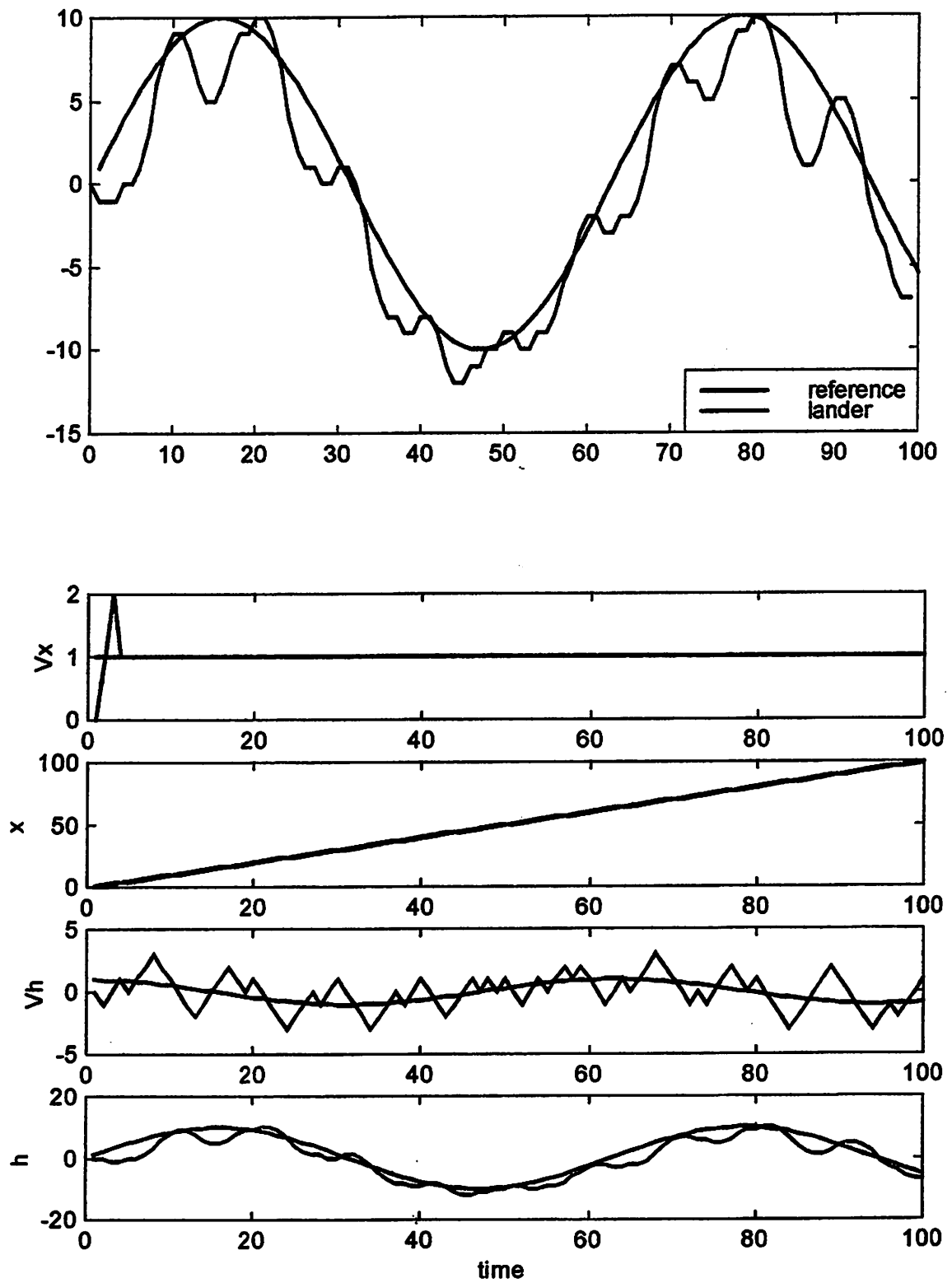
In applying MIPC to the lunar lander, special considerations had to be taken to account for the previously mentioned existence conditions. Because it is important that the online optimization always be feasible, it was generally impossible to apply the terminal equality constraints outlined in the previous section. This problem was fixed by finding the closest reachable state to the desired final state, with the aid of the 4 existence conditions (see code). Even with this adjustment, the optimization time window was at times too small. In these cases, the optimization was repeated with a larger final time until a solution was found or a maximum final time was reached.

Following are some example runs. The objective is to track a sine wave position reference (with appropriate cosine velocity reference) using predictive control. The three plots show solutions obtained when varying the minimum optimization time window (T_{win}) and the control application time (T_{app}).

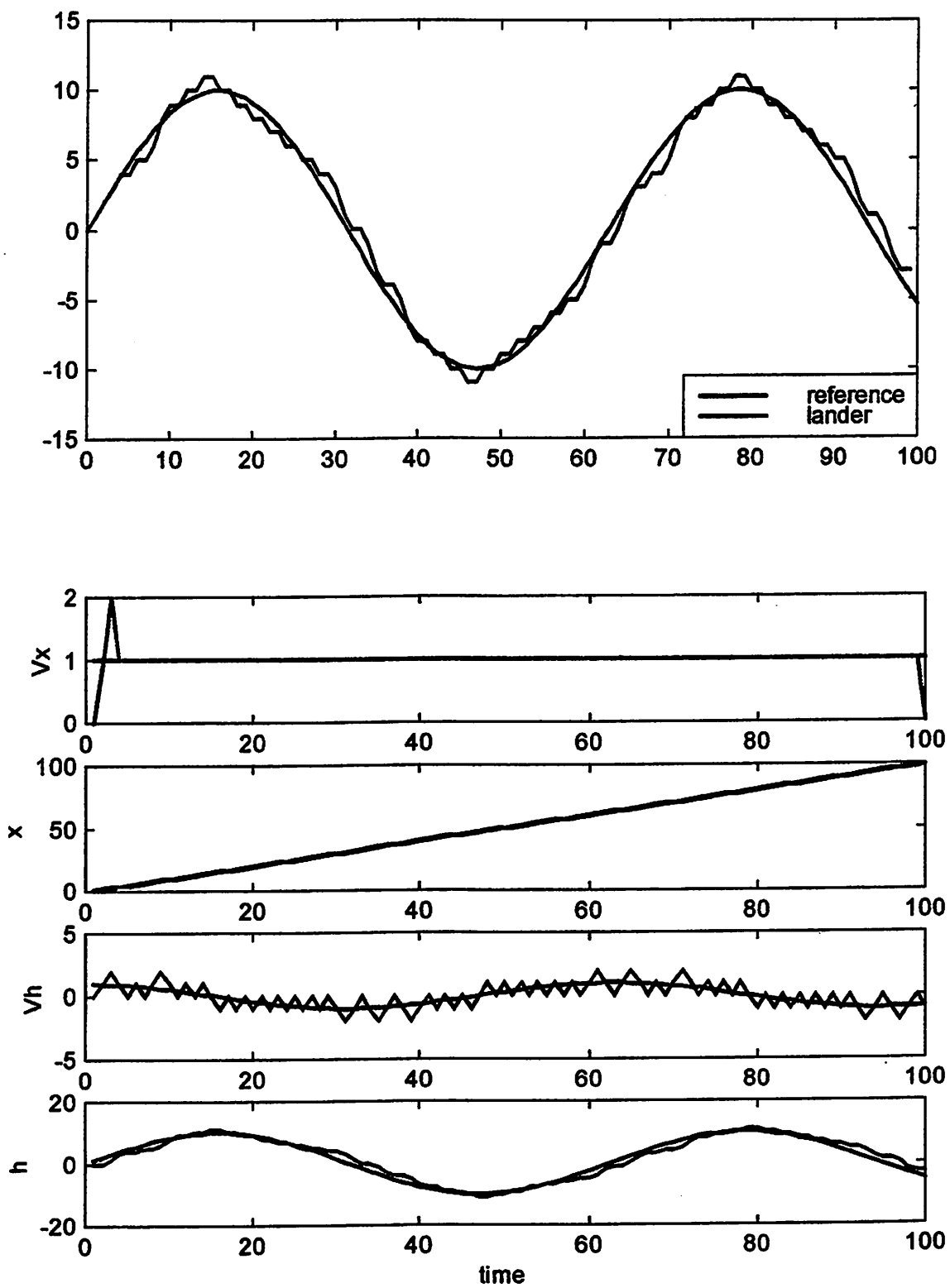
Example 1): $T_{win} \geq 10$ $T_{app} = 1$



Example 2): $T_{win} \geq 10$ $T_{app} = 10$



Example 3): $T_{win} \geq 1$ $T_{app} = 1$



Conclusions

The paper presents an alternative approach to modeling certain hybrid systems, which is especially useful when applying online optimization techniques. It is not clear to me however how this optimal control problem compares to the one obtained when emploting the hybrid systems concepts of EECS 291e, or if they are actually not very different. The technique of incorporation of logic rules does have the interesting advantage of allowing for "soft" constraints and heuristic rules, represented by disjunctive constraints in the MIQP.

As for numerics, lp_solve proved to be a very efficient and user-friendly software for solving problems of this size. The average optimization had approximately 50 variables, 100 inequality constraints and 20 equality constraints and took 1 or 2 seconds to complete.

Reference

- [1] A. Bemporad and M. Morari. *Control of systems integrating logic, dynamics, and constraints*. Automatica. V35 n3 p407-427
- [2] H.P. Williams. *Model Building in Mathematical Programming*. 2nd ed. John Wiley & Sons, 1985

```

%%% LANDER.m %%%%%%%%%%%%%%%
% This routine uses calls to lpmex functions to
% construct formulate the MIQP and solve for
% optimal lander trajectories

```

```

global T

```

```

T=12;          % final time
g=1;           % gravity
z1m=2;         % horizontal thrust
z2m=2;         % vertical thrust

```

```

z=[0 0];
whichobj=2;    %choice of objective function
EPS=0.01;

```

```

vxo=0; % initial state
xo=-8;
vho=0;
ho=4;

```

```

vxf=-4; % final state
xf=0;
vhf=0;
hf=4;

```

```

land=lpmex('make_lp',0,6*T);

```

```

% Check necessary existence conditions
% and relax necessary boundary conditions
% -----

```

```

c1=0.5*z1m;
c2=vxf-vxo;
if( (mod(c2,c1)>EPS) & (c2~=0) )
    display('The prob. does not meet cond. #1a')
    if(mod(c2,c1)>=c1/2)
        vxf=vxf-mod(c2,c1)+c1
    else
        vxf=vxf-mod(c2,c1)
    end
end
c2=xf-xo-vxo*T;
if( (mod(c2,c1)>EPS) & (c2~=0) )
    display('The problem does not meet cond #2a')
    if(mod(c2,c1)>=c1/2)
        xf=xf-mod(c2,c1)+c1
    else
        xf=xf-mod(c2,c1)
    end
end
end

```

```

c1=z2m;
c2=g*T+vhf-vho;
if( (mod(c2,c1)>EPS) & (c2~=0) )
    display('The problem does not meet cond #3a')
    if(mod(c2,c1)>=c1/2)
        vhf=vhf-mod(c2,c1)+c1
    else
        vhf=vhf-mod(c2,c1)
    end
end
end

```

```

c2=0.5*g*T*(T-1)+hf-ho-vho*T;
if( (mod(c2,c1)>EPS) & (c2~=0) )
    display('The problem does not meet cond #4a')
    if(mod(c2,c1)>=c1/2)
        hf=hf-mod(c2,c1)+c1
    else
        hf=hf-mod(c2,c1)
    end
end
end

```

```

Xf=[vxf xf vhf hf];
clear c1 c2

```

```

%CONSTRAINTS

```

```

%-----

```

```

ZRow = zeros(1,6*T); % generic empty row

```

```

% Inequality constraints for z's
for i=1:T

```

```

    %% z1 %%

```

```

    row=ZRow;
    row(GetI2('z',1,i))=1;
    row(GetI2('d',1,i))=z1m;
    lpmex('add_constraint',land,row,0,z1m);

```

```

    row=ZRow;
    row(GetI2('z',1,i))=1;
    row(GetI2('d',3,i))=-z1m;
    lpmex('add_constraint',land,row,2,0);

```

```

    row=ZRow;
    row(GetI2('z',1,i))=1;
    row(GetI2('d',2,i))=-0.5*z1m;
    lpmex('add_constraint',land,row,2,0);

```

```

    row=ZRow;
    row(GetI2('z',1,i))=1;
    row(GetI2('d',2,i))=0.5*z1m;
    lpmex('add_constraint',land,row,0,z1m);

```

```

    %% z2 %%

```

```

    row=ZRow;
    row(GetI2('z',2,i))=1;
    row(GetI2('a',1,i))=z2m;
    lpmex('add_constraint',land,row,2,z2m);

```

```

    row=ZRow;
    row(GetI2('z',2,i))=1;
    row(GetI2('a',1,i))=z2m;
    lpmex('add_constraint',land,row,0,z2m);
end

```

```

% Equality constraints for logical variables
for i=1:T

```

```

    %% deltas %%

```

```

    row=ZRow;
    row(GetI2('d',1,i))=1;
    row(GetI2('d',2,i))=1;
    row(GetI2('d',3,i))=1;
    lpmex('add_constraint',land,row,1,1);
end

```

```

% Bounds for logical variables

```

```

for i=1:T

```

```

    %% deltas %%

```

```

    row=ZRow;
    row(GetI2('d',1,i))=1;
    lpmex('add_constraint',land,row,0,1);
    lpmex('add_constraint',land,row,2,0);

```

```

    row=ZRow;
    row(GetI2('d',2,i))=1;
    lpmex('add_constraint',land,row,0,1);
    lpmex('add_constraint',land,row,2,0);

```

```

    row=ZRow;
    row(GetI2('d',3,i))=1;

```



```

lpmex('add_constraint',land,row,0,1);
lpmex('add_constraint',land,row,2,0);

row=ZRow;                %% alphas %%
row(GetI2('a',1,i))=1;
lpmex('add_constraint',land,row,0,1);
lpmex('add_constraint',land,row,2,0);
end

% Terminal condition
row1=ZRow;
row2=ZRow;
row3=ZRow;
row4=ZRow;

for i=1:T
    row1(GetI2('z',1,i))=1;
    row2(GetI2('z',1,T-i+1))=i-1;
    row3(GetI2('z',2,i))=1;
    row4(GetI2('z',2,T-i+1))=i-1;
end

lpmex('add_constraint',land,row1,1,vxf-
vxo+0.5*zlm*T);
lpmex('add_constraint',land,row2,1,xf-xo-
vxo*T+0.25*zlm*T*(T-1));
lpmex('add_constraint',land,row3,1,vhf-vho+T*g);
lpmex('add_constraint',land,row4,1,hf-ho-
vho*T+0.5*g*T*(T-1));

if(l==0)
vxwin=0;
xwin=vxwin;
vhwin=0;
hwin=vhwin;
lpmex('add_constraint',land,row1,0,vxf-
vxo+0.5*zlm*T+vxwin);
lpmex('add_constraint',land,row1,2,vxf-
vxo+0.5*zlm*T-vxwin);
lpmex('add_constraint',land,row2,0,xf-
xo+0.25*zlm*T*(T-1)-vxo*T+xwin);
lpmex('add_constraint',land,row2,2,xf-
xo+0.25*zlm*T*(T-1)-vxo*T-xwin);
lpmex('add_constraint',land,row3,0,vhf-
vho+T*g+vhwin);
lpmex('add_constraint',land,row3,2,vhf-vho+T*g-
vhwin);
lpmex('add_constraint',land,row4,0,hf-
ho+0.5*g*T*(T-1)-vho*T+hwin);
lpmex('add_constraint',land,row4,2,hf-
ho+0.5*g*T*(T-1)-vho*T-hwin);
end

% They're all integers
for i=1:T
    lpmex('set_int',land,GetI2('d',1,i),1);
    lpmex('set_int',land,GetI2('d',2,i),1);
    lpmex('set_int',land,GetI2('d',3,i),1);
    lpmex('set_int',land,GetI2('a',1,i),1);
end

% Objective function
row=ZRow;
for i=1:T
    if(whichobj==1)                % minimize total
        thrust
            row(GetI2('d',1,i))=1;
            row(GetI2('d',3,i))=1;
            row(GetI2('a',1,i))=1;
        end
    end
end

```

```

        if(whichobj==2)                % minimize thrust
            and time
                row(GetI2('d',1,i))=i;
                row(GetI2('d',3,i))=i;
                row(GetI2('a',1,i))=i;
            end
        end

lpmex('set_obj_fn',land,row);

% Solve the problem
result=lpmex('solve',land);
if(result==0)
    [obj,x,duals]=lpmex('get_solution',land);
    z = [x(4*T+1) x(5*T+1)];
else
    error('The problem is infeasible')
end

% compute and plot the state
S=[0;x0;0;ho];
d1 = [x(1:T);x(T)];
d2 = [x(T+1:2*T);x(2*T)];
d3 = [x(2*T+1:3*T);x(3*T)];
a1 = [x(3*T+1:4*T);x(4*T)];
z1 = [x(4*T+1:5*T);x(5*T)];
z2 = [x(5*T+1:6*T);x(6*T)];

for i=1:T
    S(:,i+1)=[S(1,i)+z1(i)-0.5*zlm;
                S(2,i)+S(1,i);
                S(3,i)+z2(i)-g;
                S(4,i)+S(3,i)];
end

figure(1), clf
plot(S(2,:),S(4,:),'-o');
xlabel('x position')
ylabel('h position')
title('Landing Trajectory')

figure(2), clf

subplot(311)
stairs([0:T],z1-0.5*zlm,'k+-')
hold on
stairs([0:T],z2,'k--')
axis([0 T -max(zlm,z2m)-0.5 max(zlm,z2m)+0.5])
legend('horiz.','vert.',4)
ylabel('thrust')

subplot(312)
plot([0:T],S(1,:), 'k', [0:T],S(3,:), 'k--')
legend('vx','vh',4)
ylabel('velocity')

subplot(313)
plot([0:T],S(2,:), 'k', [0:T],S(4,:), 'k--')
legend('x','h',4)
xlabel('TIME')
ylabel('position')

figure(3), clf

subplot(411)
stairs([0:T],d1,'k')
ylabel('\delta 1')
axis([0 T -0.5 1.5])
title('Logical Variables')

```

```

subplot(412)
stairs([0:T],d2,'k')
ylabel('\delta 2')
axis([0 T -0.5 1.5])

subplot(413)
stairs([0:T],d3,'k')
ylabel('\delta 3')
axis([0 T -0.5 1.5])

subplot(414)
stairs([0:T],a1,'k')
ylabel('\alpha 1')
axis([0 T -0.5 1.5])
xlabel('TIME')

%Save to file
save dat
lpmex('delete_lp',land);

```

Pred.m

```

% Predictive controller for thrust lander
% Makes successive call to the optimizer (land3.m, which is the same as lander.m, but
% without the plots) to try to track a reference % signal defined by ref.

global T z1m z2m g
z1m=2;
z2m=2;
g=1;

% Create the reference trajectory
t=1:100;
for i=1:length(t)
    ref(i,:)=[1 i cos(i/10) 10*sin(i/10)];
end

% Iteratively call optimization
timewin=1;      % number of applied control commands
Tmax=15;        % maximum optimization window
Tmin=10;        % minimum optimization window

x=[0 0 0 0]; % initial state
i=1;
imax=length(t);

while i<length(t)
    T=Tmin;
    repeat=1;
    while((repeat~=0)&(T<=Tmax))
        [z repeat]=land3(x(i,:),ref(min(i+T-1,length(t)),:));
        if(repeat~=0)
            display('Increasing window size');
            T=T+1
        end
    end
    if(repeat==0)
        for j=1:timewin
            x(i+j,:)=[x(i+j-1,1)+z(j,1)-0.5*z1m ...
                x(i+j-1,2)+x(i+j-1,1) ...
                x(i+j-1,3)+z(j,2)-g ...
                x(i+j-1,4)+x(i+j-1,3)];
        end
        i=i+timewin;
    else
        display('Could not solve the problem')
        imax=i;
        break
    end
end

```

```

end

x=x(1:imax,:);
ref=ref(1:imax,:);
t=t(1:imax);

%% PLOTS
figure(1), clf
subplot(411)
plot(t,ref(:,1),'--',t,x(:,1))
ylabel('horizontal velocity')

subplot(412)
plot(t,ref(:,2),'--',t,x(:,2))
ylabel('horizontal position')

subplot(413)
plot(t,ref(:,3),'--',t,x(:,3))
ylabel('vertical velocity')

subplot(414)
plot(t,ref(:,4),'--',t,x(:,4))
ylabel('vertical position')
xlabel('time')

figure(2), clf
plot(x(:,2),x(:,4),ref(:,2),ref(:,4),'--')

%Save the data to a file
save data

```

Rectangular Hybrid Games

Benjamin Horowitz

EE 291E Class Project

Joint Work with Tom Henzinger and Rupak Majumdar

Abstract. In order to study control problems for hybrid systems, we generalize hybrid automata to *hybrid games* – say, controller vs. plant. If we specify the continuous dynamics by constant lower and upper bounds, we obtain *rectangular games*. We show that for rectangular games with objectives expressed in LTL (linear temporal logic), the winning states for each player can be computed, and winning strategies can be synthesized. Our result is sharp, as already reachability is undecidable for generalizations of rectangular systems, and optimal – singly exponential in the game structure and doubly exponential in the LTL objective. In this way we are able systematically to generalize the theory of hybrid systems from automata (single-player structures) [8] to games (multi-player structures).

1 Introduction

A *hybrid automaton* [1] is a mathematical model for a system with both discretely and continuously evolving variables, such as a digital computer that interacts with an analog environment. An important special case of a hybrid automaton is the *rectangular automaton* [13], where each discrete variable ranges over a finite domain, the enabling condition for each discrete change is a rectangular region of continuous states, and the first derivative of each continuous variable x is bounded by constants from below and above; that is, $\dot{x} \in [a, b]$. Rectangular automata are important for several reasons. First, they generalize *timed automata* [2] (for which $a = b = 1$) and naturally model real-time systems whose clocks have bounded drift. Second, they can over-approximate with arbitrary precision the behavior of hybrid automata with general linear and nonlinear continuous dynamics, as long as all derivatives satisfy the Lipschitz condition [10, 17]. Third, they form a most general class of hybrid automata for which the LTL *model-checking problem* can be decided: given a rectangular automaton \mathcal{A} and a formula φ of linear temporal logic over the discrete states of \mathcal{A} , it can be decided in polynomial space if all possible behaviors of \mathcal{A} satisfy φ [13].

Since hybrid automata are often used to model digital controllers for analog plants, an important problem for hybrid automata is the LTL control problem: given a hybrid automaton \mathcal{H} and an LTL formula φ , can the behaviors of \mathcal{H} be “controlled” so as to satisfy φ ? However, the hybrid automaton per se is an inadequate model for studying this problem because it does not differentiate between the capabilities of its individual components – the controller and the

plant, if you wish. Since the control problem is naturally formalized in terms of a two-player¹ game, we define *hybrid games*. Because our setup is intended to be as general as possible, we do not distinguish between a “discrete player” (which directs discrete state changes) and a “continuous player” (which advances time); rather, in a hybrid game, each of the two players can itself act like a hybrid automaton. The game proceeds in an infinite sequence of rounds and produces an ω -sequence of states. In each round, both players independently choose enabled moves; the pair of chosen moves either results in a discrete state change, or in a passage of time. In the special case of a *rectangular game*, the enabling condition of each move is a rectangular region of continuous states, and when time advances, then the derivative of each continuous variable is governed by a constant differential inclusion. Now, the *LTL control problem* for hybrid games asks: given a hybrid game \mathcal{R} and an LTL formula φ over the discrete states of \mathcal{R} , is there a strategy for player-1 so that all possible outcomes of the game satisfy φ ?

Our main result shows that the LTL control problem can be decided for rectangular games. This question had been open. Previously, beyond the finite-state case, control problems have been solved only for timed games [5, 14, 15], and for rectangular games under the assumption that the controller can move only at integer points in time [12] (sampling control). Control algorithms have also been proposed for linear and nonlinear hybrid games [20, 21], but in these cases convergence is not guaranteed. For timed games and sampling controllers, convergence is guaranteed because the underlying state space can be partitioned into finitely many bisimilarity classes, and the controller does not need to distinguish between bisimilar states. Our result is, to our knowledge, the first controllability result for infinite-state systems which does not rely on the existence of a finite bisimilarity quotient. Our result is sharp, because the control problem for a class of hybrid games is at least as hard as the reachability problem for the corresponding class of hybrid automata, and reachability has been proved undecidable for several minor extensions of rectangular automata [13]. The complexity of our algorithm, which requires singly exponential time in the game \mathcal{R} and doubly exponential time in the formula φ , is optimal, because control is harder than model checking: reachability control over timed games is EXPTIME hard [12]; LTL control over finite-state games is 2EXPTIME hard [18].

For the solution of infinite-state model-checking problems, such as those of hybrid automata, it is helpful if there exists a finite quotient space that preserves the properties under consideration [8]. Specifically, every timed automaton is bisimilar to a finite-state automaton [2]; every 2D rectangular automaton (with two continuous variables) is similar (simulation equivalent) to a finite-state automaton [9]; and every rectangular automaton is trace equivalent to a finite-state automaton [13]. Since LTL model checking can be reduced to model checking on the trace-equivalence quotient, the decidability of LTL model checking for rectangular automata follows. The three characterizations are sharp; for example,

¹ For the sake of simplicity, in this paper we restrict ourselves to the two-player case. All results generalize immediately to more than two players.

the similarity quotient of 3D rectangular automata can be infinite [11], and therefore the quotient approach does not lead to branching-time model-checking algorithms for rectangular automata.

We show that for appropriate generalizations of the state equivalences, the results for rectangular automata carry over to rectangular games. The proof of our main result, the decidability of LTL control for rectangular games (Theorem 8), is structured as follows. We first introduce the notions of a single-player structure \mathcal{F} , a two-player structure \mathcal{G} , and the single-player structure $\mathcal{F}_{\mathcal{G}}$ corresponding to \mathcal{G} . Next, we describe state equivalences for these structures, in particular trace equivalence with observable moves and game trace equivalence. We note that if \equiv is a trace equivalence with observable moves on $\mathcal{F}_{\mathcal{G}}$, then \equiv is a game equivalence on \mathcal{G} (Proposition 1). Further, if \equiv is a (finite) game trace equivalence on \mathcal{G} , then an appropriately defined (finite) quotient \mathcal{G}/\equiv may be used to answer the LTL control problem, and to synthesize controllers (Proposition 2). Finally, we observe that the proofs of [13] actually show that for the single player structure $\mathcal{F}_{\mathcal{G}_{\mathcal{R}}}$ corresponding to a rectangular game \mathcal{R} , trace equivalence with observable moves is finite. Our main theorem follows. Along the way, we argue that singular and timed hybrid games have finite game bisimulations and that 2D rectangular hybrid games have finite game simulations. Together, these results cleanly generalize the theory of rectangular hybrid automata to rectangular games.

2 Symbolic Game Structures

A *transition structure* (or single-player structure)

$$\mathcal{F} = (Q, \Pi, \langle \cdot \rangle, Moves, Enabled, \delta)$$

consists of a set Q of states, a set Π of observations, an observation function $\langle \cdot \rangle: Q \rightarrow 2^\Pi$ which maps each state to a set of observations, a set $Moves$ of moves, an enabling function $Enabled: Moves \rightarrow 2^Q$ which maps each move to the set of states in which it is enabled, and a partial transition function $\delta: Q \times Moves \rightarrow 2^Q$ which maps each move m and each state in $Enabled(m)$ to a set of successor states. A *step* of \mathcal{F} is a triple $q \xrightarrow{m} q'$ such that $q \in Enabled(m)$ and $q' \in \delta(q, m)$. A *run* of \mathcal{F} is an infinite sequence $r = s_0 s_1 s_2 \dots$ of steps $s_j = q_j \xrightarrow{m_j} q'_j$ such that $q_{j+1} = q'_j$ for all $j \geq 0$. The corresponding *trace*, denoted by $\langle r \rangle$, is the infinite sequence $\langle q_0 \rangle \langle q_1 \rangle \langle q_2 \rangle \dots$ of observation sets. The corresponding *trace with observable moves*, denoted by $\langle r \rangle_{obs}$, is the infinite sequence $\langle q_0 \rangle m^0 \langle q_1 \rangle m^1 \langle q_2 \rangle m^2 \dots$ of alternating observation sets and moves. For a state q , the *outcome* R^q from q is the set of all runs of \mathcal{F} which start at q . For a set R of runs, we write $\langle R \rangle$ for the set $\{\langle r \rangle \mid r \in R\}$ of corresponding traces, and similarly for traces with observable moves.

2.1 Game Structures and the LTL Control Problem

A (two-player) *game structure*

$$\mathcal{G} = (Q, \Pi, \langle \cdot \rangle, Moves_1, Moves_2, Enabled_1, Enabled_2, \delta)$$

consists of the same components as above, only that $Moves_1$ ($Moves_2$) is the set of moves of player-1 (player-2), $Enabled_1$ maps $Moves_1$ to 2^Q , $Enabled_2$ maps $Moves_2$ to 2^Q , and the partial transition function $\delta: Q \times Moves_1 \times Moves_2 \rightarrow 2^Q$ maps each move m_1 of player-1, each move m_2 of player-2, and each state in $Enabled_1(m_1) \cap Enabled_2(m_2)$ to a set of successor states. For $i = 1, 2$, we define $mov_i: Q \rightarrow 2^{Moves_i}$ to yield for each state q the set $mov_i(q) = \{m \in Moves_i \mid q \in Enabled_i(m)\}$ of player- i moves that are enabled in q . With the game \mathcal{G} we associate the underlying transition structure

$$\mathcal{F}_{\mathcal{G}} = (Q, \Pi, \langle\langle \cdot \rangle\rangle, Moves_1 \times Moves_2, Enabled, \delta') ,$$

where $Enabled(m_1, m_2) = Enabled_1(m_1) \cap Enabled_2(m_2)$ and $\delta'(q, (m_1, m_2)) = \delta(q, m_1, m_2)$.

At each step of a game, player-1 chooses a move m_1 which is enabled in the current state q , player-2 independently chooses a move m_2 which is enabled in q , and the game proceeds nondeterministically to a new state in $\delta(q, m_1, m_2)$. Formally, a *step* of \mathcal{G} is a triple $q \xrightarrow{m_1, m_2} q'$ such that $q \xrightarrow{(m_1, m_2)} q'$ is a step of $\mathcal{F}_{\mathcal{G}}$. The *runs* and *traces* (with or without observable moves) of games are defined as for transition structures.

A *strategy for player- i* is a function $f_i: Q^+ \rightarrow 2^{Moves_i}$ such that $f_i(w \cdot q) \subseteq mov_i(q)$ for every state sequence $w \in Q^*$ and state $q \in Q$. The strategy f_i is *memory-free* if $f_i(w \cdot q) = f_i(w' \cdot q)$ for all $w, w' \in Q^*$ and $q \in Q$. Let f_1 (f_2) be a strategy for player-1 (player-2). The *outcome* R_{f_1, f_2}^q from state $q \in Q$ for f_1 and f_2 is a subset of the runs of \mathcal{G} which start at q : a run $s_0 s_1 s_2 \dots$ is in R_{f_1, f_2}^q if for all $j \geq 0$, if $s_j = q_j \xrightarrow{m_1^j, m_2^j} q_{j+1}$, then $m_i^j \in f_i(q_0 q_1 \dots q_j)$ for $i = 1, 2$ and $q_0 = q$.

The *formulas of linear temporal logic* (LTL) are generated inductively by the grammar

$$\varphi ::= \pi \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2 ,$$

where $\pi \in \Pi$ is an observation. The LTL formulas are interpreted over the traces of \mathcal{G} in the usual way [6]. Player-1 can *control the state* $q \in Q$ for φ if there exists a strategy f_1 of player-1 such that for every strategy f_2 of player-2 and every run $r \in R_{f_1, f_2}^q$, $\langle\langle r \rangle\rangle$ satisfies φ .² In this case, we say that the strategy f_1 *witnesses the player-1 controllability of q for φ* . The *LTL control problem* asks, given a game structure \mathcal{G} and an LTL formula φ , which states of \mathcal{G} can be controlled by player-1 for φ . The *LTL controller synthesis problem* asks, in addition, for the construction of witnessing strategies. If the game structure \mathcal{G} is finite, the LTL control problem is PTIME-complete in the size of \mathcal{G} and 2EXPTIME-complete in the length of φ [3, 18]. Whereas for simple LTL formulas such as safety (e.g. $\Box\pi$ for $\pi \in \Pi$) controllability ensures the existence of memory-free witnessing strategies, this is not the case for arbitrary LTL formulas [19].

² Our choice to control for LTL formulas rather than, say, ω -automata [19] is arbitrary. In the latter case, only the complexity results must be modified accordingly.

2.2 State Equivalences and Quotients for Game Structures

State Equivalences on Transition Structures. Consider a transition structure $\mathcal{F} = (Q, \Pi, \langle \cdot \rangle, \text{Moves}, \text{Enabled}, \delta)$. A binary relation $\preceq^s \subseteq Q \times Q$ is a (*forward*) *simulation* if $p \preceq^s q$ implies the following two conditions:

1. $\langle p \rangle = \langle q \rangle$;
2. $\forall m \in \text{mov}(p). \forall p' \in \delta(p, m). \exists m' \in \text{mov}(q). \exists q' \in \delta(q, m'). p' \preceq^s q'$.

We say that p is *simulated by* q , in symbols $p \preceq^S q$, if there is a simulation \preceq^s with $p \preceq^s q$. We write $p \cong^S q$ if both $p \preceq^S q$ and $q \preceq^S p$. The relation \cong^S is called *similarity*. A binary relation \cong^b on Q is a *bisimulation* if \cong^b is a symmetric simulation. Define $p \cong^B q$ if there is a bisimulation \cong^b with $p \cong^b q$. The relation \cong^B is called *bisimilarity*. A binary relation \preceq^{-s} on Q is a *backward simulation* if $p \preceq^{-s} q$ implies $\langle p \rangle = \langle q \rangle$ and

$$\begin{aligned} \forall p' \in Q. \forall m \in \text{mov}(p'). p \in \delta(p', m) \Rightarrow \\ \exists q' \in Q. \exists m' \in \text{mov}(q'). q \in \delta(q', m') \wedge p' \preceq^{-s} q' . \end{aligned}$$

A binary relation \preceq^l on Q is a *trace containment* if $p \preceq^l q$ implies $\langle R^p \rangle \subseteq \langle R^q \rangle$. Define $p \preceq^L q$ if there is a trace containment \preceq^l with $p \preceq^l q$. We write $p \cong^L q$ if both $p \preceq^L q$ and $q \preceq^L p$. The relation \cong^L is called *trace equivalence*.

We also define stronger versions of these equivalences, where the moves are observable. A simulation \preceq^s has *observable moves* if condition (2) is strengthened to

- 2a. $\text{mov}(p) \subseteq \text{mov}(q)$;
- 2b. $\forall m \in \text{mov}(p). \forall p' \in \delta(p, m). \exists q' \in \delta(q, m). p' \preceq^s q'$.

Similarity with observable moves, denoted \cong_{obs}^S , is the kernel of the coarsest simulation with observable moves; and *bisimilarity with observable moves*, \cong_{obs}^B , is the coarsest symmetric simulation with observable moves. Two states p and q are *trace equivalent with observable moves*, written $p \cong_{obs}^L q$, if $\langle R^p \rangle_{obs} = \langle R^q \rangle_{obs}$. Throughout, we say that equivalence relation \equiv_1 *refines* equivalence relation \equiv_2 if $p \equiv_1 q$ implies $p \equiv_2 q$, and that the equivalence relation \equiv is *finite* if \equiv has finitely many equivalence classes. Clearly, \cong^B refines \cong^S , and \cong^S refines \cong^L . The relations with observable moves refine the corresponding relations without observable moves. In general, all refinements are proper.

State Equivalences on Game Structures. Consider a game structure $\mathcal{G} = (Q, \Pi, \langle \cdot \rangle, \text{Moves}_1, \text{Moves}_2, \text{Enabled}_1, \text{Enabled}_2, \delta)$. A binary relation $\preceq_g^s \subseteq Q \times Q$ is a *game simulation* if $p \preceq_g^s q$ implies the following conditions:³

1. $\langle p \rangle = \langle q \rangle$;
- 2a. $\text{mov}_1(q) = \text{mov}_1(p)$ and $\text{mov}_2(p) = \text{mov}_2(q)$;
- 2b. $\forall m_1 \in \text{mov}_1(q). \forall m_2 \in \text{mov}_2(p). \forall p' \in \delta(p, m_1, m_2). \exists q' \in \delta(q, m_1, m_2). p' \preceq_g^s q'$.

³ Our results would still hold if condition (2a) were weakened to $\text{mov}_1(q) \subseteq \text{mov}_1(p)$ and $\text{mov}_2(p) \subseteq \text{mov}_2(q)$. We use the current stronger version for simplicity, in particular so that game simulations need not be parameterized by a player.

A relation \preceq_g^l on Q is a *game trace containment* if $p \preceq_g^l q$ implies that for all strategies f_1 of player-1, there exists a strategy f'_1 of player-1 such that for all strategies f'_2 of player-2, there exists a strategy f_2 of player-2 such that $\langle\langle R_{f_1, f'_2}^q \rangle\rangle_{obs} \subseteq \langle\langle R_{f'_1, f_2}^p \rangle\rangle_{obs}$. From this, *game similarity* \cong_g^S , *game bisimilarity* \cong_g^B , and *game trace equivalence* \cong_g^L are defined in the familiar way.

It is not difficult to check that \cong_g^B refines \cong_g^S , and that \cong_g^S refines \cong_g^L and that these inclusions are proper.⁴ The following proposition, which follows immediately from the definitions, characterizes the game equivalences in terms of the underlying transition structure: if the moves are observable, then the game structure can be flattened.

Proposition 1. *Two states p and q of a game structure \mathcal{G} are game bisimilar (game similar, game trace equivalent) if p and q are bisimilar (similar, trace equivalent) with observable moves in the underlying transition structure $\mathcal{F}_{\mathcal{G}}$.*

It follows that usual partition refinement algorithms [9, 16] may be applied to $\mathcal{F}_{\mathcal{G}}$ to compute the game bisimilarity and the game similarity quotients.

We will now show that the game equivalences on a game structure suggest quotient structures that can be used for control.⁵ Let \equiv be an equivalence relation that refines \cong_g^L . The *quotient structure* \mathcal{G}/\equiv of \mathcal{G} with respect to \equiv is the game structure $(Q/\equiv, \Pi, \langle\langle \cdot \rangle\rangle/\equiv, Moves_1, Moves_2, Enabled_1/\equiv, Enabled_2/\equiv, \delta/\equiv)$ with

- $Q/\equiv = \{[q]_{\equiv} \mid q \in Q\}$ is the set of equivalence classes of \equiv ;
- $\langle\langle [q]_{\equiv} \rangle\rangle/\equiv = \langle\langle q \rangle\rangle$ (note that $\langle\langle \cdot \rangle\rangle/\equiv$ is well defined since $\langle\langle \cdot \rangle\rangle$ is uniform within each equivalence class);
- $[q]_{\equiv} \in Enabled_1/\equiv(m)$ if $\exists p \in [q]_{\equiv} . p \in Enabled_1(m)$ (note that this is equivalent to $\forall p \in [q]_{\equiv} . p \in Enabled_1(m)$ since \equiv refines \cong_g^L), and analogously for $Enabled_2/\equiv(m)$;
- $[q']_{\equiv} \in \delta([q]_{\equiv}, m_1, m_2)/\equiv$ if $\exists p' \in [q']_{\equiv} . \exists p \in [q]_{\equiv} . p' \in \delta(p, m_1, m_2)$.

The following proposition reduces control for LTL formula φ in \mathcal{G} to control in the quotient structure \mathcal{G}/\equiv .

Proposition 2. *Player-1 can control $q \in Q$ for φ in \mathcal{G} if and only if player-1 can control $[q]_{\equiv}$ for φ in \mathcal{G}/\equiv . Moreover, if the strategy f_1 witnesses the player-1 controllability of $[q]_{\equiv}$ for φ in \mathcal{G}/\equiv , then the strategy f'_1 defined by $f'_1(p_0 \dots p_k) \triangleq f_1([p_0]_{\equiv} \dots [p_k]_{\equiv})$ witnesses the player-1 controllability of q for φ in \mathcal{G} .*

⁴ In fact, \cong_g^S , \cong_g^B , and \cong_g^L refine the corresponding alternating equivalence relations introduced in [4].

⁵ In contrast, alternating equivalences do not suggest such quotient structures. It is easy to construct a game structure such that player-1 can control for $\Box\pi$ at states q_1 and q_2 with q_1 alternating bisimilar to q_2 , but player-1 cannot control for $\Box\pi$ in a reasonably defined quotient.

3 Rectangular Games

In this section, we apply the techniques developed in the previous section to a particular class of infinite-state game structures: rectangular hybrid games. For infinite-state games, algorithms for computing control strategies may often be guaranteed to terminate only if the state space has a suitable finite quotient. We show that such quotients do exist for all rectangular games, and thus that the LTL control and controller synthesis problems are decidable for rectangular games.

We generalize the rectangular automata of [13] to rectangular games, which are suitable for the study of control problems. A subset of τ of \mathbb{R}^n is *rectangular* if it is the cartesian product of n intervals, all of whose (finite) endpoints are rational. For the sake of simplicity, in this paper we restrict ourselves to the case where all rectangles are closed and bounded.⁶ Let \mathfrak{R}^n denote the set of all rectangles. Denote by τ_i the projection of τ on its i th coordinate, so that $\tau = \prod_{i=1}^n \tau_i$. For $i = 1, 2$, let $Moves_i^{time} = Moves_i \uplus \{time\}$, where *time* is a special symbol not in $Moves_1$ or $Moves_2$. A *rectangular game*

$$\mathcal{R} = (L, X, Moves_1, Moves_2, Enabled_1, Enabled_2, Flow, E, Jump, Post)$$

consists of

- a finite set L of locations;
- a set $X = \{x_1, \dots, x_n\}$ of real-valued variables;
- for $i = 1, 2$, a set $Moves_i$ of moves of player- i ;
- for $i = 1, 2$, a function $Enabled_i : Moves_i^{time} \rightarrow 2^{L \times \mathbb{R}^n}$, which specifies for each move m_i of player- i and each location ℓ , the rectangle in which m_i is enabled when control is at ℓ (we require that for all $m_i \in Moves_i^{time}$ and $\ell \in L$, the set $\{(\ell, \tau) \in Enabled_i(m_i)\}$ is a singleton;
- a function $Flow : L \rightarrow \mathfrak{R}^n$ which maps each location ℓ to a rectangle which constrains the evolution of the continuous variables when control is at ℓ ;
- a set $E \subseteq (L \times Moves_1 \times Moves_2^{time} \times L) \cup (L \times Moves_1^{time} \times Moves_2 \times L)$ of edges which specifies how control may pass from one location to another;
- a function $Jump : E \rightarrow 2^{\{1, \dots, n\}}$ which maps each edge to the indices of continuous variables which are reset upon jumping along that edge; and
- a function $Post : E \rightarrow \mathfrak{R}^n$ which constrains the values of the continuous variables after a jump.

The *dimension* of \mathcal{R} is n , the number of continuous variables. We define the *invariant region* $inv(\ell)$ of location ℓ to be $Enabled_1(time, \ell) \cap Enabled_2(time, \ell)$. The *set of states* of \mathcal{R} is defined to be $\{(\ell, x) \mid x \in inv(\ell)\}$.

Informally, when a rectangular game is in state (ℓ, x) , time can progress as long as both players choose *time*, and the system is in the invariant region $inv(\ell)$. In addition, each player is allowed to choose a discrete move that is enabled at the current state. During discrete steps, for each i in the jump set $Jump(e)$, x_i

⁶ The general case can be treated analogously to [13].

is nondeterministically assigned a new value in the postguard interval $Post(e)_i$. For each $i \notin Jump(e)$, x_i is unchanged, and must lie in $Post(e)_i$.

We now formally define the semantics of rectangular games. With the n -dimensional rectangular game \mathcal{R} we associate the underlying game structure

$$\mathcal{G}_{\mathcal{R}} = (L \times \mathbb{R}^n, L, \langle \cdot \rangle, Moves_1^{time}, Moves_2^{time}, Enabled_1, Enabled_2, \delta) ,$$

where $\langle (\ell, x) \rangle = \{\ell\}$, and $(\ell', x') \in \delta((\ell, x), m_1, m_2)$ if either

- [Time step of duration $t \geq 0$] $\ell' = \ell$; $(m_1, m_2) = (time, time)$; $x' = x + t \cdot s$, where $s \in Flow(\ell)$; and for all $0 \leq t' \leq t$, $(x + t' \cdot s) \in inv(\ell)$;
- [Discrete step] there exists an edge $e = (\ell, m_1, m_2, \ell') \in E$ such that for $i = 1, 2$, $m_i \in mov_i(\ell, x)$, $x' \in Post(e)$, and $x'_i = x_i$ for all $i \notin Jump(e)$.

For a rectangular game \mathcal{R} , and an LTL formula φ , the LTL control problem asks which states of $\mathcal{G}_{\mathcal{R}}$ can be controlled for φ . Since the divergence of time can be expressed in LTL, when studying the LTL control problem there is no need to restrict our attention to the runs of a rectangular game along which the sum of durations of all time steps diverges. To express the divergence of time in LTL, we add an additional real-valued variable x_{n+1} , and for each location ℓ , we add a new location ℓ' . Call the set of locations added L' . For all $\ell \in L \cup L'$, we set $Flow(\ell)_{n+1} = [1, 1]$. Our new invariant region $inv(\ell)$ requires that $x_{n+1} \leq 1$; when $x_{n+1} = 1$, each player is allowed to make a move *reset* with $(\ell, reset, reset, \ell')$ now added to E . Upon jumping to ℓ' , x_{n+1} is reset to 0, and a jump back to ℓ is forced immediately in a similar manner. The property that time diverges may be expressed in LTL as $\Box \Diamond \bigvee_{\ell' \in L'} \ell'$.

Let x_i be a variable of a rectangular game \mathcal{R} . The variable x_i is a *clock* if for each location ℓ , $Flow(\ell)_i = [1, 1]$; and x is a *finite slope variable* if for each location ℓ , $Flow(\ell)_i$ is a singleton. The rectangular game \mathcal{R} has *deterministic jumps* if for each edge e , and each coordinate $i \in Jump(e)$, the interval $Post(e)_i$ is a singleton. The rectangular game \mathcal{R} is *initialized* if for every edge $e = (\ell, \cdot, \cdot, \ell')$ and every coordinate i , if $Flow(\ell) \neq Flow(\ell')$ then $i \in Jump(e)$. If \mathcal{R} has deterministic jumps, then \mathcal{R} is a *timed game* if every variable is a clock, and \mathcal{R} is a *singular game* if every variable is a finite-slope variable. In what follows, we shall only consider initialized rectangular games with deterministic jumps.⁷ Without loss of generality, we assume that all constants appearing in the definition of a rectangular game are integers.

The game bisimilarity (similarity, language-equivalence) quotient of a rectangular game \mathcal{R} is defined to be the game bisimilarity (similarity, language-equivalence) quotient on the underlying game structure $\mathcal{G}_{\mathcal{R}}$. In what follows, we shall speak of one rectangular game \mathcal{R}_1 simulating another rectangular game \mathcal{R}_2 , with the understanding that this refers to a simulation relation on the disjoint union of the states of $\mathcal{G}_{\mathcal{R}_1}$ and the states of $\mathcal{G}_{\mathcal{R}_2}$, i.e. a game simulation relation $\preceq_g^s \subseteq (L_1 \times \mathbb{R}^{n_1}) \times (L_2 \times \mathbb{R}^{n_2})$.

⁷ For non-initialized games, the reachability problem is already undecidable [13].

3.1 Game Bisimilarity for Singular Games

To see that every singular \mathcal{S} game has a finite game bisimilarity quotient, we first define the *region equivalence relation* \cong^R , as follows [1, 2]. Let $a = (a_1, \dots, a_n)$ be an n -tuple of integers. For a real number x , let $\text{fract}(x)$ denote the fractional part of x . For a vector x , let $\text{fract}(x)$ denote the vector whose i th coordinate is $\text{fract}(x_i)$. Define $x \equiv_a y$ iff for $i = 1, \dots, n$, (1) $\lfloor a_i x_i \rfloor = \lfloor a_i y_i \rfloor$, (2) $\text{fract}(a_i x_i) = 0$ iff $\text{fract}(a_i y_i) = 0$, and (3) for $j \neq i$, $\text{fract}(a_i x_i) < \text{fract}(a_j x_j)$ iff $\text{fract}(a_i y_i) < \text{fract}(a_j y_j)$. For each $x_i \in X$, let c_i denote the largest rational constant that appears as the i th (finite) coordinate of any rectangle in the definition of \mathcal{S} . Two states (ℓ, x) and (ℓ', x') are *region equivalent* if (1) $\ell = \ell'$; (2) for all $x_i \in X$, either $\lfloor x_i \rfloor = \lfloor x'_i \rfloor$ or both $\lfloor x_i \rfloor$ and $\lfloor x'_i \rfloor$ are greater than c_i ; and (3) $\text{fract}(x) \equiv_a \text{fract}(x')$, where $a_i = k_i$ if $\text{Flow}(\ell)_{x_i} = [k_i, k_i]$, $k_i \neq 0$, and $a_i = 1$ if $k_i = 0$. Note that $a = (1, 1, \dots, 1)$ for a timed game, and that the number of equivalence classes of \cong^R may be exponential in the size of the description of \mathcal{S} . The arguments of [1, 2] show that the region equivalence \cong^R is a bisimulation with observable moves on the single-player structure $\mathcal{F}_{\mathcal{G}_S}$ associated with \mathcal{G}_S . Using Proposition 1, we may conclude that \cong^R is a game bisimulation on \mathcal{S} .

Theorem 3. *For every singular game, the region equivalence \cong^R refines the game bisimilarity \cong_g^B .*

It follows that every singular game has a finite quotient structure with respect to game bisimulation. The game bisimilarity quotient of a singular game may have an exponential number of equivalence classes (regions). Since it refines game trace equivalence, by Proposition 2, the finite quotient can be used for LTL controller synthesis.

Corollary 4. *Given an LTL formula φ , the LTL control problem for a singular game \mathcal{S} is EXPTIME-complete in the size of the description of \mathcal{S} and 2EXPTIME-complete in the length of φ .*

Singular games are a maximal class of hybrid games for which finite game bisimilarity quotients exist. In particular, there exists a 2D rectangular game \mathcal{R} such that the equality relation is the only game bisimulation on $\mathcal{G}_{\mathcal{R}}$ [7].

3.2 Game Similarity for 2D Rectangular Games

We define the *double-region equivalence relation* \cong^{2R} on the states of a 2D rectangular game \mathcal{T} as the intersection of two region equivalences, as follows. Let \equiv_a and \equiv_b be two equivalence relations as defined in Sect. 3.1. Call the intersection of these two relations $\equiv_{a,b}$. Let c be the largest rational constant that appears in the definition of \mathcal{T} . For a location ℓ with $\text{Flow}(\ell) = [a_1, b_1] \times [a_2, b_2]$, let $a_\ell = (a_2, b_1)$ and $b_\ell = (b_2, a_1)$. Two states (ℓ, x) and (ℓ', y) of a \mathcal{T} rectangular game are *double-region equivalent*, in symbols $(\ell, x) \cong^{2R} (\ell', y)$, if (1) $\ell = \ell'$, (2) $\text{fract}(x) \equiv_{a_\ell, b_\ell} \text{fract}(y)$, and (3) for $i = 1, 2$ either $\lfloor x_i \rfloor = \lfloor y_i \rfloor$ or both $x_i > c$ and $y_i > c$. Note that the number of equivalence classes of \cong^{2R} is exponential in

the size of the description of \mathcal{T} . The arguments of [9] show that the region equivalence \cong^{2R} is a simulation with observable moves on the single-player structure associated with $\mathcal{G}_{\mathcal{T}}$. Using Proposition 1, we may conclude that \cong^{2R} is a game simulation on \mathcal{T} .

Theorem 5. *For every 2D rectangular game, the double-region equivalence \cong^{2R} refines the game similarity \cong_g^S .*

This implies that every 2D rectangular game has a finite quotient structure with respect to game similarity. The game similarity quotient may have an exponential number of equivalence classes. Since the game similarity quotient refines game trace equivalence, by Proposition 2, the finite quotient can be used for LTL controller synthesis.

Corollary 6. *Given an LTL formula φ , the LTL control problem for a 2D rectangular game \mathcal{T} can be solved in time exponential in the size of the description of \mathcal{T} , and is 2EXPTIME-complete in the length of φ .*

2D rectangular games are a maximal class of hybrid games for which finite game similarity quotients exist. In particular, there exists a 3D rectangular game \mathcal{R} such that the equality relation is the only game simulation on $\mathcal{G}_{\mathcal{R}}$ [11].

3.3 Game Trace Equivalence for Rectangular Games

Although initialized rectangular games do not have finite game similarity quotients, we can show that they have finite game language-equivalence quotients. To prove this, we sketch how to translate an n -dimensional rectangular game \mathcal{R} into a $2n$ -dimensional singular game $\mathcal{S}_{\mathcal{R}}$ such that \mathcal{R} and $\mathcal{S}_{\mathcal{R}}$ are game language equivalent. For details, see [13]. The game $\mathcal{S}_{\mathcal{R}}$ has the same vertex and move sets as \mathcal{R} . We replace each variable x_i of \mathcal{R} by two finite-slope variables $c_{l(i)}$ and $c_{u(i)}$ such that when $\text{Flow}_{\mathcal{R}}(v)(x_i) = [k_l, k_u]$, then $\text{Flow}_{\mathcal{S}_{\mathcal{R}}}(v)(c_{l(i)}) = [k_l, k_l]$, and $\text{Flow}_{\mathcal{S}_{\mathcal{R}}}(v)(c_{u(i)}) = [k_u, k_u]$. Intuitively, the variable $c_{l(i)}$ tracks the least possible value of x_i and the variable $c_{u(i)}$ tracks the greatest possible value of x_i . With each edge step, the values of the variables are appropriately updated so that the interval $[c_{l(i)}, c_{u(i)}]$ maintains the possible values of x_i .

To prove that \mathcal{R} and $\mathcal{S}_{\mathcal{R}}$ are game trace equivalent, we define a map $\gamma : Q_{\mathcal{S}_{\mathcal{R}}} \rightarrow 2^{Q_{\mathcal{R}}}$ which maps each state of $\mathcal{S}_{\mathcal{R}}$ to a set of states of \mathcal{R} , by $\gamma(\ell, \mathbf{x}) = \{\ell\} \times \prod_{i=1}^n [x_{l(i)}, x_{u(i)}]$. Call a state $(\ell, \mathbf{x}) \in Q_{\mathcal{S}_{\mathcal{R}}}$ an *upper-half state* of $\mathcal{S}_{\mathcal{R}}$ if for every index $i \in \{1, \dots, n\}$, we have $x_{l(i)} \leq x_{u(i)}$. Notice that we are only interested in upper-half states of $\mathcal{S}_{\mathcal{R}}$. We set $\gamma(q) = \emptyset$ if q is not an upper-half state of $\mathcal{S}_{\mathcal{R}}$. The arguments of [13] show that a state q of the single-player structure associated with $\mathcal{S}_{\mathcal{R}}$ forward simulates with observable moves any state $p \in \gamma(q)$ of \mathcal{R} , and any state $p \in \gamma(q)$ backwards simulates q with observable moves. Using this, and Proposition 1, we have:

Lemma 7. *Let \mathcal{R} be a rectangular game, let q be a state of the singular game $\mathcal{S}_{\mathcal{R}}$, and let $p \in \gamma(q)$ be a corresponding state of \mathcal{R} . Then p is game simulated by q , and q is backward game simulated by p .*

The above result also holds when the durations of the time moves are also observable. Note that the above lemma only ensures equivalence for *finite* traces. However, since the rectangles used in the definition of rectangular games are compact, it follows (as in [13]) that the language of the \mathcal{R} is limit closed.⁸ Hence, the above lemma is sufficient to show game trace equivalence.

Theorem 8. *For every rectangular game \mathcal{R} , every state q of the singular game $\mathcal{S}_{\mathcal{R}}$, and every state $p \in \gamma(q)$ of \mathcal{R} , the states p and q are game trace equivalent.*

Since the singular game $\mathcal{S}_{\mathcal{R}}$ has a finite game trace equivalence, it follows that the rectangular game \mathcal{R} has a finite quotient structure with respect to game trace equivalence. The game trace-equivalence quotient of \mathcal{R} can be used for controller synthesis (Proposition 2). It may have an exponential number of equivalence classes (corresponding to the regions of $\mathcal{S}_{\mathcal{R}}$).

Corollary 9. *Given an LTL formula φ , The LTL control problem for a rectangular game \mathcal{R} is EXPTIME-complete in the size of \mathcal{R} and 2EXPTIME-complete in the length of φ .*

Rectangular games are a maximal class of hybrid games for which finite game trace-equivalence quotients are known to exist. In particular, for *triangular games*, where some enabling conditions for moves have constraints of the form $x_i \leq x_j$, the reachability problem, and therefore the safety control problem, are undecidable [13]. We also note that the shape of a witnessing strategy for the LTL control of rectangular games, even for the safety control of timed games, is not necessarily rectangular, but may require triangular constraints of the form $x_i \leq x_j$ to determine which move to apply in a given state. Hence, the synthesized controller may not be implementable as another rectangular automaton. This is in contrast to the timed case, where the timed automata *with triangular enabling conditions* are reducible to finite quotients [2] and closed under controller synthesis [5, 15].

We conclude with an observation that is important for making the control of rectangular games practical. For a set $S \subseteq L \times \mathbb{R}^n$ of states, define the *uncontrollable predecessors* $\text{upre}(S)$ of S to be

$$\left\{ (\ell, x) \in L \times \mathbb{R}^n \mid \begin{array}{l} x \in \text{inv}(\ell) \wedge \\ \forall m_1 \in \text{mov}_1(\ell, x). \exists m_2 \in \text{mov}_2(\ell, x). \\ \delta((\ell, x), m_1, m_2) \cap S \neq \emptyset \end{array} \right\} .$$

For the safety control of a rectangular game \mathcal{R} , rather than constructing the region equivalence quotient of $\mathcal{S}_{\mathcal{R}}$, it is computationally much preferable to iterate a symbolic *upre* operator directly on \mathcal{R} . In other words, to compute the set of states at which player-1 can control for $\Box(\ell_{i_1} \vee \dots \vee \ell_{i_j})$, compute

$$\neg \bigcup_{i=0}^{\infty} \text{upre}^i(\{(\ell, x) \in L \times \mathbb{R}^n \mid \ell \notin \{\ell_{i_1}, \dots, \ell_{i_j}\} \wedge x \in \text{inv}(\ell)\}) .$$

⁸ An ω -language L is *limit closed* if for every infinite word ρ , if every finite prefix of ρ is a prefix of some word in L , then ρ is also in L .

This algorithm is being implemented in HYTECH. Since $upre(S)$ is a union of game language equivalence classes when S is, this iteration always terminates.

Theorem 10. *The symbolic algorithm for safety control terminates when applied to rectangular games.*

4 Conclusions

Our results for two-player hybrid games, which extend also to multiple players, are summarized in the right column of the table below. They can be seen clearly to generalize the known results for hybrid automata (i.e., single-player hybrid games), which are summarized in the center column. The number of equivalence classes of all finite equivalences in the table is exponential in the given automaton or game. The infinitary results in the right column follow immediately from the corresponding results in the center column.

Table 1. Summary of results

	Hybrid automata (single-player)	Hybrid games (multi-player)
Timed, singular	finite bisimilarity [1, 2]	finite game bisim.
2D Rectangular	infinite bisim. [7] finite similarity [9]	infinite game bisim. finite game sim.
Rectangular	infinite sim. [11] finite trace equiv. [13]	infinite game sim. finite game trace equiv.
Triangular	infinite trace equiv. [13]	infinite game trace equiv.

References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. *Theoretical Computer Science* **138** (1995) 3–34
2. Alur, R., Dill, D.L.: A theory of timed automata. *Theoretical Computer Science* **126** (1994) 183–235
3. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. In: *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press (1997) 100–109
4. Alur, R., Henzinger, T.A., Kupferman, O., Vardi, M.Y.: Alternating refinement relations. In: Sangiorgi, D., de Simone, R. (eds.): *CONCUR 97: Concurrency Theory*. Lecture Notes in Computer Science, Vol. 1466. Springer-Verlag, Berlin Heidelberg New York (1998) 163–178

5. Asarin, E., Maler, O., Pnueli, A., Sifakis, J.: Controller synthesis for timed automata. In: *Proceedings of the IFAC Symposium on System Structure and Control*. Elsevier Science Publishers (1998) 469–474
6. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.): *Handbook of Theoretical Computer Science*, Vol. B. Elsevier Science Publishers (1990) 995–1072
7. Henzinger, T.A.: Hybrid automata with finite bisimulations. In: Fülöp, Z., Gécseg, F. (eds.): *ICALP 95: Automata, Languages, and Programming*. Lecture Notes in Computer Science, Vol. 944. Springer-Verlag, Berlin Heidelberg New York (1995) 324–335
8. Henzinger, T.A.: The theory of hybrid automata. In: *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press (1996) 278–292
9. Henzinger, M.R., Henzinger, T.A., and Kopke, P.W.: Computing simulations on finite and infinite graphs. In: *Proceedings of the 36rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press (1995) 453–462
10. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control* **43** (1998) 540–554
11. Henzinger, T.A., Kopke, P.W.: State equivalences for rectangular hybrid automata. In: Montanari, U., Sassone, V. (eds.): *CONCUR 96: Concurrency Theory*. Lecture Notes in Computer Science, Vol. 1119. Springer-Verlag, Berlin Heidelberg New York (1996) 530–545
12. Henzinger, T.A., Kopke, P.W.: Discrete-time control for rectangular hybrid automata. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.): *ICALP 97: Automata, Languages, and Programming*. Lecture Notes in Computer, Vol. 1256. Springer-Verlag, Berlin Heidelberg New York (1997) 582–593
13. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? *Journal of Computer and System Sciences* **57** (1998) 94–124
14. Hoffmann, G., Wong-Toi, H.: The input-output control of real-time discrete-event systems. In: *Proceedings of the 13th Annual Real-time Systems Symposium*. IEEE Computer Society Press (1992) 256–265
15. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems. In: Mayr, E.W., Puech, C. (eds.): *STACS 95: Theoretical Aspects of Computer Science*. Lecture Notes in Computer Science, Vol. 900. Springer-Verlag, Berlin Heidelberg New York (1995) 229–242
16. Paige, R., Tarjan, R.E.: Three partition-refinement algorithms. *SIAM Journal of Computing* **16** (1987) 973–989
17. Puri, A., Borkar, V., Varaiya, P.: ϵ -approximation of differential inclusions. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.): *Hybrid Systems III*. Lecture Notes in Computer Science, Vol. 1066. Springer-Verlag, Berlin Heidelberg New York (1996), 362–376
18. Rosner, R.: *Modular Synthesis of Reactive Systems*. Ph.D. thesis, Weizmann Institute of Science (1992)
19. Thomas, W.: On the synthesis of strategies in infinite games. In: Mayr, E.W., Puech, C. (eds.): *STACS 95: Theoretical Aspects of Computer Science*. Lecture Notes in Computer Science, Vol. 900. Springer-Verlag, Berlin Heidelberg New York (1995) 1–13
20. Tomlin, C.: *Hybrid Control of Air Traffic Management Systems*. Ph.D. thesis, University of California, Berkeley (1998)
21. Wong-Toi, H.: The synthesis of controllers for linear hybrid automata. In: *Proceedings of the 36th Conference on Decision and Control*. IEEE Computer Society Press (1997) 4607–4612

Some attempts to stochastic hybrid system

Jianghai Hu
Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720
jianghai@robotics.eecs.berkeley.edu

May 10, 1999

1 Introduction

In the conventional formulation of hybrid system (See, for example, [5]), there is no place for randomness. Although the deterministic framework captures many characteristics of the real systems in applications, in other cases, the missing flavor of randomness will indeed be a fatal flaw because of the inherent uncertainty in the environment in most real world applications.

The idea of introducing stochastic hybrid system is not new. Different researchers have tried to propose different models from their own perspectives. For the most recent, and I believed, most relevant literature, the readers are referred to [1, 4, 7, 2, 8]. The most important difference lies in where to introduce the randomness.

One obvious choice is to replace the deterministic jumps between discrete states by random jumps governed by some prescribe probabilistic law. Hence the evolution of the discrete states constitutes a time homogenous Markov chain. The question remained then is when does such jump occur? In [1], the jumps occur every ϵ time, and the effect when $\epsilon \rightarrow 0$ is studied. In [7], however, the transitions follow a continuous time Markov process. In both papers, the discrete random transitions are assumed to be independent of continuous dynamics, therefore the models can actually be better viewed as an extension of Markov process with some continuous states attached whose evolutions follow state-dependent differential equations.

Another choice is to replace the deterministic dynamics inside the invariant set of each discrete state by a stochastic differential equation. Therefore, even if we keep the deterministic discrete transition part, starting from a fixed initial state, different guards can be activated depending on the realization of the solution stochastic process, thus different discrete transitions occurred randomly. More general models can be proposed by blending the above two choices.

This report is organized as following: in Section 2, we will try to give a general definition of hybrid system based on the second choice mentioned above. An example will be shown in Section 3 together with its analysis. In Section 4, the idea will be applied to a more general framework, in which we will approximate the solution of the stochastic differential equation in \mathbb{R}^1 by the stochastic hybrid automata obtained from state space discretization.

2 General Definition

Definition 1 (Stochastic Hybrid System) *A stochastic hybrid system (or automata) is a collection $H = (Q, X, Inv, f, g, G, R)$ where*

- Q is a discrete variable taking countably many values in $\mathbf{Q} = \{q_1, q_2, \dots\}$;
- X is a continuous variable taking values in $\mathbf{X} = \mathbb{R}^N$ for some N ;
- $Inv : \mathbf{Q} \rightarrow 2^{\mathbf{X}}$ assigns to each $q \in \mathbf{Q}$ an invariant open subset of \mathbf{X} ;
- $f, g : \mathbf{Q} \times \mathbf{X} \rightarrow T\mathbf{X}$ are vector fields;
- $G : E = \mathbf{Q} \times \mathbf{Q} \rightarrow 2^{\mathbf{X}}$ assigns to each $e \in E$ a guard $G(e)$ such that
 - For each $e = (q, q') \in E$, $G(e)$ is a measurable subset of $\partial Inv(q)$ (possibly empty);
 - For each $q \in \mathbf{Q}$, the family $\{G(e) : e = (q, q') \text{ for some } q' \in \mathbf{Q}\}$ is a disjoint partition of $\partial Inv(q)$.
- $R : E \times \mathbf{X} \rightarrow \mathcal{P}(\mathbf{X})$ assigns to each $e = (q, q') \in E$ and $x \in G(e)$ a reset probability kernel on \mathbf{X} concentrated on $Inv(q')$. Here $\mathcal{P}(\mathbf{X})$ denote the family of all probability measures on \mathbf{X} . Furthermore, for any measurable set $A \subset Inv(q')$, $R(e, x)(A)$ is a measurable function in x .

Remark The measurability assumption on R in the preceding definition is made to ensure that the events we encounter later are measurable w.r.t. the underlying σ -field, hence their probabilities make sense.

Definition 2 (Stochastic Execution) A stochastic process $(X(t), Q(t)) \in \mathbf{X} \times \mathbf{Q}$ is called a stochastic execution iff there exists a sequence of stopping times $\tau_0 = 0 \leq \tau_1 \leq \tau_2 \leq \dots$ such that for each $n \in \mathbb{N}$,

- In each interval $[\tau_n, \tau_{n+1})$, $Q(t) \equiv Q(\tau_n)$ is constant, $X(t)$ is a (continuous) solution to the stochastic differential equation

$$dX(t) = f(Q(\tau_n), X(t)) dt + g(Q(\tau_n), X(t)) dB(t)$$

where $B(t)$ is the standard Brownian motion in \mathbb{R} ;

- $\tau_{n+1} = \inf\{t \geq \tau_n : X(t) \notin \text{Inv}(Q(\tau_n))\}$;
- $X(\tau_{n+1}^-) \in G(Q(\tau_n), Q(\tau_{n+1}))$ where $X(\tau_{n+1}^-)$ denotes $\lim_{t \uparrow \tau_{n+1}} X(t)$;
- The probability distribution of $X(\tau_{n+1})$ given $X(\tau_{n+1}^-)$ is governed by the law $R(e_n, X(\tau_{n+1}^-))$, where $e_n = (Q(\tau_n), Q(\tau_{n+1})) \in E$.

Under some not-so-stringent assumptions, the existence and uniqueness of stochastic execution up to distribution can be established. We omit the proof here.

Definition 3 (Embedded Markov Chain) In the notation of previous definition, define $Q_n \triangleq Q(\tau_n)$, $X_n \triangleq X(\tau_n)$. Then $\{(Q_n, X_n), n \geq 0\}$ is called the embedded Markov chain for the stochastic execution $(X(t), Q(t))$.

The following lemmas can be easily checked.

Lemma 1 $\{(Q_n, X_n)\}$ defined above is indeed a MC (Markov chain) with transition probability:

$$\begin{aligned} & P(Q_{n+1} = q', X_{n+1} = x' | Q_n = q, X_n = x) \\ &= \int_{G(e)} R(e, y)(x') P(Y_x(\eta) = y) dy \end{aligned} \tag{1}$$

where $e = (q, q')$. $Y_x(t)$ is the solution to the stochastic differential equation

$$dY(t) = f(q, Y(t)) dt + g(q, Y(t)) dB(t)$$

with initial condition $Y(0) = x$. $\eta = \inf\{t \geq 0 : Y_x(t) \notin \text{Inv}(q)\}$ is the first escape time of $Y_x(t)$ from the invariant set $\text{Inv}(q)$.

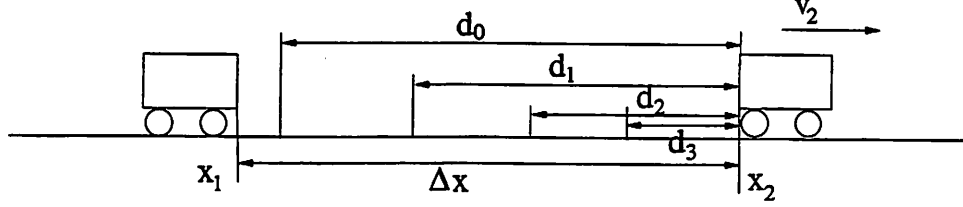


Figure 1: AHS example

Lemma 2 *If the reset kernel $R((q, q'), x) = R(q')$ does not depend on q or x , then $\{Q_n\}$ itself is a MC with transition probability ($n \geq 1$):*

$$P(Q_{n+1} = q' | Q_n = q) = \int_{Inv(q)} P(Y_x(\eta) \in G(e)) R(q)(dx) \quad (2)$$

where e, Y_x, η is defined in the previous lemma. For $n = 0$, the transition probability depends on the initial distribution of $X(0)$.

The reason we introduce the embedded MC is that in most cases, it is hard if not impossible to get an explicit expression of the stochastic execution for a stochastic hybrid system. If all we are interested in is the reachability analysis of the discrete states transitions, then $\{Q_n\}$ will capture all the necessary information. This is the case if a subset of the discrete states is defined to be the “bad” states and a controller is designed to minimize the probability of reaching these states within a given time horizon. Or alternatively, some states are defined to be safe and we want to maximize the probability that the execution will remain in these states for as long as possible. At first sight these observation does not seem to be applicable in general, since in most cases, the definition of bad states and safe states involve both the discrete and continuous states. However, by breaking up the corresponding invariant sets and adding more discrete states and trivial reset kernels, we can always reduce the original system to a new one satisfying the above conditions, at least in the case when the support of any reset kernel is contained exclusively in safe or bad set.

3 A Simple Example

To fix idea, consider the following simple example. Two cars, labeled 1 and 2 with car 2 in the lead, are moving from left to right on a highway (See

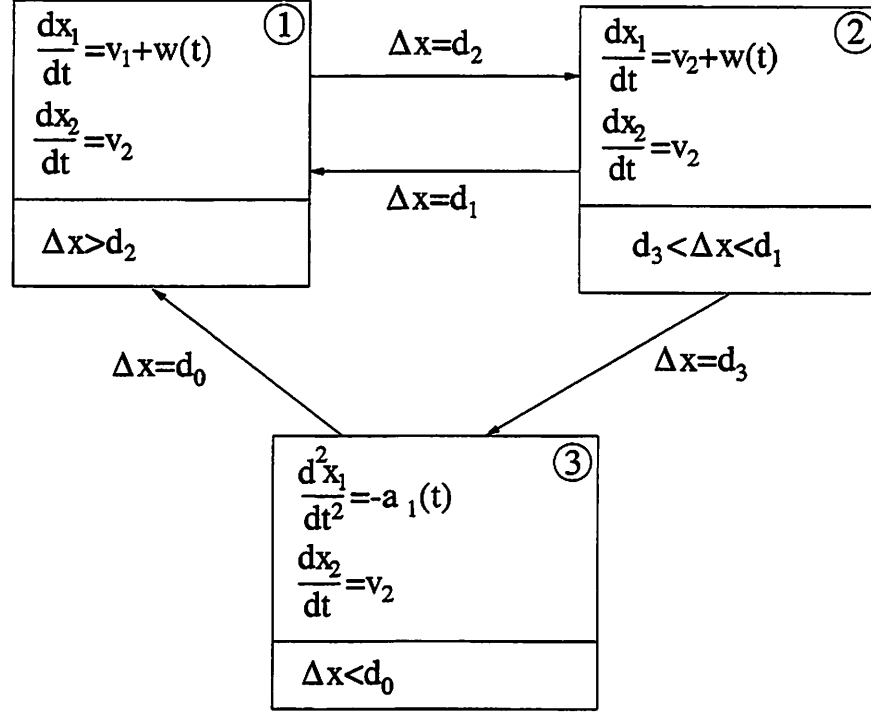


Figure 2: Diagram for the stochastic hybrid system

Figure 1). Due to various random factors such as road condition, wind, and the presence of human being, the motions of both cars are stochastic. If we absorb all the randomness into the motion of car 1 and ignore the possible occurrence of emergency brakings, then the motion of car 2 can be modeled as having a constant speed v_2 .

We propose the following simple control scheme for car 1: Let Δx be the distance between the two cars. Let $d_0 > d_1 > d_2 > d_3 > 0$.

1. **Chasing:** If $\Delta x \geq d_2$, then car 1 will try to move at speed $v_1 > v_2$. So the perturbed motion of car 1 is governed by

$$\frac{dx_1}{dt} = v_1 + w(t)$$

where $w(t)$ is the white noise with power spectral density 1;

2. **Keeping:** If $d_3 \leq \Delta x \leq d_1$, then car 1 will try to move at v_2 ;

3. **Braking:** If $\Delta x \leq d_3$, then car 1 will brakes according to some prescribed procedure until $\Delta x = d_0$.

For a diagram of the corresponding stochastic hybrid system H , see Figure 2. It consists of 3 discrete states $\{1, 2, 3\}$ corresponding to chasing, keeping and braking respectively. The invariant sets and guards for each discrete state are also shown. The reset kernels are trivial, or more precisely, $R(e, x)$ is concentrated at x for any $e = (q, q') \in E$ and any $x \in G(e)$.

It is easily seen that H satisfies the condition of Lemma 2. Hence the successive visits to the discrete states $\{Q_n\}$ is a MC. Actually its probability transition matrix is

$$P = \begin{pmatrix} 0 & 1 & 0 \\ p & 0 & 1-p \\ 1 & 0 & 0 \end{pmatrix}$$

where

$$p = \frac{d_2 - d_3}{d_1 - d_3}$$

The first and third row of P is obvious and the second row follows from the following lemma:

Lemma 3 *Let B_t be a standard BM. For $a, b > 0$, define $T_{-a} = \inf\{t \geq 0 : B_t = -a\}$, $T_b = \inf\{t \geq 0 : B_t = b\}$. Then*

1. $P(T_{-a} < T_b) = \frac{b}{a+b}$;
2. $E(T_{-a} \wedge T_b) = ab$. Here $T_{-a} \wedge T_b$ means $\min(T_{-a}, T_b)$

Proof: See [3]. ■

Calculation shows that the stationary distribution for P is:

$$\pi = \left(\frac{1}{3-p}, \frac{1}{3-p}, \frac{1-p}{3-p} \right)$$

Therefore by ergodicity theorem and strong law of large number [3], the fraction of time the system spends in each discrete state is distributed as:

$$\left(\frac{ET_1}{3-p}, \frac{ET_2}{3-p}, \frac{(1-p)ET_3}{3-p} \right)$$

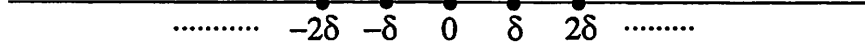


Figure 3: Discretization of state space

where

$$ET_1 = \frac{d_0 - d_2}{v_1 - v_2}, \quad ET_2 = (d_1 - d_2)(d_2 - d_3), \quad ET_3 = t_3$$

are the expected sojourn time in each discrete state respectively.

In practice, we want to maximize the time the stochastic hybrid system spends in the keeping state and minimize the time it spends in braking state. This can be done by adjusting the thresholds d_0, d_1, d_2, d_3 properly. Sometimes this choice is restricted by other physical constraints. However, we can always use more thresholds and thus more complex stochastic hybrid controller to achieve the goal within the various physical constraints. This technique will be illustrated in the next section.

4 State Discretization of 1-D Stochastic Differential Equation

Consider the following stochastic differential equation in \mathbb{R} :

$$\frac{dX(t)}{dt} = f(X(t)) + w(t), \quad X(0) = 0 \quad (3)$$

where $f : \mathbb{R} \rightarrow \mathbb{R}$ is smooth and $w(t)$ is white noise with spectral density 1. Define a series of stopping times as: ($\tau_0 = 0$)

$$\tau_n = \inf\{t \geq \tau_{n-1} : |X(t) - X(\tau_{n-1})| = \delta\}$$

Let $S_n = X(\tau_n)$. Then $\{S_n\}$ is a MC taking values in $\delta \cdot \mathbb{Z}$. S_n captures many sample path properties of the solution process $X(t)$, for example, whether $X(t)$ is recurrent. or less obviously, whether $X(t)$ crosses an interval of length less than δ infinitely many times,

Define $\tau_t = \sup_n\{\tau_n : \tau_n \leq t\}$ and let $Y_t = X(\tau_t)$. Then Y_t is piecewise constant with value S_n in time interval $[\tau_n, \tau_{n+1})$. Define $Z(t)$ to be the solution process to the stochastic differential equation:

$$\frac{dZ(t)}{dt} = f(Y_t) + w(t) \quad (4)$$

Comparing equation (3) and (4) and noticing that during time interval $[\tau_n, \tau_{n+1})$, $|X(t) - Y_t| \leq \delta$ by the definition of τ_n 's and f is continuous, we can expect that as $\delta \rightarrow 0$, $Z(t)$ approaches $X(t)$ in distribution, hence $Z(t)$ is a good approximation to $X(t)$ which is often impossible to calculate explicitly. However, it is still difficult to solve equation (4) since Y_t depend on the original solution process $X(t)$ through τ_n 's and S_n 's. So to solve equation (4), theoretically we still have to solve equation (3) first.

One way to get out of this loop is to use the fact that $X(t)$ can be approximated by $Z(t)$, hence τ_n 's and S_n 's can also be approximated by the corresponding random variables defined from $Z(t)$. This will lead to the discretized stochastic hybrid system (DSHS) defined below.

Definition 4 (Discretized Stochastic Hybrid System) *The discretized stochastic hybrid system for equation (3) is $H = (Q, X, Inv, f, g, G, R)$ where*

- $Q = \mathbb{Z}$, $X = \mathbb{R}$;
- $Inv(k) = ((k-1)\delta, (k+1)\delta)$ for any $k \in Q$;
- $f(k, \cdot) = f(k\delta)$, $g(k, \cdot) = 1$ are constant functions;
- $G(k, k-1) = \{(k-1)\delta\}$, $G(k, k+1) = \{(k+1)\delta\}$ are singleton and $G(k, k) = \emptyset$ otherwise;
- Reset kernels are trivial.

Since H satisfies the condition of Lemma 2, $\{Q_n\}$ defined as in section 2 is a MC. By discussion at the beginning of this section, it is expected that $\{Q_n\}$ approximates the MC $\{S_n\}$ defined from the solution $X(t)$ to equation (3). In the following development, we will use H_δ to stress the dependency of H on the discretization step δ if necessary.

Obviously $\{Q_n\}$ has probability transition matrix of the form:

$$Q = \begin{pmatrix} \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & p_{-1} & 0 & q_{-1} & 0 & 0 & \dots \\ \dots & 0 & p_0 & 0 & q_0 & 0 & \dots \\ \dots & 0 & 0 & p_1 & 0 & q_1 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

or equivalently, $\forall i, j \in \mathbb{Z}$,

$$q(i, j) = \begin{cases} p_i & \text{if } j = i - 1; \\ q_i = 1 - p_i & \text{if } j = i + 1; \\ 0 & \text{otherwise.} \end{cases}$$

The following lemma is needed in the calculation of p_k 's.

Lemma 4 *Let B_t be a 1-D standard BM with $B_t = 0$. Then $B_t^\mu = B_t + \mu t$ is the BM with drift μ . For any $b \in \mathbb{R}$, define $T_b^\mu = \inf\{t \geq 0 : B_t^\mu = b\}$ to be the first time B_t^μ hits b . Then*

$$P(T_{-b}^\mu < T_b^\mu) = P(T_{-1}^{b\mu} < T_1^{b\mu}) = P(T_{-b\mu}^1 < T_{b\mu}^1)$$

In other word, $P(T_{-b}^\mu < T_b^\mu)$ only depends on $b\mu$.

Proof: This is a direct implication of the scaling invariant property of BM (see [3]), i.e. for any $\lambda > 0$, $\{\frac{1}{\lambda}B(\lambda^2 t), t \geq 0\}$ has the same distribution as $\{B_t, t \geq 0\}$. ■

We will denote $\phi(b\mu) = P(T_{-1}^{b\mu} < T_1^{b\mu})$. Then $\phi : \mathbb{R} \rightarrow (0, 1)$ is decreasing and $\phi(0) = \frac{1}{2}$ by symmetry. The exact closed form expression of ϕ is difficult to obtain, yet there are various ways to estimate it.

Coming back to the calculation of p_k . Since the solution to the stochastic differential equation

$$dY(t) = f(k\delta) dt + dB_t, \quad Y(0) = k\delta$$

is $Y(t) = k\delta + f(k\delta)t + B_t$, by equation (2),

$$\begin{aligned} p_k &= P(Q_{n+1} = k - 1 | Q_n = k) \\ &= P(Y(t) \text{ reaches } (k - 1)\delta \text{ before it reaches } (k + 1)\delta) \\ &= \phi[\delta f(k\delta)] \end{aligned} \tag{5}$$

After the probability transition matrix Q is obtained, the natural questions we will ask ourself are: Under what conditions is $\{Q_n\}$ recurrent? transient? Are these conditions connected in any way with the stability of the deterministic part of equation (3)?

Since we won't have enough space and time for a thorough discussion, we restrict our attention on the case when f is odd, i.e. $f(-x) = -f(x)$. In this case, $p_k = q_{-k}$ for $k \in \mathbb{Z}$ (In particular, $p_0 = q_0 = \frac{1}{2}$).

Define $G_n = |Q_n|$, $n = 0, 1, 3, \dots$, then by symmetry, we have

Lemma 5 *For any initial distribution of $\{Q_n\}$, $\{G_n\}$ is a MC with state space \mathbb{N} and transition probability matrix G defined by*

$$g(i, j) = \begin{cases} 1 & \text{if } i = 0, j = 1; \\ p_i & \text{if } i > 1, j = i - 1; \\ q_i & \text{if } i > 1, j = i + 1; \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, $\{G_n\}$ is (postive) recurrent if and only if $\{Q_n\}$ is.

Notice that the transition matrix G has the property $g(i, j) = 0$ when $|i - j| > 1$. This kind of chain is called *birth and death chain*. The following lemma is a standard result from probability theory (see [3]):

Lemma 6 *Let $p_0 = 0$, define a sequence*

$$\psi(n) = \sum_{m=0}^{n-1} \prod_{j=1}^m \frac{q_j}{p_j}$$

(The product is interpreted as 1 when $m = 0$). Then the birth and death chain $\{G_n\}$ is recurrent if and only if $\psi(n) \rightarrow \infty$ as $n \rightarrow \infty$.

Assembling what we have so far, we get

Theorem 1 (Recurrency of DSHS) *Under the assumptions on f in the preceding paragraphs, the embedded MC of the discretized hybrid system is recurrent if and only if*

$$\psi(n) \triangleq \sum_{m=0}^{n-1} \prod_{j=1}^m \frac{1 - \phi[\delta f(k\delta)]}{\phi[\delta f(k\delta)]} \rightarrow 0 \quad \text{as } n \rightarrow \infty \quad (6)$$

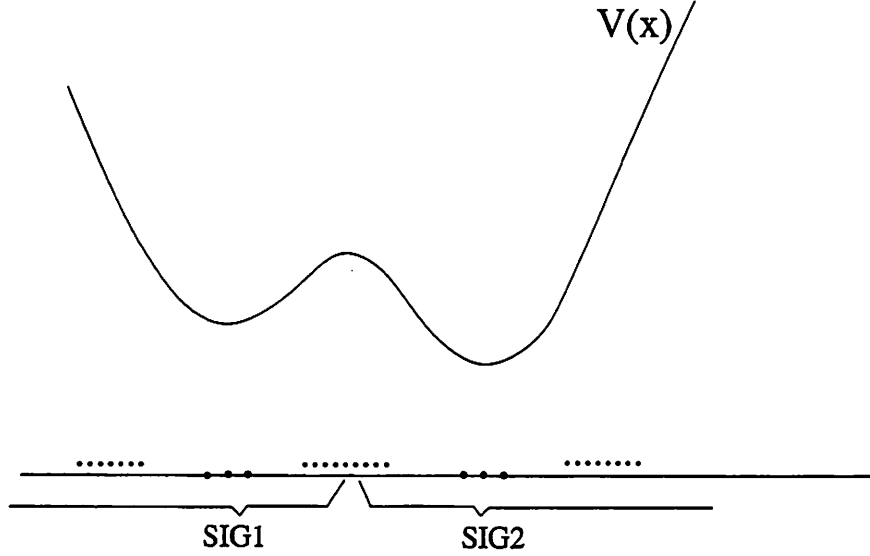


Figure 4: DSHS for a gradient system

5 Future Direction

Several possible extensions to the above effort are listed below:

- The criteria proposed in (6) is not easy to check for a given f since our knowledge of ϕ is limited. However it is a very interesting (and challenging as well) problem to estimate the rate at which $\phi(x) \rightarrow \frac{1}{2}$ as $x \rightarrow 0$, since that is where the boundary between recurrent and transient lies. In other words, we are interested in how fast should $f(x)$ decay to 0 as $x \rightarrow \infty$ such that $\{Q_n\}$ is still recurrent. (If f has compact support, *i.e.* if $f(x) = 0$ for x outside a compact subset of \mathbb{R} , then it is readily shown that $\{Q_n\}$ is recurrent but not positive recurrent which is the boundary case).
- If the deterministic part of equation (3) is a gradient system ([6]) of the form:

$$\frac{dx}{dt} = f(x) = -\text{grad } V(x) \quad (7)$$

for some $V \in C^2(\mathbb{R})$, then each local minimum of $V(x)$ corresponds to a *strongly interacting group* (SIG) of the embedded MC $\{Q_n\}$ of

the DSHS H_δ (See Figure 4). A typical execution of $\{Q_n\}$ is as the following: it jumps inside one SIG for some time and then jumps to another SIG and so on. The mean sojourn time in each SIG is of interest to us since usually we will make some of the SIG's our "desired" valley, while some others the undesired trap.

References

- [1] E. Altman and V. Gaitsgory. Asymptotic optimization of a nonlinear hybrid system governed by a Markov decision process. *SIAM Journal of Control and Optimization*, 35(6):2070–2085, 1997.
- [2] Gopal K. Basak, Arnab Bisi, and Mrinal K. Ghosh. Ergodic control of random singular diffusions. In *IEEE Conference on Decision and Control*, pages 2545–2550, Kobe, Japan, 1996.
- [3] Richard Durrett. *Probability: theory and examples, 2nd edition*. Duxbury Press, 1996.
- [4] J.A. Filar and V. Gaitsgory. Control of singularly perturbed hybrid stochastic systems. In *IEEE Conference on Decision and Control*, pages 511–516, Kobe, Japan, 1996.
- [5] John Lygeros. *Hybrid systems: modeling, analysis and control*. preprint, 1999.
- [6] L. Perko. *Differential equation and dynamical systems, 2nd edition*. Springer-Verlag, 1996.
- [7] Ching-Chih Tsai. Composite stabilization of singularly perturbed stochastic hybrid systems. *International Journal of Control*, 71(6):1005–1020, 1998.
- [8] Ching-Chih Tsai and Abraham H. Haddad. Averaging, aggregation and optimal control of singularly perturbed stochastic hybrid systems. *International Journal of Control*, 68(1):31–50, 1997.

**Vehicle String Models:
Collision Analysis for an Emergency Deceleration Maneuver**

**Laura M. Muñoz
EE291E Final Project Report
Due: May 17, 1999**

Introduction

Two vehicle string models were investigated. In [1], a representation of a string of vehicles was developed using the Hybrid Input / Output Automaton formalism. Necessary and sufficient conditions were derived for vehicle safety in the context of an emergency deceleration maneuver. A description based on interactions of a pair of vehicles was presented in [2], along with computational tools designed for collision and throughput analysis of an automated highway. Numerical experiments were conducted to estimate the number and severity of collisions that could result from an emergency braking maneuver. An attempt was made to compute collision statistics via random sampling of the braking distributions of [2], for the case of 2 vehicles. Time was invested towards the development of a vehicle string simulation in SHIFT (a hybrid systems modeling language), but the code requires further troubleshooting before it can produce any meaningful results.

Vehicle String Model

The vehicle string model of [1] was developed in the Hybrid Input/Output Automaton (HIOA) modeling formalism. From [3], a hybrid I/O automaton is described as follows:

$H = (U, X, Y, \Sigma^{in}, \Sigma^{int}, \Sigma^{out}, \Theta, D, W)$, where

$V = U \cup X \cup Y$ is the set of all variables. U, X , and Y are input, internal, and output variables, respectively,

V is the set of valuations for the variables in V ,

$\Sigma = \Sigma^{in} \cup \Sigma^{int} \cup \Sigma^{out}$ is the set of *actions* for H . For an HIOA, actions encode the discrete dynamics of the system.

Thus, when an action occurs, the system state jumps immediately to a new value. $\Sigma^{in}, \Sigma^{int}, \Sigma^{out}$ are the input, internal, and output actions, respectively,

$\Theta \subseteq V$ is the set of initial states,

$D \subseteq V \times \Sigma \times V$ is the set of discrete transitions for the automaton, and

W is the set of trajectories over V . In this formalism, a trajectory is a mapping from an interval of the time axis to some $Z \subseteq V$. For a trajectory to be in W , it must satisfy three axioms (existence of point trajectories, closure under subintervals of time, and prefix closure of trajectories). There are also restrictions on Θ and D which I will not reiterate here. According to [3], the HIOA formalism is useful for performing compositions of hybrid automata, and for carrying out analysis of hybrid systems at various levels of abstraction.

The vehicle string model developed by Lygeros & Lynch in [1] represents N vehicles traveling one behind another in a single lane, with the first vehicle labeled 0 and the last $N-1$. The plant is a hybrid I/O automaton,

$$P = (U_p, X_p, Y_p, \Sigma_p^{in}, \Sigma_p^{int}, \Sigma_p^{out}, \Theta_p, D_p, W_p).$$

The set of input variables is $U_p = \{u\}$, $u = [u_0, \dots, u_{N-1}]$, where u_i is the commanded acceleration for vehicle i . The commanded acceleration is assumed to lie within a closed interval, $u_i(t) \in [a_i^{min}, a_i^{max}]$. Another assumption on the model is that both negative and positive acceleration values are allowed: $a_i^{min} < 0 < a_i^{max}$. $X_p = \{x, acc, Touching\}$

is the set of internal variables, with $x = [[\Delta x_0 \ v_0], \dots, [\Delta x_{N-1} \ v_{N-1}]]$, $acc = [acc_0, \dots, acc_{N-1}]$, and $Touching = \{Touching_1, \dots, Touching_{N-1}\}$. The variable Δx_i is the spacing between vehicles i and $i-1$, and v_i is the velocity of vehicle i . The actual acceleration acc_i of vehicle i may differ from the commanded acceleration u_i if other vehicles are in contact with i . $Touching$ is a set of Boolean variables; $Touching_i$ is true whenever vehicle i is touching vehicle $i-1$. The automaton has no input or output actions: $\Sigma_P^{in} = \Sigma_P^{out} = \emptyset$, but does have internal actions: $\Sigma_P^{int} = \{Collision_1, \dots, Collision_{N-1}, Touch_1, \dots, Touch_{N-1}, Separate_1, \dots, Separate_{N-1}\}$. The model is such that an action takes place whenever its precondition (described on the valuation space V_P) becomes true, at which point continuous evolution stops in order to allow the action to occur. The effect of an action (in $V_P \times V_P$) describes allowable states before and after the transition. It is assumed that the initial state of the plant satisfies $\Delta x_i(0) \geq 0$, $v_i(0) \geq 0$, and $Touching_i(0) = \text{False}$.

A definition for string safety is established in [1]. Based earlier safety analysis, the threshold for “safe” collisions is taken to be $v_A = 3\text{m/s}$. That is, any collisions occurring with relative velocity at or below v_A are considered safe. More formally, a string is defined to be *safe* if, for all $i=1, \dots, N-1$, $[(\Delta x_i = 0) \Rightarrow (v_i \leq v_{i-1} + v_A)]$ is an invariant property. A *property* is a function on V_P (the set of valuations of the state variables) that maps states of P to $\{\text{True}, \text{False}\}$, and a property is *invariant* if it is true at all reachable states.

The question of safety is important when considering emergency braking of a string of vehicles. In the situation proposed in [1], the first vehicle in the string applies its maximum deceleration until it comes to a stop, perhaps due to a malfunction or other abnormal condition. This presents hazard to the remaining vehicles in the string, as it is possible that the followers may not have greater braking capability than the leader. The emergency deceleration maneuver used for analysis in [1] was characterized by the following default deceleration strategy: (1) at time $t=0$, lead vehicle applies maximum deceleration a_0^{\min} until it stops (at which point its commanded acceleration becomes 0); (2) after delay d_i , i^{th} vehicle applies maximum deceleration a_i^{\min} . Lygeros and Lynch arrived at safety conditions for a string of length N . Some preliminary definitions:

Near-uniform-mass string: all coefficients of restitution are equal ($\alpha_k \equiv \alpha$), and $\alpha M_{k-1} \leq M_k \leq M_{k-1}/\alpha$. If this condition is satisfied, vehicles will not end up going backward due to a collision. The minimum and maximum values of the deceleration are $\hat{a}_{\min} = \min_{0 \leq k \leq N-1} a_k^{\min}$, $\hat{a}_{\max} = \max_{0 \leq k \leq N-1} a_k^{\min}$

$$\Delta x_{ij} = \sum_{k=i+1}^j \Delta x_k$$

Sufficient Condition: A near-uniform-mass string of N vehicles is safe if initially $P(\Delta x_{ij}, v_i, v_j) = (v_j - (\hat{a}_{\max} / \hat{a}_{\min}) v_i - v_A) \leq 0$ for all i, j with $0 \leq i < j \leq N-1$.

The sufficient condition can be proved by induction.

Necessary Condition: If all strings of N vehicles satisfying $a_i^{\min} \in [a_l, a_u]$, $M_i \in [M_l, M_u]$, and $\alpha_i \equiv 1$, are safe under the default deceleration strategy, then initially $(P_1(\Delta x_{ij}, v, v)) \vee (P_2(\Delta x_{ij}, v, v))$ is true for all i, j with $0 \leq i < j \leq N-1$ and for all $a_i^{\min}, a_j^{\min} \in [a_l, a_u]$.

The necessary condition can be proved constructively. According to [1], this condition essentially means that if any two vehicles in the string are not safe, then string may be unsafe.

Collision Analysis

In the vehicle model used by Godbole and Lygeros in [2], four vehicles follow one another in a single lane. The vehicles are labeled C, A, B, D, where D is the lead vehicle. A fourth-order model was used to describe the interaction of the pair A-B using state vector

$$x = \begin{bmatrix} \dot{x}_A & \ddot{x}_A & D_{AB} & \dot{D}_{AB} \end{bmatrix}^T = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$$

The velocity and acceleration of vehicle A are denoted by x_1 and x_2 , respectively. The spacing and relative velocity between A and B are given by x_3 and x_4 . The system can be modeled as a hybrid automaton, as in [4], with 3 discrete states: one in which A is moving without touching B, another where both vehicles are moving while pushing against one another, and one where A is stopped. The form of the continuous dynamics depends on the discrete state of the automaton. When the vehicles are moving freely, the acceleration of each vehicle is given by the commanded acceleration; in the state where the vehicles are touching and pushing against one another, the acceleration of each vehicle may be modeled as a weighted average of the commanded accelerations, as in [1]. Another kind of hybrid representation could include 2^{N-1} discrete states for a string of N vehicles; one state accounts for the case where all vehicles are moving freely without touching, and the other states give all possible combinations of touching vehicles.

The collision tool was used in [2] to estimate the number and severity of collisions for a string of vehicles undergoing an emergency deceleration maneuver. The default deceleration strategy was: at time $t=0$, the lead vehicle applies maximum deceleration a_0^{\min} , and (for a hop-by-hop communication architecture) after a delay $\delta_i = i\delta$, the i^{th} vehicle applies maximum deceleration a_i^{\min} (similar to the strategy proposed in [1]). The collision tool of [2] takes as input initial spacings, velocities, acceleration levels, masses, delays, and coefficients of restitution for the vehicles. It determines the smallest collision time, solves for states of vehicles after collision, and iterates until no more collisions are possible. The output includes relative velocities at impact for all collisions. To perform a statistical analysis of collisions caused by emergency deceleration, knowledge of the braking capability distribution for the vehicles was required.

Pessimistic and optimistic braking distributions were compiled, as shown in [2] p.12. To obtain collision statistics, the collision tool was run for all possible assignments of deceleration capability for N vehicles. This was possible since the probability distributions were divided into a finite number of bins; only a finite number of capability values were considered. Exact expected values for collision data were calculated by weighting the outcomes by their corresponding probabilities. For either distribution, there were 11 possible values for maximum braking, which required 11^N collision tool runs to capture all scenarios.

An alternative method proposed by [5] was to estimate collision statistics by random sampling of the capability distribution. This is related to the idea of Monte Carlo simulation, wherein the expected value of a function f of an N -dimensional random variable x is estimated by taking m random samples of x : x_1, \dots, x_m , and calculating the average value $\hat{E} = (1/m)(f(x_1) + \dots + f(x_m))$. The estimate is then within ϵ of the true value E

($|E - \hat{E}| \leq \epsilon$) with a probability of at least $1 - 2e^{-2m\epsilon^2}$ [6]. If this reasoning holds for collision calculations, the

random sampling method would have the advantage of being independent of number of vehicles N , with a corresponding decrease in confidence in the accuracy of the results.

A simplified collision analysis was performed for the case $N=2$ using random sampling of the capability distributions. The initial conditions were the same as in [2], p.16; velocities $v_A(0) = v_B(0) = 25$ m/s, accelerations $a_A(0) = a_B(0) = 0$ m/s², initial spacing = 1m, initial relative velocity = 0 m/s. The setup was as before with vehicle A following vehicle B. At $t=0$, B applies maximum deceleration a_B^2 (superscript, not squared). A applies its maximum deceleration (a_A^2) $\delta = 0.05$ seconds later. A Matlab function was coded to solve the spacing equation $x_3(t) = (a/2)t^2 + bt + c + x_3(0)$ for the times of possible collisions given input deceleration values a_A^2 and a_B^2 . In the event of a collision, the relative velocity was calculated from $x_4(t) = at + b$. The coefficients a , b , and c were special cases of the parameters on p. 26 of [2], in agreement with the preceding initial conditions.

The probability distributions (shown in Figures 1 and 2) were represented in Excel based on my visual approximation of the distributions of [2]. The discrete probability values sum to 1 for both cases, but it is not clear how close these values are to the original distributions. Each selected deceleration value was my estimate of the number occurring at the center of the corresponding bar in [2]. For each discrete density function, 50 random (a_A^2 , a_B^2) pairs were generated and input to the collision calculator. Some results are listed in Table 1, along with values from [2].

Table 1: Collision Statistics

	Optimistic (random)	Pessimistic (random)	Pessimistic (combinatorial)	Ref. [2] ($N=2$)
Worst-case c. speed	-1.4146	-1.6920	-1.6920	-2.4
$E(v_{rel})$	-0.7153	-0.7882	-0.4134	-0.8955
Prob of no collision	0.46	0.44	0.4545	0.41

The first two columns show results for random sampling of the braking distributions, for the optimistic and pessimistic cases, respectively. The values in the third column are a result of my interpretation of the method used in [2], with the pessimistic distribution of Figure 1. The collision calculation was performed for all possible deceleration assignments (121 calculations for $N=2$).

The first row lists the worst-case collision speed; for columns 1-3, this was calculated as the v_{rel} resulting from assigning A the worst possible deceleration and B the best possible deceleration for the specified distribution. Thus it was not calculated as a probabilistic quantity. The worst-case speed in [2] has magnitude 1.4 times larger than that obtained from the distribution of Figure 1; this could indicate a large discrepancy in my estimation of the support of the distribution, or there could be an error in the collision calculator. All worst-case collision speeds are below the “safe” threshold of 3m/s.

In the second row, the expected value of the relative velocity at collision is given. For columns 1 and 2, this value was calculated by averaging v_{rel} for all collisions that did occur. For the combinatorial case, the expected value was calculated as a sum of the v_{rel} for all collisions, with each value weighted by the probability of its occurrence. I assumed that the deceleration assignments were independent, so the joint probability of a given pair was computed by $P(a_A^2 = X, a_B^2 = Y) = P(a_A^2 = X)P(a_B^2 = Y)$. It was hoped that the expected v_{rel} values of columns

2 and 3 would be in better agreement. Possible sources of error include a mistake in my method of calculating

$E(v_{rel})$ for the combinatorial case, or a problem with the generation of random values from the pessimistic

distribution. The EV in column 4 was estimated from the $N=2$ relative velocity distribution of [2].

The third row shows the probability of no collision occurring. For columns 1-3 this was taken as the no-

collision frequency (total number of non-collisions / number of trials). The fact that these numbers are all near 40%

does not inspire much confidence in light of other discrepancies with results of [2].

Due to the spacing between discrete values of the distributions, some collision scenarios (accounted for by

the collision calculator) were not realizable. For instance, it is possible for A and B to collide when the deceleration

of A is slightly more negative than the deceleration of B (i.e., even if A can decelerate more quickly than B, it is not

always enough to prevent a collision). This situation can not be observed under the current discretization.

Future work should include further development and debugging of the collision calculator. It was intended

to compute collisions for longer strings ($N > 2$) by resolving collisions pairwise until it was no longer possible for any

vehicle to overtake another. The calculator must be enhanced to include the momentum and restitution equations so

that the state of vehicles after collision can be known. Eventually, a completed vehicle string model in SHIFT could

be used to conduct statistical trials with random assignment of braking capabilities.

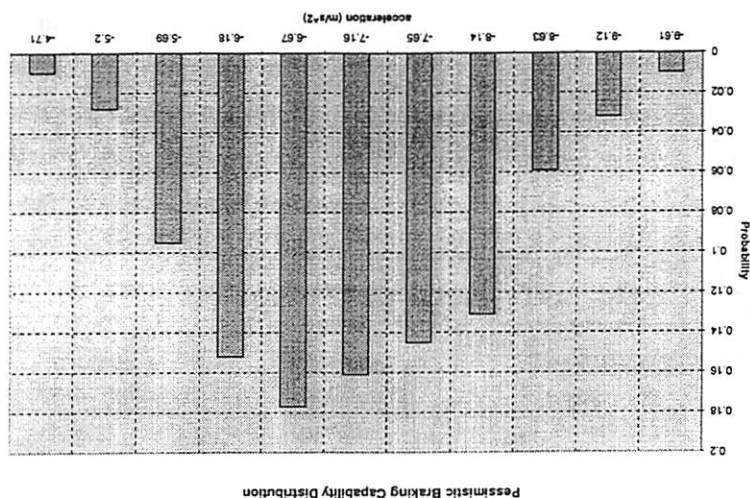


Figure 1

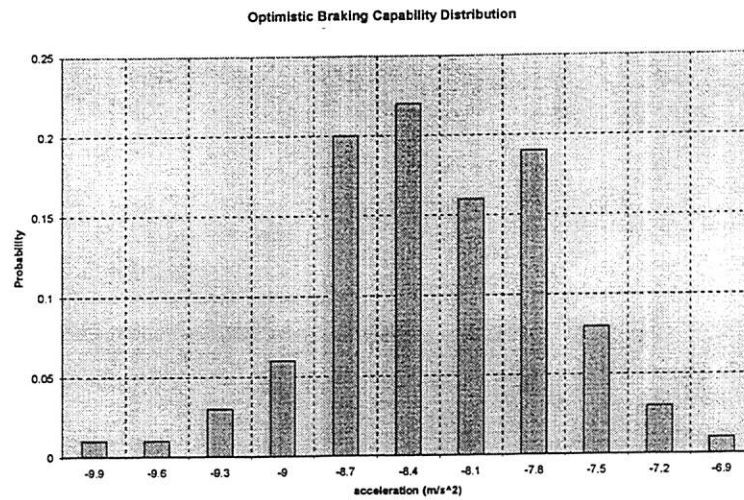


Figure 2

References:

1. J. Lygeros and N. Lynch, "Strings of Vehicles: Modeling & Safety Conditions," in *Hybrid Systems: Computation and Control* (S. Sastry and T. Henzinger, eds.), no. 1386 in LNCS, pp.273-288, Springer Verlag, 1998.
2. D. N. Godbole and J. Lygeros, "Safety & Throughput Analysis of Automated Highway Systems," in *Transportation Research, C*, 1999. Accepted for publication, to appear in 1999.
3. N. Lynch, R. Segala, F. Vaandrager, H. B. Weinberg, "Hybrid I/O Automata," in R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control* (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, October 1995), volume 1066 of Lecture Notes in Computer Science, pages 496-510. Springer-Verlag 1996.
4. J. Lygeros and D.N. Godbole, "Verified Hybrid Controllers for Automated Vehicles," in *IEEE Transactions on Automatic Control*, vol. 43, no. 4, April 1998.
5. J. Lygeros.
6. M. Vidyasagar, *A Theory of Learning and Generalization*, Springer-Verlag, 1997. (one page)

```

function out = coll_1(accn2A, accn2B)

%calculate relative velocity of
%collisions that result from emergency deceleration maneuver

%acceleration trajectories
%maximum deceleration values are assumed to be negative

a1_A = 0;
a2_A = accn2A;
a1_B = 0;
a2_B = accn2B;

%initial conditions
x1_A = 25; %initial velocity of A (m/s)
x2_A = 0; %initial acceleration of A (m/s^2)
D_AB = 1; %initial spacing (m)
D_AB_dot = 0; %initial relative velocity (m/s);
m_A = 1500; %mass of vehicle A (kg)
m_B = 1500; %mass of vehicle B (kg)
alpha = 1; %coefficient of restitution
del = 0.05; %delay (sec)

A = (a2_B - a2_A)/2;
B = a2_A*del;
C = 1 - a2_A*del^2/2;
dis = B^2 - 4*A*C;
t_temp = sqrt(-2*D_AB/a2_B);

if (del ~= 0)
    aB_critical = -2/del^2;
else
    aB_critical = 0;
end

collision_flag = 0;

%first check if a collision can occur before del seconds
%have elapsed (before A begins to brake)

if (t_temp <= del)
    t_c = t_temp; %time of collision
    v_rel = a2_B*t_c; %relative velocity at collision time
    collision_flag = 1;
else
    %find any collision times that occur after del seconds
    if (A == 0) & (B ~= 0)
        t_c = -C/B;
        collision_flag = 2;
    elseif (A == 0) & (B == 0) %no solution
        t_c = 0;
        collision_flag = 3;
    elseif (dis < 0) %roots will be imaginary
        t_c = 0;
        collision_flag = 4;
    elseif (dis == 0)
        t_c = -B/(2*A);
        collision_flag = 5;
    elseif (dis > 0)
        t_c = (-B - sqrt(B^2 - 4*A*C))/(2*A);
        collision_flag = 6;
    end

    if (t_c > 0)
        v_rel = ((a2_B - a2_A)/2)*t_c + a2_A*del;
    else
        v_rel = 0;
    end
end

```

end

out = [collision_flag t_c v_rel del aB_critical];

EE291e Project: LMI Conditions for the Stability of Hybrid Systems

John Musacchio

May 10, 1999

1 Introduction

It is simple to show the stability of linear systems using a LMI. One only need only find a P such that

$$A^T P + P A < 0 \quad (1)$$

The purpose of this work is to find similar LMI conditions that guarantee the stability of more complex systems. In particular, we focus on hybrid systems with vector fields characterized by linear differential inclusions, affine guards, and affine reset relations. This work is an extension of [1] and [2]. These works develop LMI conditions for the stability of piecewise linear systems, and hybrid systems with trivial reset relations (trivially meaning the reset relation is equal to the identity map.) Therefore the main contributions of this work is the ability to deal with non-trivial reset relations, and also the fact that the development has been put into the notation used throughout the EE291e course.

We begin by formulating the problem in Section 2. In Section 3 we develop the LMI conditions in a casual, easy to understand manner. In Section 4, we state the theorem, and prove the theorem in Section 5. Note that most of the proof given in Section 5 is original work. Section 6 gives an example of how to apply the theorem. Finally, Section 7 makes some final remarks, and discusses some of the other directions that were explored while doing this project.

2 Problem Formulation

Consider the hybrid automaton $H = (Q, X, f, \text{Init}, I, E, G, R)$ with the following properties:

- Define $Q_0 = \{q \in Q : 0 \in I(q)\}$, $Q_1 = Q/Q_0$
 $I(q) \subseteq \{x : \bar{L}_q \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0, x \in X\}$ where $\bar{L}_q \begin{bmatrix} x \\ 1 \end{bmatrix}$ is vector valued and \geq is defined element by element.

Also, $\bar{L}_q = [L_q \quad l_q]$ and $l_q = 0$ whenever $q \in Q_0$

- $f(q, x) \in \bar{c} \bar{o}_{k \in K(q)} \{A_{qk} + a_{qk}\}$
 where $a_{qk} = 0$ if $q \in Q_0$ and $\bar{A}_{qk} \triangleq \begin{bmatrix} A_{qk} & a_{qk} \\ 0 & 0 \end{bmatrix}$
- $\forall (q, r) \in E$
 $G(q, r) = \{x : m_{qr}^T x = 0\}$ if $q \in Q_0$ and $r \in Q_0$
 $G(q, r) = \{x : \bar{m}_{qr}^T \begin{bmatrix} x \\ 1 \end{bmatrix} = 0\}$ if $q \in Q_1$ or $r \in Q_1$
- $\forall (q, r) \in E$
 $R(q, r, x) = N_{qr} x$ if $q \in Q_0$ and $r \in Q_0$
 $\begin{bmatrix} R(q, r, x) \\ 1 \end{bmatrix} = \bar{N}_{qr} \begin{bmatrix} x \\ 1 \end{bmatrix}$ if $q \in Q_1$ or $r \in Q_1$

Our problem will be to find sufficient conditions that guarantee the exponential stability of H . Our approach will be to show the existence of Lyapunov function, $V(q(t), x(t))$ with the following properties:

- $\alpha \|x(t)\|^2 < V(x(t), q(t)) < \beta \|x(t)\|^2$
- $\frac{d}{dt} V(x(t), q(t)) < -\gamma \|x\|^2$ for almost all t
- $V(x(t), q(t))$ decreases during Discrete Transitions

In the section that follows, we will derive sufficient conditions for the existence of a Lyapunov function that satisfies each of the above three properties.

3 Development of Conditions

In the development that follows, we consider Lyapunov functions of the form:

$$V(x, q) = \begin{cases} x^T P_q x & \text{if } q \in Q_0 \\ \begin{bmatrix} x^T & 1 \end{bmatrix} \bar{P}_q \begin{bmatrix} x \\ 1 \end{bmatrix} & \text{if } q \in Q_1 \end{cases}$$

We will motivate the conditions in the upcoming theorem statement in the remainder of this section by just studying the discrete states in Q_0 . A more rigorous treatment is given in Section 5 where the theorem is proven.

3.1 Conditions For $\alpha\|x(t)\|^2 < V(x(t), q(t))$

One sufficient condition to guarantee that there exists an α such that $\alpha\|x(t)\|^2 < V(x(t), q(t))$ would be if

$$0 < P_q \quad \forall q \in I(q) \quad (2)$$

However, this condition is stronger than necessary. We only need $x^T P_q x > 0$ for all $x \in I(q)$. If we knew of some matrix S with $x^T S x > 0$ for all $x \in I(q)$ and possibly < 0 outside $I(q)$ we could write a LMI that might possibly be easier to satisfy.

$$0 < P_q - S$$

We Can Parameterize a family of matrices to serve as S

$$x^T L_q^T W_q L_q x > 0 \quad \forall x \in I(q)$$

Where W_q is our “free” matrix and is symmetric with all non-negative entries.

Thus a less conservative LMI condition than (2) is

$$0 < P_q - L_q^T W_q L_q \quad (3)$$

We illustrate why this is less conservative than (2) with an example. Consider a discrete state q with $I(q)$ defined as

$$I(q) = \{x : L_q x \geq 0\}$$

$$L_q \triangleq \begin{bmatrix} 1 & -2 \\ 1 & 2 \end{bmatrix}$$

The $I(q)$ region is shown as the shaded area of Figure 1. A valid choice for P_q is

$$P_q = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Even though its not p.d., the value of $x^T P_q x$ is positive for $x \in I(q)$, as illustrated in Figure 1. Indeed Equation (3) allows for such a P_q because we can find a value of W_q that makes $P_q - L_q^T W_q L_q$ positive definite.

$$0 < P_q - L_q^T \begin{bmatrix} 0 & 0.25 \\ 0.25 & 0 \end{bmatrix} L_q \quad (4)$$

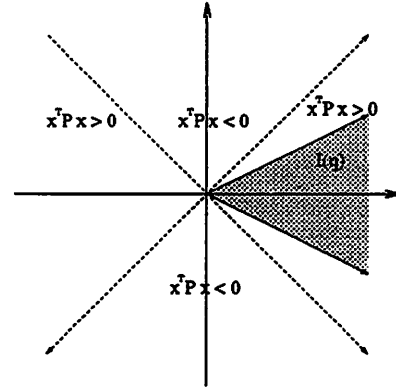


Figure 1: $I(q)$ and a P_q that is not p.d.

3.2 Conditions for $\frac{d}{dt}V(x(t), q(t)) < -\gamma\|x\|^2$ for almost all t

One sufficient condition to guarantee that there exists a γ such that $\frac{d}{dt}V(x(t), q(t)) < -\gamma\|x\|^2$ for almost all t would be if

$$0 > A_{qk}^T P_q + P_q A_{qk} \quad \forall k \in K(q) \quad (5)$$

Yet we only need $\frac{d}{dt}(x^T P_q x) < 0$ for all $x \in I(q)$. If we knew of some matrices S_k with $x^T S_k x > 0$ for all $x \in I(q)$ and possibly < 0 outside $I(q)$ we could write a less strict LMI

$$0 > A_{qk}^T P_q + P_q A_{qk} + S_k \quad \forall k \in K(q)$$

We Can Parameterize a family of matrices to serve as S_k

$$x^T L_q^T U_{qk} L_q x > 0 \quad \forall x \in I(q)$$

Where U_{qk} is a “free” matrix and is symmetric with all non-negative entries. Thus a less conservative LMI condition than (5) is

$$0 > A_{qk}^T P_q + P_q A_{qk} + L_q^T U_{qk} L_q \quad \forall k \in K(q)$$

3.3 Conditions for $V(x, q)$ to decrease during Discrete Transitions

Suppose the event (q, r) occurs at time τ'_i . To meet our condition, we require:

$$0 < x(\tau'_i)^T P_q x(\tau'_i) - x(\tau_{i+1})^T P_r x(\tau_{i+1})$$

Substituting the Reset Relation,

$$0 < x(\tau'_i)^T P_q x(\tau'_i) - x(\tau_i)^T N_{qr} P_r N_{qr} x(\tau_i)$$

This Suggests the following LMI Test to guarantee that $V(x, q)$ decreases during Discrete Transitions:

$$0 < P_q - N_{qr}^T P_r N_{qr} \quad (6)$$

But we don't need $x^T(P_q - N_{qr}^T P_r N_{qr})x > 0$ to be true for any x . Just any $x \in G(q, r)$. If we could find an S with $x^T S x = 0$ for all $x \in G(q, r)$ and possibly > 0 for x outside the guard, a less conservative LMI would be

$$0 < P_q - N_{qr}^T P_r N_{qr} + S$$

We can parameterize a family of matrices to serve as S

$$S = m_{qr} t_{qr}^T + t_{qr} m_{qr}^T$$

Where t_{qr} is a "free" vector. Note that $x^T S x = 0$ for all $x \in G(q, r)$ as required. Thus a less conservative LMI condition than (6) is

$$0 < P_q - N_{qr}^T P_r N_{qr} + m_{qr} t_{qr}^T + t_{qr} m_{qr}^T$$

4 Theorem Statement

Suppose H is non-blocking, non-zeno, has no attractive sliding modes. Also suppose there exists positive definite matrices P_q, \bar{P}_q , symmetric matrices with non-negative entries, U_{qk}, W_q , and vectors t_{qr} such that:

$$\begin{cases} 0 > A_{qk}^T P_q + P_q A_{qk} + L_q^T U_{qk} L_q \\ 0 < P_q - L_q^T W_q L_q \\ q \in Q_0 \ k \in K(q) \end{cases} \quad (7)$$

$$\begin{cases} 0 > \bar{A}_{qk}^T \bar{P}_q + \bar{P}_q \bar{A}_{qk} + \bar{L}_q^T U_{qk} \bar{L}_q \\ 0 < \bar{P}_q - \bar{L}_q^T W_q \bar{L}_q \\ q \in Q_1 \ k \in K(q) \end{cases} \quad (8)$$

$$0 < P_q - N_{qr}^T P_r N_{qr} + m_{qr} t_{qr}^T + t_{qr} m_{qr}^T \quad (9)$$

$(q \in Q_0) \wedge (r \in Q_0)$

$$0 < \bar{P}_q - \bar{N}_{qr}^T \bar{P}_r \bar{N}_{qr} + \bar{m}_{qr} t_{qr}^T + t_{qr} \bar{m}_{qr}^T \quad (10)$$

$(q \in Q_1) \vee (r \in Q_1)$

Where

$$\begin{aligned} \bar{P}_q &= \begin{bmatrix} P_q & 0 \\ 0 & 0 \end{bmatrix} \text{ for } q \in Q_0 \\ \bar{N}_{qr} &= \begin{bmatrix} N_{qr} & 0 \\ 0 & 1 \end{bmatrix} \text{ for } q \in Q_0 \end{aligned}$$

then H is Exponentially Stable.

4.1 Theorem Proof

We develop the proof of the theorem in a series of 4 lemmas. The first 3 lemmas are original work, while Lemma 4 is taken from [1].

Lemma 1 *There exists $\alpha > 0$ and $\beta > 0$ such that*

$$\alpha \|x(t)\|^2 < V(x(t), q(t)) < \beta \|x(t)\|^2 \quad (11)$$

for all t and for all $x \in \mathcal{H}$

Proof: First consider the case where $q \in Q_1$ and $x \in \cup_{q \in Q_1} I(q)$

By construction,

$$\begin{bmatrix} x^T & 1 \end{bmatrix} \bar{L}_q^T W_q \bar{L}_q \begin{bmatrix} x \\ 1 \end{bmatrix} > 0 \quad \forall x \in I(q) \quad (12)$$

Also, as a consequence of (8) we have

$$\bar{P}_q - \bar{L}_q^T W_q \bar{L}_q > Q_q \quad (13)$$

For some positive definite Q_q . Combining (13) and (12),

$$\begin{bmatrix} x^T & 1 \end{bmatrix} \bar{P}_q \begin{bmatrix} x \\ 1 \end{bmatrix} > \begin{bmatrix} x^T & 1 \end{bmatrix} \bar{Q}_q \begin{bmatrix} x \\ 1 \end{bmatrix} \quad \forall x \in I(q) \quad (14)$$

Because Q_q is positive definite, we have $V(x, q) > \alpha_q(\|x\|^2 + 1) \forall x \in I(q)$ for some $\alpha_q > 0$. We can weaken the inequality to conclude, $V(x, q) > \alpha_q\|x\|^2$

Because $x = 0$ is not in the compact region $\cup_{q \in Q_1} I(q)$ there must exist some minimum distance d , such that if $x \in \cup_{q \in Q_1} I(q)$ then $\|x\| > d$.

Also, we can observe that $V(x, q) < \hat{\beta}_q(\|x\|^2 + 1) \forall x \in I(q)$ for some $\hat{\beta}_q > 0$ (Simply pick $\hat{\beta}_q$ to be larger than the largest singular value of \bar{P}_q .) Now define $\beta_q = (1 + 1/d^2)\hat{\beta}_q$. One can verify that $\beta_q\|x\|^2 \geq \hat{\beta}_q(\|x\|^2 + 1) \forall x \in I(q)$. Thus, $V(x, q) < \beta_q\|x\|^2$.

The case where $q \in Q_0$ follows similarly. The final α and β in the lemma statement are found by computing $\min_{q \in Q} \alpha_q$ and $\max_{q \in Q} \beta_q$.

Evaluating the derivative of the Lyapunov function we find,

$$\begin{aligned} \frac{d}{dt} V(t) &= \begin{bmatrix} (x(t)^T A(t) + a^T) & 0 \end{bmatrix} \bar{P}_q \bar{x}(t) \\ &\quad + \bar{x}(t)^T \bar{P}_q \begin{bmatrix} A(t)x(t) + a \\ 0 \end{bmatrix} \\ &= \bar{x}(t)^T (\bar{A}^T(t) \bar{P}_q + \bar{P}_q \bar{A}(t)) \bar{x}(t) \end{aligned} \quad (18)$$

But, as a consequence of condition (8) of the theorem we have

$$(\bar{A}_{qk}^T \bar{P}_q + \bar{P}_q \bar{A}_{qk} + \bar{L}_q^T U_{qk} \bar{L}_q) = -Q_{qk} \quad \forall k \in K(q) \quad (19)$$

For some positive definite Q_{qk} . By construction,

$$\begin{bmatrix} x^T & 1 \end{bmatrix} \bar{L}_q^T U_{qk} \bar{L}_q \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \quad \forall x \in I(q) \quad (20)$$

combining (20) and (19),

$$\bar{x}(t)^T [\bar{A}_{qk}^T(t) \bar{P}_q + \bar{P}_q \bar{A}_{qk}(t)] \bar{x}(t) < -\bar{x}(t)^T Q_{qk} \bar{x}(t) \quad \forall k \in K(q) \forall x \in I(q) \quad (21)$$

Because Q_{qk} is positive definite, there exists $\gamma_{qk} > 0$ such that $\bar{x}(t)^T Q_{qk} \bar{x}(t) > \gamma_{qk}(\|x\|^2 + 1) \forall x \in I(q)$

multiplying (21) by $\lambda_k(t)$ pairwise for each k , summing the result, and combining with (18), we conclude

$$\frac{dV(x(t), q(t))}{dt} < -\gamma_q \|x\|^2 \quad (22)$$

where γ_q is $\min_{k \in K(q)} \gamma_{qk}$.

The analysis for the case where $q \in Q_0$ and $x \in \cup_{q \in Q_0} I(q)$ is very similar. The final γ in the statement of the lemma is found by finding $\min_{q \in Q} \gamma_q$

so that we simply have

$$f(q, x(t)) = \bar{A}(t) \begin{bmatrix} x \\ 1 \end{bmatrix} \quad (16)$$

To simplify notation in the remainder of the proof of this lemma, we define

$$\bar{x}(t) = \begin{bmatrix} x(t) \\ 1 \end{bmatrix} \quad (17)$$

Lemma 3 For any execution of H , every discrete transition satisfies the following

$$(q(\tau_{i+1}), x(\tau_{i+1})) \neq (q(\tau'_i), x(\tau'_i)) \Rightarrow V(x(\tau_{i+1}), q(\tau_{i+1})) < V(x(\tau'_i), q(\tau'_i)) \quad (23)$$

Proof: if $(q(\tau_{i+1}), x(\tau_{i+1})) \neq (q(\tau'_i), x(\tau'_i))$ then there exists an event $(q, r) \in E$ such that $x(\tau'_i) \in G(q, r)$ and $x(\tau_{i+1}) \in R(q, r, x(\tau'_i))$

Now suppose either $q \in Q_1$ or $r \in Q_1$. By our definitions of $G()$ and $R()$ we have

$$\bar{m}_{qr}^T \begin{bmatrix} x(\tau'_i) & 1 \end{bmatrix} = 0 \quad (24)$$

$$\begin{bmatrix} x(\tau_{i+1}) \\ 1 \end{bmatrix} = \bar{N}_{qr} \begin{bmatrix} x(\tau'_i) \\ 1 \end{bmatrix} \quad (25)$$

and thus

$$V(x(\tau'_i), q(\tau'_i)) - V(x(\tau_{i+1}), q(\tau_{i+1})) = \begin{bmatrix} x(\tau'_i)^T & 1 \end{bmatrix} (\bar{P}_q - \bar{N}_{qr}^T \bar{P}_r \bar{N}_{qr}) \begin{bmatrix} x(\tau'_i) \\ 1 \end{bmatrix} \quad (26)$$

Multiplying (10) on the left with $\begin{bmatrix} x(\tau'_i)^T & 1 \end{bmatrix}$ and on the right by $\begin{bmatrix} x(\tau'_i) \\ 1 \end{bmatrix}$ we arrive at

$$V(x(\tau'_i), q(\tau'_i)) - V(x(\tau_{i+1}), q(\tau_{i+1})) > 0 \quad (27)$$

The proof for the case when $q \in Q_0$ and $r \in Q_0$ follows similarly. ■

The following lemma, which is proven in [1], that can also be proven using basic Lyapunov theory given in [3] will be used to complete the proof of our theorem.

Lemma 4 Let $x(t) : [0, \infty) \rightarrow \mathbf{R}^n$ and let $V(t) : [0, \infty) \rightarrow \mathbf{R}$ be a nonincreasing and piecewise C^1 function satisfying

$$\frac{d}{dt} V(t) \leq -\gamma \|x(t)\|^p \quad (28)$$

for some $\gamma > 0$ and some $p > 0$, almost everywhere on $[0, \infty)$

If there exists $\alpha > 0$ and $\beta > 0$

$$\alpha \|x(t)\|^2 < V(x(t), q(t)) < \beta \|x(t)\|^2 \quad (29)$$

then $\|x(t)\|$ tends to zero exponentially.

Proof of Theorem: The proof of the theorem involves checking that the conditions of Lemma 4 hold. First we observe that for all executions, $x(t)$ is indeed defined for $t \in [0, \infty)$ because H is assumed to be non-blocking and non-zeno. Next, we observe that $V(t)$ is a nonincreasing function, for any execution of H , because we have shown that it decreases along both discrete and continuous evolution. Next, we observe that Equation (28) is true for some γ as we verified this with Lemma 2 ($p = 2$ here). Finally, we observe that Equation (29) is true, because this was verified with Lemma 1. Thus, we conclude that under the conditions of our theorem, $\|x(t)\|$ tends to zero exponentially. ■

5 Example

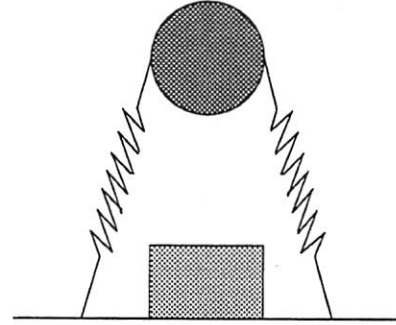


Figure 2: Spring Driven Bouncing Ball

We illustrate the theorem with the example of a spring driven bouncing ball. The system is illustrated in Figure 2. The height of the ball above the block is defined to be x_1 and the ball's velocity is defined to be x_2 . The forces acting on the ball are the spring tension, equal to kx_1 , and the drag from air friction, equal to dx_2 . When the ball hits the block, it rebounds with a velocity equal to $-\alpha x_2$.

We model the spring driven bouncing ball as a hybrid system with two discrete states. The first we call "Fly Down" and assign it the value $q = 1$. The other we call "Fly Up" and assign it the value $q = 2$. A state diagram of the hybrid system is shown in Figure 3. The vector fields in each discrete are $f(x, 1) = A_1 x$ and $f(x, 2) = A_2 x$ where

$$A_1 = A_2 = \begin{bmatrix} 0 & 1 \\ -k & -d \end{bmatrix}$$

The guard for the (1,2) event is the set of states where $x_1 = 0$, and the guard for the (2,1) event is the set of

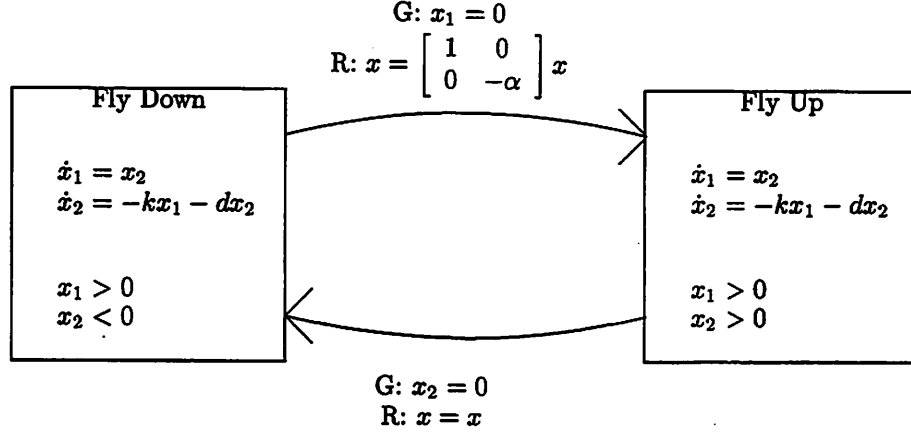


Figure 3: State Diagram of the Spring Driven Bouncing Ball.

states where $x_2 = 0$. Thus, these guards are described by the vectors,

$$m_{12} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } m_{21} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The matrices that describe the reset relations for the (1,2) event and the (2,1) event are

$$N_{12} = \begin{bmatrix} 1 & 0 \\ 0 & -\alpha \end{bmatrix} \text{ and } N_{21} = I$$

Finally, the matrices that describe the invariants of state 1 and 2 are

$$L_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } L_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

We now pick reasonable values for the parameters: $d = 0.1$, $\alpha = 0.8$, $k = 1$. To use the theorem to show this system is exponentially stable, we must search for P_1 , P_2 , W_1 , W_2 , U_1 , U_2 , t_{12} , t_{21} that satisfy the theorem conditions. Through search, we find the following values,

$$P_1 = \begin{bmatrix} 1.005 & 0.05 \\ 0.05 & 1 \end{bmatrix} \text{ and } P_2 = \begin{bmatrix} 1.206 & 0.06 \\ 0.06 & 1.2 \end{bmatrix}$$

$$t_{12} = \begin{bmatrix} 0.5 \\ -0.9 \end{bmatrix} \text{ and } t_{21} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$W_1 = W_2 = U_1 = U_2 = 0$$

satisfy the conditions for the theorem. Thus, the spring driven bouncing ball is exponentially stable.

6 Final Remarks

When I first began my EE291e class project, the first direction I tried was to derive a simultaneous parameterization of Lyapunov function and state feedback matrix, so that one could pose the question of stabilizability using an LMI. This is possible with pure linear systems, and the LMI takes the form

$$QA^T + AQ + BR + R^T B < 0 \quad (30)$$

Where $Q = P^{-1}$, and $R = KP^{-1}$ (P is the Lyapunov matrix, K is the state feedback matrix). However, I was unable to reduce the simultaneous parameterization in the hybrid system case to an LMI. The trick of writing the LMI in terms of P^{-1} doesn't work because Equation (9) cannot be written in terms of P^{-1} and remain an LMI. Also (7) ends up having 2nd order terms between Q and W_q . Indeed, [1] reported the same difficulty.

Another direction I explored was using this framework to write conditions for observability. The approach was to show that the LMI's imply the stability of the error dynamics of some observer, as is done in the analysis of linear systems. Indeed, one can do this for hybrid systems of the form outlined in Section 2, but only if one assumes perfect knowledge of the discrete state. Also, if one tries to simultaneously search for Lyapunov functions and observer feedback matrices, one runs into exactly the same difficulty as discussed earlier – the stability conditions are no longer LMIs.

The extension of the results in [1] and [2] to nontrivial reset relations was the final direction I tried, and led to the work presented here. A useful future extension of this work would be to develop conditions that allow for reset

relations to be any matrix in the convex hull of some set of matrices.

Also, one may also try using relaxed conditions on the Lyapunov function, as developed in [4]. [4] essentially shows that a system is stable if $V(q, x)$ has a lower value each time the system switches into discrete state q , and $V(q, x)$ decreases along continuous evolution in state q . Therefore in the framework of [4], it is not required that $V(q, x)$ decrease during every discrete transition. Devising a scheme to book-keep the value of $V(q, x)$ every time the system enters state q would be the principle challenge.

Even without these future extensions, the work presented here permits Lyapunov analysis of a rich variety of hybrid systems exhibiting both interesting discrete and continuous dynamics.

References

- [1] M. Johansson, *Piecewise Linear Control Systems*, PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1999.
- [2] M. Johansson, and A. Rantzer, "Computation of Piecewise Quadratic Lyapunov Functions for Hybrid Systems." *IEEE Transactions on Automatic Control*, vol. 43, no. 4, 1998.
- [3] S. Sastry, *Nonlinear Systems: Analysis, Stability & Control* Class Notes For EECS 222, UC Berkeley, Berkeley, 1997.
- [4] M. Branicky, "Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, 1998.

ME290S(EE291E) Final Project

Game Theoretic Approach to Control of Drilling

13342743
Motoyoshi OZAKI

Abstract

Drilling process always involves contact of a drill to the surface of a workpiece, and there exists a hybrid control problem, which is similar to the contact stability problem of a manipulator's contacting a surface. This study utilizes the game theoretic approach of hybrid control design to obtain the initial condition and the control law for the drilling of composite materials with allowable thrust force and maximal speed when the drill start drilling.

1. Introduction

Although composite materials provide distinctive advantages in manufacture of advanced products, they are easily damaged unless machining is performed properly. The typical damage is delamination during drilling when the drilling force exceeds a threshold value at critical stages. A quantitative model based on a delamination fracture mechanics approach was suggested by Hocheng and Dharan [1]. Their model relates delamination damage of the laminate to drilling parameters and composite material properties: i.e. the critical thrust force at the onset of crack propagation at the exit is

$$F_o = \pi \sqrt{\frac{8G_{IC}EH^3}{3(1-\nu^2)}} \left(\frac{h}{H}\right)^{\frac{3}{2}} \quad (1)$$

and the critical cutting force at the onset of delamination at the entrance is

$$F_c = k_{slope} \pi \sqrt{\frac{8G_{IC}EH^3}{3(1-\nu^2)}} \left(1 - \frac{h}{H}\right)^{\frac{3}{2}} \quad (2)$$

where H is the thickness of the laminate, h is the uncut depth under the tool, and G_{IC} is the critical crack energy release rate. E is the modulus of elasticity and ν is the Poisson's

ratio. k_{slope} is a constant defined by λ (the helix angle at the drill tip) and μ (the coefficient of friction between tool and work). Eqs. (1) and (2) show that the thrust force must be small at the exit and at the entrance to preclude drilling damage.

The best approach to control the thrust force to be less than the value suggested by (1) and (2) is force controlled drilling. On the other hand, the drill must be simply position- or velocity-controlled before and after the drilling. Hence, there needs the existence of hybrid controller around the transient phase between force-controlled region and position/velocity-controlled region.

Cavusoglu, Yan and Sastry proposed a hybrid system approach to contact stability and force control in robotic manipulator [2]. They applied the game theoretic approach of hybrid control design to synthesize the least restrictive control law for a robotic manipulator to establish and maintain contact with a surface, while keeping interaction forces within specified bounds. If the drilling mechanism of the material is added to the

model of the robotic manipulator, the design method can be similarly applied to the case of hybrid controlled drilling of composite materials.

This paper explores the development of hybrid control strategies, especially at the upper surface, i.e. the transient from position control to force control, in the drilling of composite materials, by applying the game theoretic approach.

2. Game Theory Approach to Hybrid Control Design

This chapter briefly introduces the game theory approach to hybrid control design. This is a summary of 'Chapter 2: Game Theory Approach to Hybrid Control Design' of the paper written by Cavusoglu, et al. [2]. Some textbooks on game theory explain more mathematically about the content of this chapter. ([3], [4])

Consider a dynamical system

$$\dot{x} = f(x, u, d, t) \quad (3)$$

where $x(t)$ is the state, $u(t)$ is the control input and $d(t)$ is the disturbance. The game is played between the control input and the disturbance as the opponent. Disturbances can be the environmental disturbances, control inputs from higher level controller, etc. A cost function, $J(x(0), u, d)$, is defined as the objective of the game, a desired behavior that the controlled dynamical system is desired to satisfy. Suppose u is trying to achieve

$$J \leq C_1 \quad (4)$$

while d is trying to maximize it. The system is said to admit a saddle solution if there exists a $u^*(t)$ and $d^*(t)$ such that

$$J(x(0), u^*, d) \leq J(x(0), u^*, d^*) \leq J(x(0), u, d^*) \quad (5)$$

If the system admits a saddle solution, the analysis gives the optimum control strategy $u^*(t)$ and a set of safe states in which the

control can win the game regardless of the disturbance:

$$S = \{x \in X : J(x, u^*, d^*) \leq C_1\} \quad (6)$$

Given that the initial condition is in the safe set, the least restrictive control law to achieve $J \leq C_1$ is to use u^* when $x(t)$ is at the boundary of S , without any restriction on the control action when $x(t)$ is inside the safe set.

3. Controller Design for Drilling

3-1 Initial Condition

The drilling process is modeled with the force-position characteristic given by

$$F(x) = \begin{cases} 0, & x < 0 \\ C(x)\dot{x}, & x \geq 0 \end{cases} \quad (7)$$

where $C(x)$ ($C_{\max}(x) \geq C(x) \geq C_{\min}(x)$) multiplied by the velocity of the drill is the thrust force of drilling, and is considered as a disturbance. The system is governed by the differential equation

$$m\ddot{x} + C(x)\dot{x} = \tau(t) \quad (8)$$

where $\tau(t) \in [-\tau_m, \tau_m]$ is the control input. The cost functions the thrust force of the drill, and the controller tries to achieve $F(x) \leq F_{\max}(x)$ at any time, where $F_{\max}(x)$ is the maximal thrust force. The initial condition is given by $x(0)=0$ and the initial velocity $\dot{x}(0)$.

The initial energy of the system is:

$$E_0 = \frac{1}{2} m \dot{x}(0)^2 \quad (9)$$

The energy input is until the position x is

$$E_{in}(x) = \int_0^x \tau(x) dx = \tau(\varepsilon)x \quad (10)$$

for some $\varepsilon \in [0, x]$.

The kinetic energy at x when $t=t_x$ is

$$E_x(x) = \frac{1}{2} m \dot{x}(t_x)^2 \quad (11)$$

The energy consumed until the position x is

$$E_{out} = \int_0^x C(x) \dot{x} dx \quad (12)$$

By applying the energy conservation law:

$$E_0 + E_m(x) = E_x(x) + E_{out}(x) \quad (13)$$

to these four values of (9) – (12), we can obtain

$$\frac{1}{2} m \dot{x}(0)^2 + \tau(\varepsilon)x = \frac{1}{2} m \dot{x}(t_x)^2 + \int_0^x C(x) \dot{x} dx \quad (14)$$

Then, the saddle solution is obtained from

$$\tau^*(t) \equiv -\tau_m \quad (15)$$

$$\dot{x}^*(t_x) = \frac{F_{max}(x)}{C_{max}(x)} \quad (16)$$

$$C(x)\dot{x} = F_{max}(x) \quad (17)$$

hence,

$$\frac{1}{2} m \dot{x}(0)^2 = \tau_m x + \frac{1}{2} m \left(\frac{F_{max}(x)}{C_{max}(x)} \right)^2 + \int_0^x F_{max}(x) dx \quad (18)$$

i.e.

$$\dot{x}(0) = \sqrt{\frac{2}{m} \left(\tau_m x + \frac{1}{2} m \left(\frac{F_{max}(x)}{C_{max}(x)} \right)^2 + \int_0^x F_{max}(x) dx \right)} \quad (19)$$

This is the critical speed at the beginning of drilling where the force at x can be the same as $F_{max}(x)$.

3-2 Initial Condition for Special Case

If $F_{max}(x)$ is constant, then the safe set of initial condition with respect to any drill position x is expressed as

$$\dot{x}(0) \leq \sqrt{\frac{2}{m} \left((\tau_m + F_{max})x + \frac{1}{2} m \left(\frac{F_{max}}{C_{max}(x)} \right)^2 \right)} = g(x) \quad (20)$$

As a typical example for the drilling of composite materials, the following two conditions can be added (the first is above mentioned):

$$F_{max}(x) = F_{max} \text{ (constant)} \quad (21)$$

$$C_{max}(x) = \alpha + \beta x \quad (22) \quad \text{not}$$

where (21) comes from the fact that F_{max} does change inside the first layer of the surface of the material, and (22) comes from general empirical equation. Then

$$\dot{x}(0) \leq \sqrt{\frac{2}{m} \left((\tau_m + F_{max})x + \frac{1}{2} m \left(\frac{F_{max}}{(\alpha + \beta x)} \right)^2 \right)} = g(x) \quad (20)$$

is the safe set of initial conditions with respect to x .

If $\alpha \geq \sqrt[3]{\frac{m F_{max}^2 \beta}{\tau_m + F_{max}}}$ is satisfied, then

$$x_{c1} = 0 \quad (21)$$

(when the drill touched the surface of the material) is the critical position, and $g(0)$ is the smallest value. On the other hand, if

$\alpha < \sqrt[3]{\frac{m F_{max}^2 \beta}{\tau_m + F_{max}}}$ then

$$x_{c2} = \frac{1}{\beta} \left(\sqrt[3]{\frac{m F_{max}^2 \beta}{\tau_m + F_{max}}} - \alpha \right) \quad (22)$$

is the critical position, and $g(x_{c2})$ decides the maximum approaching speed of the drill.

3-3 Controller Design during Drilling

By modifying the content of 3-1, the relation between two arbitrary positions x_1 and x_2 ($x_1 < x_2$) can be expressed as

$$\begin{aligned} & \frac{1}{2} m \dot{x}(t_{x1})^2 + \tau(\varepsilon)(x_2 - x_1) \\ &= \frac{1}{2} m \dot{x}(t_{x2})^2 + \int_{x1}^{x2} C(x) \dot{x} dx \end{aligned} \quad (23)$$

hence,

$$\dot{x}(t_{x1}) \leq \sqrt{\frac{2}{m} \left(\tau_m x_2 + \frac{1}{2} m \left(\frac{F_{max}(x_2)}{C_{max}(x_2)} \right)^2 + \int_{x1}^{x2} F_{max}(x) dx \right)} = g(x_2) \quad (24)$$

for any position $x_2 (> x_1)$ is the safe set for the velocity at x_1 . If the velocity is strictly less

than this value for any x_2 , then the controller can choose $\tau(t) = \tau_m$ to make the drilling speed faster until equality of (24) can be satisfied for at least one of x_2 , but should never exceeds $g(x)$ for any value for x_2 .

4. Conclusions and Discussion

Game theory was applied to calculate the drill approaching speed that will not cause problems such as delamination of composite materials. The optimal controller design during drilling in the first layer of the composite materials was also examined. The same approach can be used for the exit of the drill at the lower surface of the workpiece. Drilling of metals without burr formation can be also the objective of that approach.

Reference

- [1] H. Hocheng and C. K. H. Dharan, "Delamination during Drilling in Composite Laminates," *ASME: Journal of Engineering for Industry*, Vol. 112, pp. 236-239, 1990.
- [2] M. C. Cabusoglu, J. Yan, and S. S. Sastry, "A Hybrid System Approach to Contact Stability and Force Control in Robotic Manipulators", to be published.
- [3] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, Academic Press, first edition, 1982.
- [4] W. Jianhua, *The Theory of Games*, Clarendon Press, 1988.

Reachability Calculation for hybrid systems

Arnab Nilim

1 Introduction

Reachability calculation is very important for the verification of hybrid systems. If we can solve the reachability problem, we can answer a lot of important questions about safety properties. But so far, the reachability calculation for the hybrid system is not a completely solved problem. In the continuous system, reachability calculation is performed by the procedure based on the ellipsoidal approximation. The calculation of the tight external ellipsoidal approximations of reach sets and reach tubes for the linear time-invariant control is already performed [1]. But the extension of this model for the hybrid system is yet to be done. Our work is to extend this model for the hybrid system. We also will use Linear Matrix Inequalities technique to take care the discrete transition [4]. The class of system we are dealing with is the open hybrid automation.

2 Problem Statement

The system is an H is an open hybrid automation. There are various discrete states in this system which are called d_1, d_2, \dots and the differential equation governing each discrete state d_i is,

$$\dot{x} = A_i(t)x + B_i(t)u \quad (1)$$

where $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ is the control. $A(t)$ and $B(t)$ are both continuous and the system is completely controllable. The $Init \subseteq \mathbb{R}^n$ and let it is approximated by an outer ellipsoid $E(x^0, X^0)$. The quadratic form,

$$W(\tau, \sigma) = \int_{\sigma}^{\tau} (l, X(\tau, s)B_i(s)B_i'(s)X'(\tau, s)l) ds \quad (2)$$

where $l \in \mathbb{R}^n$ and $W(\tau, \sigma)$ is positive definite for any $\tau > \sigma$. Here, $X(t, s)$ is the transition matrix for the homogeneous system (1),

$$\delta X(t, s)/\delta t = A_i(t)X(t, s) \quad (3)$$

where $X(s, s) = I$ where I is the identity matrix. For a linear time invariant system $A_i(t) = A_i = \text{const}$, $X(t, s) = \exp(A_i(t - s))$.

The control $u = u(t)$ is any measurable function restricted by the hard bounds $u \in P(t)$ for almost all t . The set valued bound $P_i(t)$ is a non degenerate ellipsoid in t , $P_i(t) = E(q_i(t), Q_i(t))$, where

$$E(q_i(t), Q_i(t)) = \{u : (u - q_i(t), Q_i^{-1}(t)(u - q_i(t)) \leq 1\} \quad (4)$$

with $q_i \in \mathbb{R}^m$ (the center of the ellipsoid) and positive definite matrix $Q_i(t) \in \mathbb{R}^{m \times m}$ (the matrix of the ellipsoid) continuous in t .

We have to find, given position $\{t_0, x^0\}$, the reach set $\chi(\tau, t_0, x^0)$ at time $\tau > t_0$ from this position is the set, $\chi[\tau] = \chi(\tau, t_0, x^0) = \{x[\tau] = x(\tau, t_0, x^0) \text{ reachable at time } \tau \text{ by system 1, with } x(t_0) = x^0 \text{ through all possible controls } u \text{ that satisfy the constraint.}$

The set valued function $\tau \mapsto \chi[\tau] = \chi(\tau, t_0, x^0)$ is the reach tube.

The reach set $\chi(\tau, t_0, x^0)$ (at time τ from set $\chi^0 = \chi(t_0)$) is the union $\chi(\tau, t_0, X^0) = \bigcup \{\chi(\tau, t_0, x^0) \mid x^0 \in \chi^0\}$. The set valued function $\tau \mapsto \chi[\tau] = \chi(\tau, t_0, x^0)$ is the reach tube from set χ^0 .

Here, we are first calculating the reach set and reach tube for the continuous state. As we know the each continuous region belonged by its corresponding discrete state, we can easily calculate the reach set for the discrete state if we have the reach tube for the continuous state. We know the discrete transition takes place when there is a change in vector field. The boundary between the different vector fields is also known, which distinguishes one discrete state from another.

In this work we are using ellipsoidal technique [1] to calculate the reach set within a discrete state. We then get an optimal ellipsoid by Linear Matrix inequality (LMI) such that it contains the reach set and it is of the minimum volume. We further calculate the portion of the boundary between two discrete states that is intersected by the optimal ellipsoid over all time, which can be further approximated optimally by an ellipsoid (using LMI technique). This ellipsoid which is in the boundary can be restated as the initial condition for the next state. And if we know the ellipsoid that contains initial condition within a discrete state (with bounded control), we can calculate the reach set by the previous procedure described in [1]. In this way, we will be able to take care the discrete transition in a hybrid system and carry out our calculation for a hybrid system.

2.1 Ellipsoidal approximation of reach set within a discrete state

The initial set $E(x^0, X^0)$ and the control set $E(q_i(t), Q_i(t))$ are ellipsoids. But the reach set $\chi[t] = \chi(t, t_0, E(x^0, X^0))$ will not generally be an ellipsoid. As indicated in [2], the reachability set $\chi[t]$ may be approximated both externally and internally by ellipsoids E_- and E_+ , with $E_- \subseteq E \subseteq E_+$ implies $E = E_+$.

Theorem 2.1 The reach set $\chi[t] = \chi(t, t_0, E(x^0, X^0))$ is a convex compact set in \mathbb{R}^n , continuous in t . ([2]).

Definition 2.1 The support function of an ellipsoid is [3],

$$\rho(l \mid E(q_i, Q_i(t))) = \max\{(l, x) \mid x \in E(q_i(t), Q_i(t))\} \quad (5)$$

$$= (l, q_i(t)) + (l, (Q_i(t)l)^{1/2}) \quad (6)$$

And the continuity in $Q(t)$ means that its support function $\rho(l \mid Q(t))$ is continuous in t uniformly in l with $(l, l) \leq 1$.

Now the problem is, given a unit vector $l(t), (l, l) = 1$, continuously differentiable in t , we will have to find an external ellipsoid $E_+^l[t] \supseteq \chi[t]$ that would ensure $\forall t \geq t_0$, the equality $\rho(l(t) | \chi[t]) = \rho(l(t) | E_+^l[t]) = (l(t), x^l)$ so that the supporting hyper plane for $\chi[t]$ generated by $l(t)$, namely the plane $(x - x^l(t), l(t)) = 0$ that touches $\chi[t]$ at point $x^l(t)$ would also be supporting hyper plane for $E_+^l[t]$ and touch it at the same point.

Theorem 2.2 The class $\mathbf{E} = \{E_+\}$ consists of ellipsoids that approximate the reach set $\chi[t]$ externally and are of the form $E_+ = E(x^l, X_+^l[t])$, where vector l is given and x^l satisfies the equation,

$$\dot{x}^l = A_i(t)x^l + q_i(t) \quad (7)$$

where $x^l(0) = x^0, t \geq t_0$ and With the controllability assumption we will further assume, without loss of generality, that $B(t) = I$. (To obtain the case $B(t) \neq I$, we will have to substitute $Q(t)$ by $B(t)Q(t)B'(t)$). And if we assume that the function $l(t)$ is of the following form $l(t) = X(t_0, t)l$, for the time invariant case $l(t) = \exp(-A'(t - t_0)l)$. Then $X_+^l[t]$ can be determined by the following differential equation,

$$\dot{X}_+^l = A_i(t)X_+^l + X_+^l A_i'(t) + \pi^l(t)X_+^l + (\pi^l(t))^{-1}Q_i(t) \quad (8)$$

where $X^l(t_0) = X^0$ and

$$\pi^l(t) = (l, X(t_0, t)Q_i'(t)X'(t_0, t)l)^{1/2}(l, X_+[t]l)^{1/2} \quad (9)$$

which can be further simplified as following, as done in

$$\pi^l(t) = \frac{(l(t), Q_i(t)l(t))^{1/2}}{(l(t), X_+[t]l(t))^{1/2}} \quad (10)$$

[1]

Theorem 2.3 Under the above assumption the solution to the the problem is given by $E_+^l[t] = E(x^l(t), X_+^l[t])$ where x^l satisfies the equation 7 and $X_+^l[t]$ is a solution to the equation 8 and 10. ($X_+^l[t]$ depends on the vector l) [1]

Theorem 2.4 For any $t \geq t_0$ the reach set $\chi[t]$ may be described as

$$\chi[t] = \cap \{E(x^l, X_+^l[t]) \mid l : (l, l) = 1\} \quad (11)$$

With the equations given above we can calculate the reachability of a system in the i th discrete state. Right now, we will try to extend it if we want to calculate the reachability in the $(i + 1)$ the state.

2.2 Extension of calculation incorporating the $(i + 1)$ th discrete state

As we have seen that, the reach set is the intersection of all the ellipsoids with different vector l . It is proved [1] that if we take infinite ellipsoids like this, the intersection of all ellipsoids will converge to the real reach set. It will be cleared with the figure 1.

Right now, the problem lies in determining the part of the boundary of the two discrete states which is reachable from state i . And that will be the initial condition for the $(i + 1)$ th state. Let us assume that assume that,

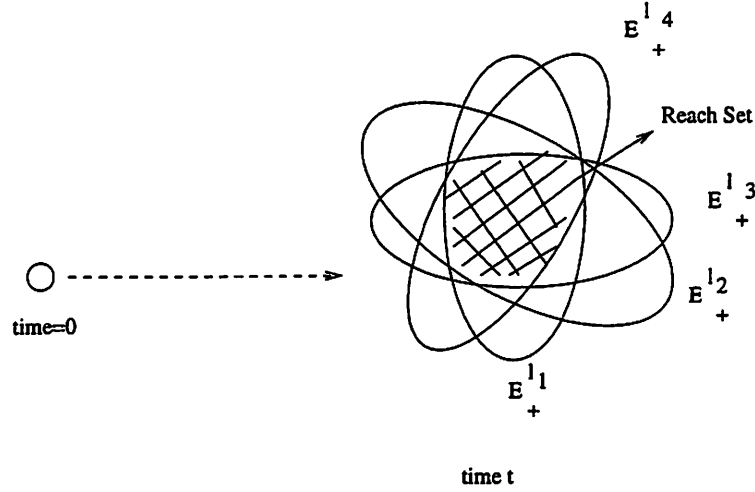


Figure 1: Reach set approximated by the intersection of ellipsoids

1. The boundary of the two discrete state q_i and q_{i+1} is a straight line/plane/hyper plane.
2. Reach set does not intersect with more than two discrete states. Or reach tube just intersects with one boundary.
3. In all the discrete states, vector fields are linear time invariant.

Intersection of the ellipsoids is not an ellipsoid. Though it is a convex set. it does not have a well defined shape which makes it very hard to determine its intersection with the boundary. So we will approximate it with an ellipsoid that contains it with a minimum volume.

2.3 Ellipsoidal Approximation of the intersection of ellipsoids

The problem of finding the smallest volume that contains a polytope described by a set of linear inequalities, i.e., $\{x \mid a_i^T x \leq b_i, i = 1, \dots, p\}$ is solved by using Linear Matrix Inequalities as in [4]. This problem is np hard. And this problem is equivalent to the general concave quadratic programming problem.

If we consider subsets formed by the intersection of $E_+^{l1}, E_+^{l2}, E_+^{l3}, \dots, E_+^{lp}$. We approximate the set by an ellipsoid E_0 . We can describe the ellipsoid E_+^{li} in the following way.

$E_+^{li} = \{x \mid T_i(x) \leq 0\}$, where $T_i(x) = x^T P_i x + 2x^T q_i + r_i$. And $P_i = P_i^T > 0$ and $q_i^T P_i^{-1} - r_i > 0$ (which ensures that E_+^{li} is nonempty and does not reduce to a single point).

We want to calculate a small ellipsoid E_0 that covers the union of the ellipsoids $E_+^{l1}, E_+^{l2}, E_+^{l3}, \dots, E_+^{lp}$, the convex hull of the intersection. We will describe the ellipsoids by the associated quadratic functions $T_i(x) = x^T P_i x + 2x^T q_i + r_i$. We have

$$E_0 \supseteq \cap_{i=1}^p E_+^{li} \quad (12)$$

Which holds and given that $E_+^{l1}, E_+^{l2}, E_+^{l3}, \dots, E_+^{lp}$ and E_0 are all np complete. So finding E_0 is a Linear Matrix Inequality problem.

There are two ways in which we can calculate the optimal ellipsoids for this problem solving convex problems. The first procedure uses S -procedure to derive an LMI that is sufficient for the above condition to hold.

As we know, $E_0 = \{x \mid T_0(x) \leq 0\}$ where $T_0(x) = x^T P_0 x + 2x^T q_0 + r_0$. Since our representation of E_0 is homogeneous, we will now normalize P_0, q_0, r_0 in a convenient way such that $q_0^T P_0^{-1} q_0 - r_0 = 1$. In other words we set,

$$r_0 = q_0^T P_0^{-1} q_0 - 1 \quad (13)$$

From the S -procedure there exists positive scalars $\tau_1, \tau_2, \tau_3, \dots, \tau_p$ such that

$$\begin{pmatrix} P_0 & q_0 \\ q_0^T & q_0^T P_0^{-1} q_0 - 1 \end{pmatrix} - \sum_{i=1}^p \tau_i \begin{pmatrix} P_{l_i} & q_{l_i} \\ q_{l_i}^T & r_{l_i} \end{pmatrix} \leq 0 \quad (14)$$

which can be rewritten as the LMI (in variables P_0, q_0 and r_0 and $\tau_1, \tau_2, \tau_3, \dots, \tau_p$)

$$\begin{pmatrix} P_0 & q_0 & 0 \\ q_0^T & -1 & q_0^T \\ 0 & q_0 & -P_0 \end{pmatrix} - \sum_{i=1}^p \tau_i \begin{pmatrix} P_{l_i} & q_{l_i} & 0 \\ q_{l_i}^T & r_{l_i} & 0 \\ 0 & 0 & 0 \end{pmatrix} \leq 0 \quad (15)$$

The LMI is sufficient but not necessary for 23 to hold. We can find the best such approximation (i.e. the smallest volume) by solving the convex problem.

Minimize $\log \det P_0^{-1}$ subject to, $P_0 > 0, \tau_1 \geq 0, \tau_2 \geq 0, \dots, \tau_p \geq 0$. (which is solved in any LMI tool box directly). SO solving this optimization problem, we will get the optimal ellipsoid E_0 .

2.4 Using LMI Ellipse for the Reachability calculation of the Hybrid system

In the previous section, we have calculated E_0 which comprises the intersection of all the ellipsoids $E_+^{l_1}, E_+^{l_2}, E_+^{l_3}, \dots, E_+^{l_p}$ with the least volume. So $E_0(t)$ also consists of the reach set $\chi[t]$ for the i th discrete state. Right now we have to extend the reach set for the $(i+1)$ th discrete state. The situation can be well visualized in the following figure.

As in the figure, the boundary between the i th state and the $(i+1)$ th the state is defined by M . As long as the ellipsoid $E_0(t)$ is in the left side of the boundary M , it is in the state i . The transition can take place as soon as $E_0(t)$ hits the boundary M . As $E_0(t)$ can be calculated at every time instant, we can calculate the time t_1 at which E_0 touches the boundary M for the first time. So any time, $t > t_1$ the trajectory can be in the either state. Right now, we need to calculate the portion of M that can be reached by the system. The vector field is different in the two sides of the boundary. So whenever the ellipse E_0 crosses the boundary, different parts of it are in different vector fields. But for the sake of calculation, let's calculate the time t_3 (it is the time when the ellipse E_0 has just completely crossed the boundary M) using the vector field of the i th state. So at any time t such that $t_1 < t < t_3$, the ellipse $E_0(t)$ is cut by the boundary. And at any time $t > t_3$, the $E_0(t)$ is completely in the $(i+1)$ th state. In each time instant $t_1 < t < t_3$, the boundary M intersects with the ellipsoid .

Theorem 2.5 If an n -dimensional ellipsoid intersects with a plane, the intersection is a $(n-1)$ th dimensional ellipsoid (unless it is tangential where it will be an ellipsoid of dimension $n-2$).

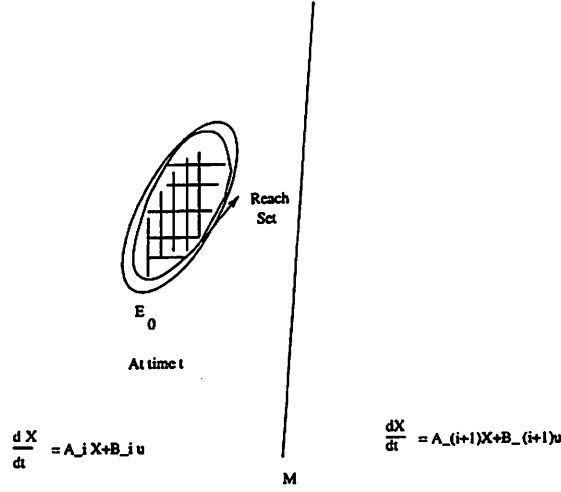


Figure 2: LMI Approximation of the reach set

Proof. Without the loss of generality, let the ellipsoid is $x^T M x = 1$ ($x \in \mathbb{R}^n, M \in \mathbb{R}^{n \times n}$) where,

$$M = \begin{pmatrix} \frac{1}{a_1^2} & 0 & \dots & \dots \\ 0 & \frac{1}{a_2^2} & 0 & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \frac{1}{a_n^2} \end{pmatrix}$$

The plane $H^T x = 0$ where $H \in \mathbb{R}^{n \times n}$. As $x \in \text{null}(H^T)$ we can write x as, $x = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_{n-1} v_{n-1}$ where v_1, v_2, \dots, v_{n-1} are the eigenvectors of H . So we can write x as,

$$x = \begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_{n-1} & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_{n-1} \\ 0 \end{pmatrix} \quad \text{Plugging this value of } x \text{ in } x^T M x = 1 \text{ shows that the}$$

intersection is an $(n - 1)$ th dimensional ellipsoid. The result can be easily further generalized even if the center of the ellipsoid is not the origin and the plane not containing the origin.

Again intuitively it can be understood very easily. Any ellipsoid $x^T A x = 1$ can be transformed into a unit sphere with a linear transformation (As $x^T A x = 1$ is an ellipsoid, A is positive definite and can be represented as $A = P^T P$. Let, $Px = y$ and this linear transformation will yield $y^T (P^{-1})^T P^T P P^{-1} y = 1$ or $y^T y = 1$ which is a sphere). As plane remains as a plane in a linear transformation, our original system transforms into a sphere intersecting with a plane, As we know that if an n dimensional sphere intersects with a plane, it will be a sphere with $n-1$ dimensions. And with the linear transformation ($P^{-1}y = x$) the intersection will be an ellipsoid (sphere is a special type of ellipsoid) with one less dimension. But if it intersects tangentially, then the dimension will be $n-2$.

From the above theorem, it is clear that at any time $t_1 < t < t_3$, the ellipsoid E_0 intersects the boundary and the intersection is a reduced dimensional ellipsoid.

So if our sample calculation time interval is m , there will be approximately $(t_3 - t_1)/m + 1 = k$ no of $(n - 1)$ dimensional ellipsoids generated by the intersection of E_0 and the boundary M .

Let's define \bar{M} such that,

$$\bar{M} = \cup_{t \in (t_1, t_3)} (E_0(t) \cap M) \quad (16)$$

\bar{M} is not an ellipsoid as it is the union of the ellipsoids. As we will see that we can calculate the LMI ellipsoid that will contain \bar{M} with having the minimum volume. The concept can be well visualized in the figure 2,

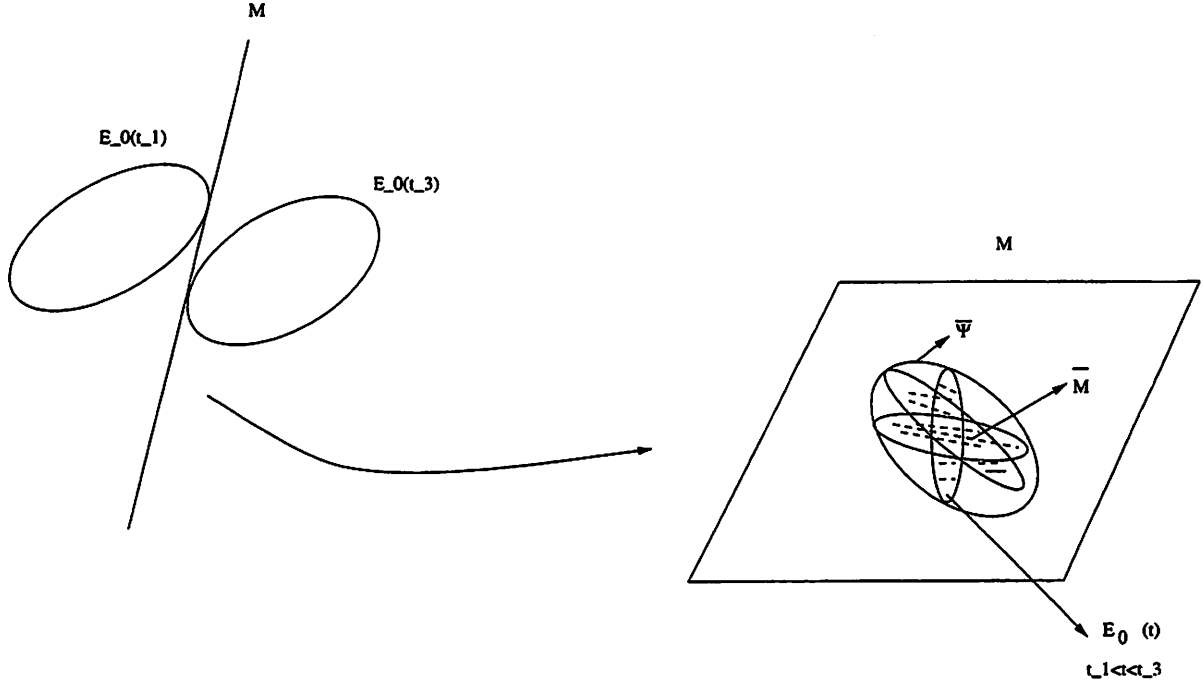


Figure 3: LMI Approximation of the reach set

2.5 Outer Approximation of the union of ellipsoid

We seek a small ellipsoid $\bar{\psi}$ that covers the union of ellipsoid $(E_0(t_1), (E_0(t_2), \dots, (E_0(t_k))$ that can also be calculated by the LMI technique. We can describe these ellipsoids with the associated quadratic functions $T_i(x) = x^T E_i x + 2x^T f_i + g_i$ ($\bar{\psi} = \{x \mid T(x) \leq 0\}$). We have to get,

$$\bar{\psi} \supseteq \cup_{i=1}^k E_0(t_i) \quad (17)$$

where $\bar{T}(x) = x^T \bar{E}x + 2x^T \bar{f} + \bar{g}$.

There exists nonnegative scalars τ_1, \dots, τ_k such that for every x ,

$$\bar{T}(x) - \tau_i T_i \leq 0, i = 1, 2, \dots, k.$$

We can normalize \bar{E}, \bar{f} and \bar{g} in a convenient way such that $\bar{g} = \bar{f}^T (\bar{E})^{-1} \bar{f} - 1$.

Using the scour complements,

$$\begin{pmatrix} \bar{E} & \bar{f} & 0 \\ \bar{f}^T & -1 & \bar{f}^T \\ 0 & \bar{f} - \bar{E} & \bar{f}^T \end{pmatrix} - \tau_i \begin{pmatrix} E_i & f_i & 0 \\ f_i & g_i & 0 \\ 0 & 0 & 0 \end{pmatrix} \leq 0 \quad (18)$$

where $i = 1, 2, \dots, k$.

With the normalization, the volume of $\bar{\psi}$ is proportional to the $\sqrt{\det \bar{M}^{-1}}$. Thus we can find the smallest volume ellipsoid containing the union of ellipsoid $(E_0(t_1), (E_0(t_2), \dots, (E_0(t_k))$ by solving the convex problem

Minimize $\log \det \bar{M}^{-1}$ subject to $\bar{M} > 0$, $\tau_1 \geq 0$, $\tau_2 \geq 0, \dots, \tau_k \geq 0$. We can get the ellipsoid $\bar{\psi}$. This is called Lowner-John ellipsoid.

So we can calculate an LMI approximation of the union of ellipsoids $\bar{\psi}$ such that,

$$\bar{\psi} \supseteq \bar{M} \quad (19)$$

where, $\bar{M} = \cup_{i=1}^k (E_0(t_i) \cap M)$. And $\bar{\psi}$ is the minimum volume that contains it.

So $\bar{\psi}$ is within the boundary and it contains the reach of the i th state. And as $\bar{\psi}$ is a convex set, we can further continue our calculation to get the reach set within the state $(i + 1)$ where the initial condition lies within $\bar{\psi}$. So we will use the following equations,

$$\dot{x}^l = A_{i+1}(t)x^l + q_{i+1}(t) \quad (20)$$

$$\dot{X}_+^l = A_{i+1}(t)X_+^l + X_+^l A_{i+1}'(t) + \pi^l(t)X_+^l + (\pi^l(t))^{-1}Q_{i+1}(t) \quad (21)$$

where $X^l(t_0) = X^0$ and

$$\pi^l(t) = (l, X(t_0, t)Q_{i+1}'(t)X'(t_0, t)l)^{1/2} \{l, X_+[t]l\}^{1/2} \quad (22)$$

The center of the $\bar{\psi}$ is the initial condition for the equation 28 and $\bar{\psi}$ is the initial condition for equation 29. So we can calculate the reach set of $(i + 1)$ th state by solving above equations for various l . But there will be two problems. Firstly, we will have to reset the time after every discrete transition. We will have to fix one time from which we will carry out our calculation for $(i + 1)$ the state. But it will not cause any problem in determining the reach tube. But in the future work we will investigate the optimal way of handling this issue. Secondly, we have a $(n - 1)$ dimensional ellipsoid as an initial set with the system of dimension n . So there will be a problem of singularity. So we will derive an equation of ellipsoid with the n dimensions which will contain this $(n - 1)$ dimensional ellipsoid and have the minimum volume.

2.6 Converting an $(n-1)$ dimensional ellipsoid to an n dimensional ellipsoid with minimum volume

Let the equation of the $(n - 1)$ dimensional ellipsoid which is the initial condition for $(i + 1)$ th state is $x^T P x = 1$. But the matrix P will be singular in this case. After diagonalizing the matrix P , let us assume that we will get only one eigenvalue of P as 0. Let the eigenvalues of the matrix is, $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_{n-1}$ and 0 (The eigen vector corresponds to the 0 eigenvalue value is

the direction which is perpendicular to the plane). And let the corresponding eigenvectors are $v_1, v_2, v_3, \dots, v_{n-1}, v_n$. v_n is the eigenvector corresponding to the 0 eigenvalue. If we diagonalize the matrix, we will get,

$$\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} \begin{pmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \alpha_{n-1} \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}^{-1} = P.$$

So to make P nonsingular, we will put a very small value ϵ (which is close to zero) replacing the eigenvalue 0. In this way, we will get an nonsingular matrix P' , such that,

$$\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} \begin{pmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \alpha_{n-1} \\ 0 & 0 & 0 & \epsilon \end{pmatrix} \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}^{-1} = P'.$$

In this way, P' will contain P and it's volume will be lesser as ϵ is really small.

3 Algorithm

Our algorithm can be written as follows,

1. (Step 1). $Init \subseteq \mathbb{R}^n$. Choose a minimum volume ellipsoid that will contain the *init*.
2. (Step 2). Choose a vector $l(t) \in \mathbb{R}^n$. So the problem is, given a unit-vector function $l(t)$, $(l, l) = 1$, continuously differentiable in t , we will have to find an external tight ellipsoid $E_+^l[t] \supseteq \chi[t]$ (where $\chi[t]$ is the reach set) that would ensure for all $t \geq t_0$, the equality,

$$\rho(l(t) | \chi[t]) = \rho(l(t) | E_+^l[t]) = (l(t), x(t))$$

3. (Step 3). Choose different $l_1(t), \dots, l_p(t)$ and compute $E_+^l(t)$ for all those different $l(t)$ s. $Reach(Init) \subseteq \cap_{i=1}^p (E_{l_i})$
4. (Step 4). Using the Linear Matrix Inequalities (LMI) technique, we will calculate the ellipsoid $E_0(t)$ that will contain the $\cap_{i=1}^p (E_{l_i})$ and have the minimum volume. So at the each sampling time we have p different ellipsoids associated with the different vectors and the optimal ellipsoid $E_0(t)$.
5. (Step 5) Carry out the calculation of E_0 till it hits the boundary of the two states M . The first time it intersects M is t_1 . We will also get a time t_3 such that the ellipsoid $E_0(t)$ has just crossed the boundary using the vector field of *ith* state. In each time $t_1 \leq t \leq t_3$ the ellipsoid $E_0(t)$ is intersected by the boundary. As we know that if an n dimensional ellipsoid intersects with a hyper plane, the intersection will be a $(n-1)$ dimensional ellipsoid. If the sample time interval is m , we will get $(t_3 - t_1)/m + 1 = k$ number of $(n-1)$ dimensional ellipsoids.

6. (Step 6) Using the LMI technique, we will get an $(n - 1)$ dimensional ellipsoid $\bar{\psi}$ that will contain the union of all the $(n - 1)$ dimensional ellipsoids. Or $\bar{\psi} \supseteq \bar{M} = \cup_{t \in (t_1, t_3)} (E_0(t) \cap M)$.
7. (Step 7) $\bar{\psi}$ is the $(n - 1)$ dimensional ellipsoid . We will calculate an ellipsoid an n dimensional ψ that contains $\bar{\psi}$ and will be of the minimum volume.
8. (Step 8) We have the initial ellipsoidal set for the $(i + 1)$ th state. So we can carry out the calculation from 1 to 3 and will get reach set in $(i + 1)$ th state. ($Reach(Init \subseteq \psi)$)
9. (Step 9) We will carry out our calculation as long as we keep getting new reach sets.

An alternative algorithm will be as follows,

1. Step 1-3 is same as previous.
2. Instead of calculating the LMI optimal ellipsoid, we will use the p different ellipsoids $(E_+^{l_1}, E_+^{l_2}, \dots, E_+^{l_p})$. Each $E_+^{l_i}$ generates W_i number of $(n - 1)$ dimensional ellipsoids in boundary when it is intersected with the boundary. If t_1^i is the first time that $E_+^{l_i}$ intersects with the boundary and t_3^i is the first time that $E_+^{l_i}$ intersects with the boundary, then $W_i = \frac{t_3^i - t_1^i}{m} + 1$
3. Let S_i^j be each of these ellipsoids that are obtained by the intersection of $E_+^{l_i}$ with the boundary. So, $Reach(Init) \subseteq \cap_{i=1}^p (\cup_{j=1}^{W_i} S_i^j)$. We can approximate these $(n - 1)$ dimensional ellipsoid by the LMI matrix such that it will contain the ellipsoid with the minimum volume (first LMI for the unions and then LMI for the intersection) . Let the Ellipsoid is ψ .
4. We will follow step 6-9 of the previous algorithm and we can obtain the reach set.

4 Simulation

The system,

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u \quad (23)$$

where,

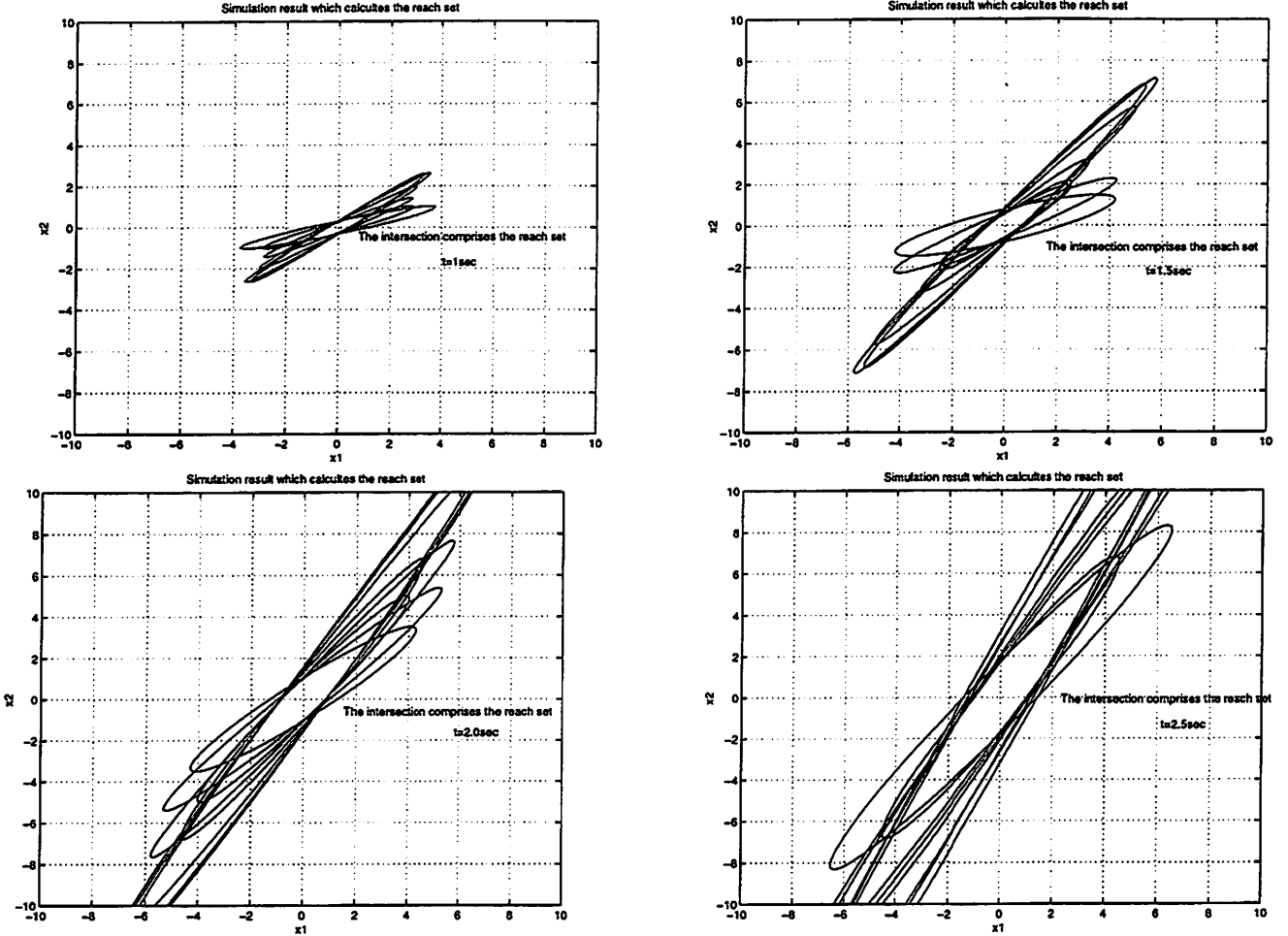
$$\begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (24)$$

and control $\|u\| \leq 1$. And A and B are completely controllable.

We chose 10 different vectors $l \in \mathbb{R}^2$. And that gives the parametric family of curves that cover the reach tube $\chi[.]$. At any time instant, $\cap_{i=1}^{10} E_{l_i} \supseteq \chi[.]$. If we could do the simulation with the infinite no of ellipsoid, we could get the exact reach set.

The simulation result is given below. The reach set is getting expanded with the time increase.(the time chosen are 1 sec, 1.5 sec, 2.0 sec and 2.5 sec).

Due to the unavailability of the LMI tool box (MATLAB) in our lab, we could not implement our model completely. But we are going to get the LMI soon and will implement our model fully.



5 Conclusion and Future Research scope

We have an algorithm to calculate the reach set for the hybrid system of the class open hybrid automation. In this work, we have completely implemented the reach set calculation where there is no discrete transition. The further work should take into account,

1. We will implement our model having the discrete transition. In order to do so, we will use LMI tool box of MATLAB .
2. In the algorithm, it is assumed that the boundary between two consecutive discrete states is a plane. But the boundary can have curvature. In that case we might have to use manifold theory to solve this problem.

3. The intersections were only between two discrete states. But it may happen that the boundaries of the three or more states boundary intersect with the reach set. In that case, the calculation will be further complex.
4. A calculation should be carried out for the non-linear or time-varying systems.
5. If the center of the ellipsoid dynamics is spiral, the reach tube calculated in our way will give a much bigger over-approximation. We will look into this issue in our future work.

References

- [1] A.B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for the reachability calculation. (preprint), 1998.
- [2] Valyii Kurzhanski A.B. *Ellipsoidal Calculus for estimation and Control*. Birkhauser, Boston, USA, 1996.
- [3] R.T Rockafellar. *Convex Analysis*. Princeton University Press, New Jersey, USA, 1970.
- [4] Eric Feron Stephen Boyd, Laurent El Ghaoui. *Linear Matrix Inequalities in system and Control Theory*. Siam Studies in Applied Mathematics, Philadelphia, 1994.

Stability issues for the Fast Positive Force Transient Control

EE291E Spring 1999, Project Report
Prof. S. Sastry & J. Lygeros
C. Pinello

Abstract

The engine control problem can be decomposed into a set of sub-problems corresponding to *regions of operation* identified by the settings of the control devices available to the driver (for example accelerator pedal angle, selected gear). One of these regions is the so called Fast Positive Force Transient, where a quick acceleration is requested, maintaining certain comfort standards. In [1] this control problem is formulated as a hybrid control problem and solved by approximation of an auxiliary continuous control problem. In this report we investigate the stability (convergence) of the closed loop system.

1 Proposed Solution

In [1] we formulated the fast positive force transient problem as an hybrid minimum time control problem. We solved the problem relaxing it to a continuous time control problem, abstracting away all the dynamics related to the torque generation. So we assumed we had an ideal torque generator, then we inverted the dynamics of the system so to have the jerk as input and the torque as output. In this way the control problem has only got constraints on the input (jerk $j \in [0, j_{\max}]$) and the solution can be found relatively easily by means of Pontryagin's Maximum Principle. The solution is well known to be a bang-bang control and in the linear case it corresponds to a state feedback which switches input values across a given surface.

2 Stability Analysis

In this section we want to investigate what happens close the desired manifold (the cylinder \mathcal{C}_c). The discretization of the times at which the control actions take place, may cause the ideal switching to be delayed so much that we may actually be past the cylinder. In this case continuing to apply zero jerk does not steer the state to \mathcal{C}_c . This would lead to a violation of the constraints,¹ since we need a negative jerk to get back to the cylinder. The situation is illustrated in figure (1).

So at low engine speed and for some initial condi-

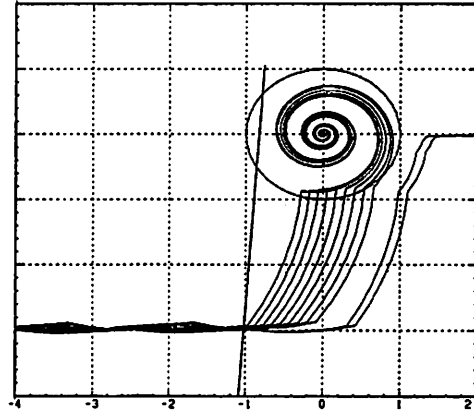


Figure 1: Late switching

tions, trajectories have one admissible switching point right before the the switching surface (i.e. too early to switch) and the next one so far from it that the cylinder cannot be reached.

The proposed strategy to asses stability is to analyze the reachable sets under the given feed-back law, and determine sufficient conditions to ensure convergence to the cylinder.

2.1 Reachable sets of closed loop system

In the closed loop system the state evolves under a piecewise constant torque, hence the corresponding jerk cannot be constant. In [1], we devised feedback laws corresponding to the two desired values of jerk, zero and j_{\max} .

¹Note that once the cylinder is reached, we apply the desired final torque and we let the system evolve along its natural modes. This means that the complex modes give rise to longitudinal oscillation and hence to negative jerk. However this oscillation has a magnitude less than the threshold of perception, it is for this reason that the constraints on the jerk are a concern only outside the cylinder.

In particular we devised a closed form for the input torque, corresponding to desired jerk zero, which keeps the jerk positive and achieves the zero value only at the final point.

Claim 1 *During the evolution of the closed loop system, under constant torque $u(t) = u_k \forall t \in [t_k, t_{k+1})$, the minimum value of the jerk is always attained at the end points.*

Recall that in [1] we defined the output of the system as

$$j = cA \zeta + cb u. \quad (1)$$

Moreover, since the torque $u(t)$ is constant between two transitions i.e. $\forall t \in [t_k, t_{k+1})$, the state evolution is given by

$$\zeta_{k+1} = \zeta(t_{k+1}) = \hat{A}\zeta_k + \hat{b}u_k, \text{ where}$$

$$\hat{A} = e^{A\tau}; \quad \hat{b} = (\hat{A} - I)A^{-1}b; \quad \tau = t_{k+1} - t_k.$$

Then the value of the jerk at the end of the period is

$$j_{k+1} = cA(\hat{A}\zeta_k + \hat{b}u_k) + cb u_k$$

imposing $j_{k+1} = 0$ leads to

$$u_k = -(c\hat{A}\hat{b} + cb)^{-1}(c\hat{A}\hat{A}\zeta_k) \quad (2)$$

which is exactly the value of the torque used in the proposed feedback law, where the solution to the continuous time optimal control problem calls for $j = 0$.

We now plot the set of states which are mapped onto the surface of the cylinder in one step. In figure (2) we

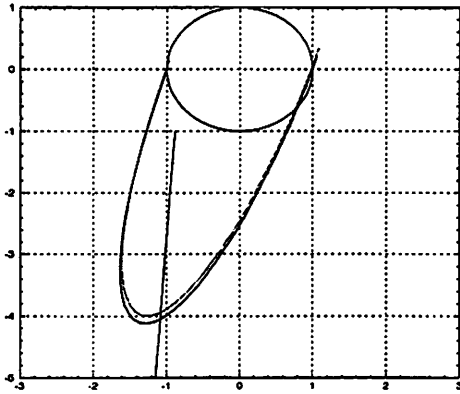


Figure 2: $Pre^1_{(j=0, \tau)}(C_\zeta)$

plotted the Pre set $Pre^1_{(j=0, \tau)}(C_\zeta)$ of the cylinder under control (2), assuming constant speed² ($\omega_c = 1000$ rpm).

Points are in the x coordinates, namely the natural modes decomposition coordinates (Real Jordan form), more specifically the figure shows the projection onto the x_2, x_3 plane.

Data are obtained scanning the surface ∂C_ζ along a set of circumferences. For various values of x_1 we projected backwards for time τ the points on the circumference $x_2^2 + x_3^2 = 1$ obtaining a curve. Given a point $\bar{\zeta} \in \partial(C_\zeta)$ and a timestep τ , the preceding point is computed as

$$\bar{\zeta}_{pre} = \hat{A}^{-1}(\bar{\zeta} - B_d u_{\bar{\zeta}}), \text{ where}$$

$$u_{\bar{\zeta}} = -(cb)^{-1}cA(\bar{\zeta})$$

is obtained directly from (1).

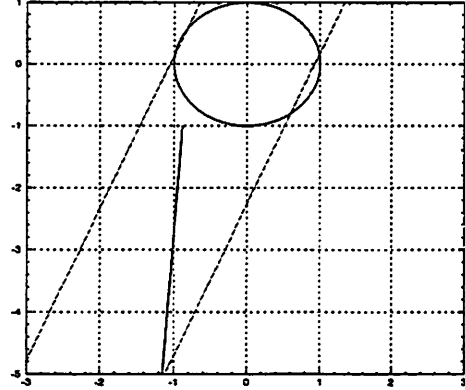


Figure 3: $Pre^3_{(j=0, \tau)}(C_\zeta)$

Figure (3) shows the set $Pre^3_{(j=0, \tau)}(C_\zeta)$, namely the set of points that reach C_ζ in three steps under control (2). Finally figure (4) shows sets $Pre_{(j=0, \tau_i)}(C_\zeta)$, i.e. the Pre sets in multiple steps (a relatively large number of steps) for engine speeds $\omega_c = 1000, 3000, 5000, 7000$ rpm.

The curves slightly rotate as x_1 changes. We are particularly interested on the vertical portion in the lower right quadrant, which represents the last points which will reach the cylinder under (2). The term “last” refers

²In reality engine speed is one of the components of the state (roughly proportional to x_1), so the constant speed assumption is not exact. However we use this approximation in a conservative manner. In fact we use the same constant speed for all initial conditions of interest, and we determine the minimum speed for which the closed loop system is convergent. Then we allow the use of the control scheme only for speeds higher than the minimum, thus implicitly eliminating the regions of the space where the approximation of constant speed was not conservative (in the reachable sets computation we used a constant speed higher than the (varying) one corresponding to those states).

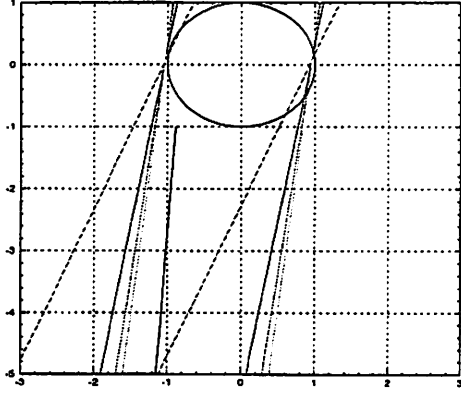


Figure 4: $Pre_{(j=0,\tau)}(C_\zeta)$

to the fact that under $j = j_{\max}$ the state evolves from left to right (increasing values of x_2), so that if the transition is delayed we may fall outside of $Pre_{(j=0,\tau)}(C_\zeta)$ and to the right of the interesting portion of the boundary $\partial Pre_{(j=0,\tau)}(C_\zeta)$.

From this picture we see how for smaller values of x_1 we get (slightly) tighter conditions on the switching accuracy (the boundary is closer to the switching surface). So the rest of the analysis will only report the data relative to the smallest value of x_1 which is of interest in the application.

2.2 Projection of the Switching Surface

Now we want to check what happens when crossing the switching surface under $j = j_{\max}$ motion. The most critical condition corresponds to having one decision point infinitely close to the switching surface but still before it, so that the next decision point is delayed with respect to the continuous case, by exactly τ seconds. Figure (5) shows these projections for various engine speeds. For higher engine speeds the projection is further away from the surface itself and the frontier $\partial Pre_{(j=0,\tau)}(C_\zeta)$ is closer to it. From figure (5) we see that for engine speed of 1000 rpm and lower the switching surface is projected outside $Pre_{(j=0,\tau)}(C_\zeta)$ and for engine speed of 3000 rpm and higher it is projected inside it. Hence there exists a speed $\omega_{c\min}$ for which the projection of the switching surface is tangential to the frontier $\partial Pre_{(j=0,\tau)}(C_\zeta)$. We performed a binary search to find such speed ($\omega_{c\min} \simeq 1685$ rpm) and the corresponding projection is shown in figure (6).

2.3 Stability Analysis

From the previous analysis we can ensure stability of the closed loop system (convergence to the cylinder C_ζ)

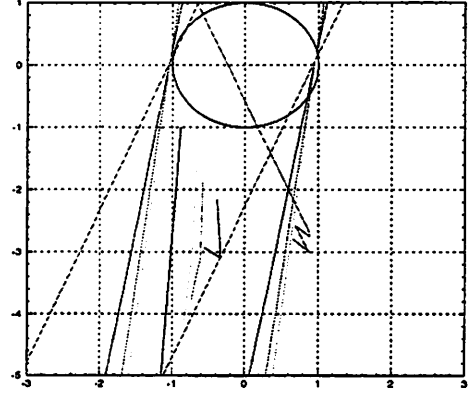


Figure 5: Projection of the switching surface

for engine speed in the range [1685, 7000] rpm.

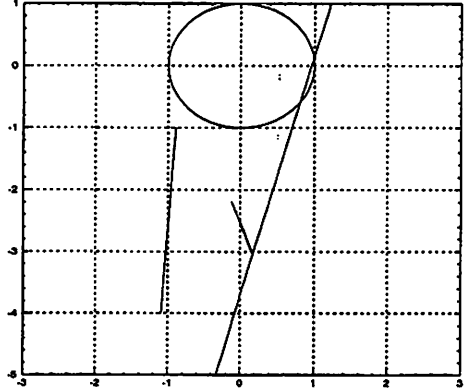


Figure 6: Tangential condition ($\omega_{c\min} \simeq 1685$ rpm)

However the normal range of operation of a car engine can be approximately [800, 7000] rpm, so it would be interesting to modify the switching surface so that we achieve convergence for lower engine speed also. Again, from the analysis of the reachable sets one can see how “anticipating” the switching surface improves stability in the sense that the projected points are inside the $Pre_{(j=0,\tau)}(C_\zeta)$ for lower engine speed. The original switching surface was obtained from the minimum time optimal solution of the continuous time problem, so moving away from it is supposedly going to reduce performance of the closed loop system.

One possible choice for the switching surface is the

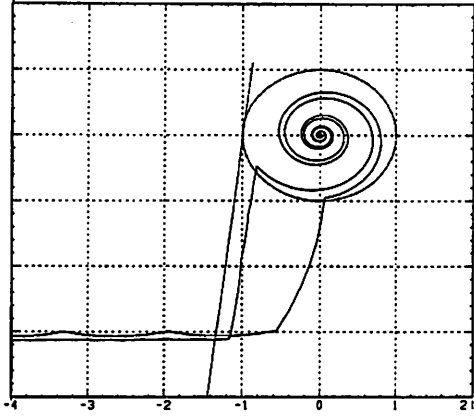


Figure 7: Using $\partial Pre_{(j=0,\tau)}(C_\zeta)$ for $\tau = 30/7000$, as the switching surface

other boundary of the set $Pre_{(j=0,\tau)}(C_\zeta)$ itself, for some fixed τ . Figures (7) and (8) report closed loop trajectories for two different choices of the τ parameter, namely corresponding to 7000 rpm and 1000 rpm respectively. Using the first choice we increase the stability range (evaluated with the binary search algorithm described above) to [1437, 7000] rpm and with the second choice it increased to [1120, 7000] rpm. However the second choice looks inadequate at high engine speed, in fact at higher speed the feedback law is able to keep the jerk closer to zero and consequently trajectories are steeper. In particular trajectories happen to cross the switching surface more than once hence producing a slower motion and reduced performance.

A better choice is a switching surface that anticipates the switching at low speed more than it does at high speed.

2.4 Speed Varying Switching Surface

We want to leave the switching surface unchanged at high revolution speed and anticipate it at lower speed. We decided to use the same switching surface as in the continuous time case only adding a component depending on the x_1 coordinate. This allows us to maintain the same slope in $x_2 - x_3$ plane and avoid multiple switchings. To pick a “good” anticipation factor, we used the intersection line of the asymptotic plane $\beta_{j_{\max}}$ and the $\partial Pre_{(j=0,\tau)}(C_\zeta)$ corresponding to 1000 rpm. The asymptotic plane is described by

$$\beta_{j_{\max}} = \left\{ \xi \in \mathbb{R}^3 \mid [001]\xi = -\frac{1}{\eta}[001]b_\xi j_{\max} \right\}$$

in the modal decomposition coordinates ξ of the inverse system (the one with jerk as input). This plane

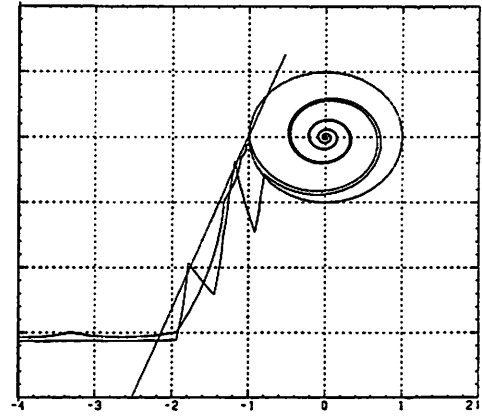


Figure 8: Using $\partial Pre_{(j=0,\tau)}(C_\zeta)$ for $\tau = 30/1000$, as the switching surface

is the “slow manifold” under constant jerk $j = j_{\max}$, for the continuous dynamics. The trajectories of the hybrid system under j_{\max} reach the switching surface chattering close to this plane. Finally we calculated the point x_{\min} on the intersection of the line and plane $x_1 = x_{1\min}$, and modified the switching surface so that it passed through this point. The resulting surface and closed loop trajectories are depicted in figures (9) and (10). For this choice of the switching surface the binary search algorithm provides as an estimate of the stability range [1205, 7000] rpm.

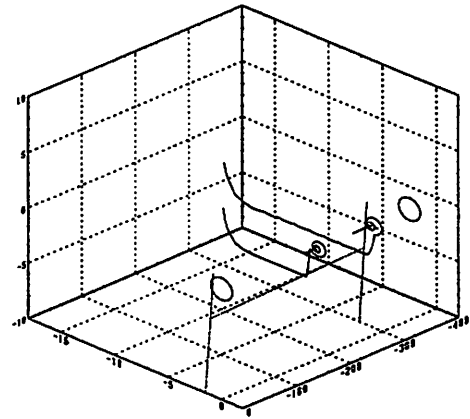


Figure 9: Trajectories of the system in the x state space

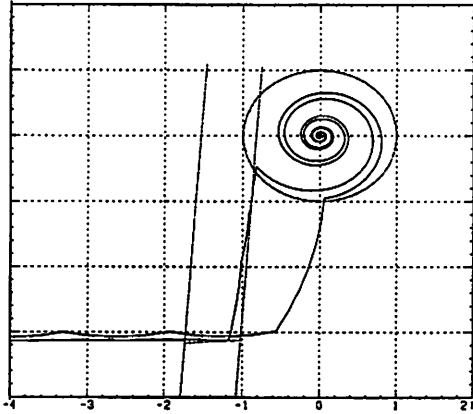


Figure 10: Trajectories of the system in the x_1, x_2 state plane

3 Conclusions and Future Work

We assessed the stability of the feedback law proposed in [1] based on the analysis of the reachable sets. We devised a binary search algorithm for finding the minimum speed to achieve convergence. In particular we gave estimates of the engine speed range for which the closed loop system converges to the cylinder C_c ([1685, 7000] rpm).

Moreover we modified the switching surface so as to increase this range and not to lose performance at high speed. The estimated convergence range is in this case [1205, 7000] rpm.

We intend to calculate analytical bounds to quantify how distant the hybrid solution that we found is from the continuous time optimal solution, in terms of performance (time to reach the cylinder). This bound is also a bound to quantify the distance of the proposed hybrid solution from the (unknown) optimal hybrid solution since, for any fixed initial condition, the following relation holds

$$T_{optcont} \leq T_{opthyb} \leq T_{proposedhyb}.$$

The rightmost relation comes from the definition of optimal solution and the leftmost comes from the definition of relaxed control problem. In the continuous time problem in fact, we have less constraints than in the hybrid problem.

Moreover we would like to include the air dynamics in the optimal control problem in the continuous case, so to obtain more realistic (less conservative) estimates of the time needed to reach C_c .

References

- [1] A. BALLUCHI, M. D. BENEDETTO, C. PINELLO, AND A. SANGIOVANNI-VINCENTELLI, *A hybrid approach to the fast positive force transient problem in automotive engine control*, in 37th CDC, Tampa, FL, 1998.

A Hybrid Approach to the Fast Positive Force Transient Tracking Problem in Automotive Engine Control¹

A. Balluchi† M. Di Benedetto‡ C. Pinello‡

A. Sangiovanni-Vincentelli‡§

†PARADES, Via San Pantaleo, 66, 00186 Roma, Italy, balluchi,pinello,alberto@parades.rm.cnr.it

‡Dip. di Ingegneria Elettrica, Università dell'Aquila, Poggio di Roio, 67040 L'Aquila, Italy, dibenede@giannutri.caspur.it

§Dep. of Electrical Engineering and Computer Science, University of California at Berkeley, CA 94720, alberto@eecs.berkeley.edu

Abstract

The engine control problem can be decomposed into a set of sub-problems corresponding to *regions of operation* identified by the settings of the control devices available to the driver (for example accelerator pedal angle, selected gear). One of these regions is the so called Fast Positive Force Transient, where a quick acceleration is requested, maintaining certain comfort standards. In this paper, this control problem is formulated as a hybrid control problem and solved by approximation of an auxiliary continuous control problem. The quality of the results is backed by a set of simulations on a commercial car model.

1 Introduction

Automotive engine control is an important application domain for hybrid systems (see e.g. [5]) and for control in general. We argue that most of the engine control problems are hybrid control problems since the plant itself, having discrete as well as continuous components (e.g. engine cycle and power-train), is described by a hybrid model. In our approach, the system specifications are captured using a top-level Finite State Machine (FSM), whose states correspond to different regions of operation of the engine. The transitions are determined by driver's actions or by engine conditions. Each region of operation is characterized by a set of constraints, related to driving performance like comfort and safety or gas and noise emission, and a cost function that identifies the desired behavior of the controlled system. The goal of the controller is to act on the inputs to the plant so that it behaves according to the specifications summarized in the FSM.

In previous papers [1, 2], we introduced a hybrid model for the engine to solve the cut-off control problem, a sub-set of the control problem corresponding to the

Fast Negative Force Transient region of operation. In this paper, in our quest for a general solution to the engine control problem, we consider the control problem corresponding to the *Fast Positive Force Tracking* region of operation. In this state, we have to react to a fast gas pedal motion that is interpreted as a request for a fast increase of the torque delivered by the engine while maintaining a reasonable level of comfort, specified in terms of vehicle acceleration and jerk. The goal is to control the evolution of the system from an initial condition characterized by the delivery of a torque u_0 to a final condition characterized by the delivery of a requested torque u_R in minimum time subject to constraints on acceleration and jerk. The available control actions are on fuel injection, spark ignition and, since we are considering cars equipped with drive-by-wire electronics, throttle angle (which regulates the air entering the cylinders). Our approach to the hybrid problem at hand is to first introduce an auxiliary relaxed problem: a continuous time problem with jerk as input. This problem is solved optimally. Then the solution is mapped back in the hybrid domain obtaining feedback laws for the throttle angle, the spark advance and the fuel injection inputs. The quality of the control law is demonstrated by a set of simulations on a model of a commercial car.

2 Problem formulation

2.1 Plant model

The hybrid model of a vehicle with a 4-stroke N -cylinder gasoline engine proposed in [2] is here reviewed. Such model is the composition of N sub-models, one per cylinder (see Figure 1). Each of them consists of: a FSM describing internal combustion engine's cycle (direct graph); a Discrete Event (DE) system modeling torque generation (dashed boxes); a Continuous Time (CT) dynamics modeling power-train and air dynamics (solid boxes).

Power-train model. The power-train behavior is described by the linear CT dynamics

$$\dot{\zeta} = A\zeta + bu \quad (1)$$

¹This research has been partially sponsored by PARADES, a Cadence, Magneti-Marelli and SGS-Thomson GEIE, and by CNR. ISI provided the X-Math environment to carry out the simulation.

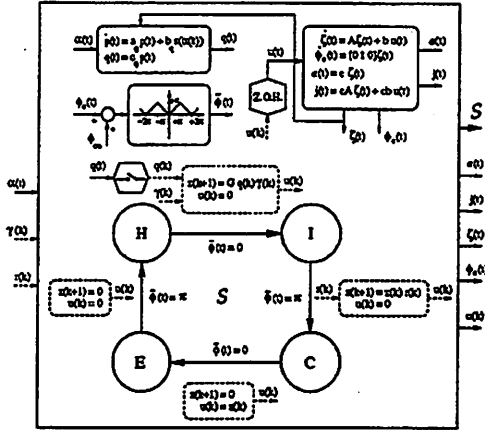


Figure 1: Hybrid model for a single cylinder engine.

$$\dot{\phi}_c = [0 \ 1 \ 0] \zeta \quad (2)$$

where ζ components are: the axle torsion angle α_e , crankshaft revolution speed ω_c , and wheel revolution speed ω_p ; and ϕ_c denotes the crankshaft angle. The input u is the torque acting on the crank. Being a passive system, dynamics (1) is asymptotically stable and has a real dominant pole λ_1 , and a pair of complex poles $\lambda \pm j\mu$. Assuming vehicle speed equal to peripheral wheel speed, vehicle acceleration is $a = R_w \dot{\omega}_p$, where R_w is the wheel radius. Let $c \in \mathbb{R}^{1 \times 3}$ be the product between R_w and the third row of A , and let j denote vehicle jerk. Since the third entry in b is 0, we have

$$a = c \zeta, \quad (3)$$

$$j = \frac{da}{dt} = (cA) \zeta + (cb) u. \quad (4)$$

Cylinder's behavior. The behavior of each cylinder in the internal combustion engine is represented by the FSM. State S assume values in the set $\{I, C, E, H\}$ related to the *intake*, *compression*, *expansion* and *exhaust* strokes. State transitions occur when the piston reaches the bottom or top dead center. Guard conditions are written in terms of the piston position expressed by ϕ , which denotes the absolute value of the crank angle w.r.t. the upper dead center position. ϕ_{co} is the angle the crank is mechanically mounted on the shaft. The torque produced by the cylinder during its expansion phase is modeled as a constant signal.

Torque generation. The torque generation mechanism is characterized by a transport process represented by a DE system synchronized with FSM transitions. At each time t_k where a transition occurs the event counter k is incremented by one. The DE output $u(k)$ is converted by the zero-order hold block to the piece-wise constant signal $u(t) = u(k)$ for $t \in [t_k, t_{k+1})$, which feeds the power-train. DE inputs are: the mass of air $q(k) \in \mathbb{R}^+$ loaded in the intake stroke, which depends on air dynamics; the mix composition factor

$\gamma(k) \in \{0\} \cup [\gamma_{\min}, \gamma_{\max}]$ which represents the ratio between the mass of fuel injected and the mass of air loaded, normalized w.r.t. the stoichiometric value; the spark modulation factor $r(k) \in [r_{\min}, 1]$ due to spark ignition timing. If $\gamma = 0$ no fuel is injected. At the $H \rightarrow I$ transition, the maximum amount of torque achievable during the next expansion phase, from the given air $q(k)$ and fuel $\gamma(k)$, is stored in the DE state $z \in \mathbb{R}$; then at the $I \rightarrow C$ transition the spark factor $r(k)$ is applied and at the $C \rightarrow E$ transition the torque $u(k)$ is output based on the value stored in z .

Air dynamics. The model of the quantity of air entering the cylinder during the intake stroke is obtained from the equation of the flow of a compressible fluid through a converging nozzle, whose section is controlled by the throttle valve [3]. Let $p(t)$, $\alpha(t)$ and $q(t)$ resp. denote the manifold pressure, the throttle angle, and the mass of air loaded by the cylinder at time t . We have

$$\dot{p} = a_q(\omega_c, p) p + b_q(p) s(\alpha) \quad (5)$$

$$q = c_q(\omega_c, p) p \quad (6)$$

where $s(\alpha)$ is the so called equivalent throttle area and

$$a_q(\omega_c, p) = -\frac{V_d}{4\pi(V_d + V_m)} \eta_v(\omega_c, p) \omega_c, \quad (7)$$

$$b_q(p) = \frac{p_{atm} \sqrt{R_g T_{atm}}}{V_d + V_m} \beta\left(\frac{p}{p_{atm}}\right), \quad (8)$$

$$c_q(\omega_c, p) = \frac{V_d}{4R_g T_{atm}} \eta_v(\omega_c, p), \quad (9)$$

where V_d, V_m are resp. the displacement and the manifold volume, p_{atm}, T_{atm} are the ambient pressure and air temperature, R_g is the air gas constant, $\eta_v(\omega_c, p)$ is the volumetric efficiency in cylinder intakes and

$$\beta(y) = \begin{cases} \sqrt{\frac{2\gamma_c}{\gamma_c-1}} \left[y^{\frac{2}{\gamma_c}} - y^{\frac{\gamma_c+1}{\gamma_c}} \right] & \text{if } y \geq \left(\frac{2}{\gamma_c+1} \right)^{\frac{\gamma_c}{\gamma_c-1}} \\ \sqrt{\gamma_c} \left(\frac{2}{\gamma_c+1} \right)^{\frac{\gamma_c+1}{2(\gamma_c-1)}} & \text{if } y < \left(\frac{2}{\gamma_c+1} \right)^{\frac{\gamma_c}{\gamma_c-1}} \end{cases}$$

with $\gamma_c = \frac{c_p}{c_v}$. While in the past the throttle valve was directly connected to the gas pedal, in modern cars equipped with drive-by-wire, the throttle valve is controlled by the engine control unit.

Engine hybrid model. The overall model of torque generation for a N -cylinder engine is the combination of N FSMs and of N DE systems representing the behavior of each cylinder. The hybrid model of the complete engine is obtained by adding to the torque generation model the power-train CT dynamics (1), (2) and air CT dynamics (5) which are shared among all cylinders. In this paper we focus on the most relevant case of a 4-cylinder engine, whose model is referred to as \mathcal{M}_{4cyl} . \mathcal{M}_{4cyl} inputs are: the throttle angle α , a CT signal in the class \mathcal{A}_{4cyl} of functions $\mathbb{R}^+ \rightarrow [0, \frac{\pi}{2}]$; the mix composition factors $\gamma =$

$[\gamma_1, \gamma_2, \gamma_3, \gamma_4]^T$, a DE signal in the class \mathcal{G}_{4cyl} of functions $\mathbb{N} \rightarrow (\{0\} \cup [\gamma_{\min}, \gamma_{\max}])^4$, with γ_i synchronized with the i -th DE model; the spark modulation factors $r = [r_1, r_2, r_3, r_4]^T$, a DE signal in the class \mathcal{R}_{4cyl} of functions $\mathbb{N} \rightarrow [r_{\min}, 1]^4$, with r_i synchronized with the i -th DE model. \mathcal{M}_{4cyl} states are: $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$, $\mathcal{z} = [z_1, z_2, z_3, z_4]^T$ and (ζ, ϕ_c, p) . The states of \mathcal{S} are constrained by the mechanics of the four-stroke engine to the following set: $(H, I, C, E), (I, C, E, H), (C, E, H, I), (E, H, I, C)$. Without loss of generality, let $\phi_{co_1} = \phi_{co_3} = \pi, \phi_{co_2} = \phi_{co_4} = 0$.

2.2 The optimization problem

The objective is to steer the system from a given point, characterized by the torque delivered to the crankshaft $u(0) = u_0$, to a new point with torque value $u_R > u_0$ in minimum time satisfying comfort requirements. The oscillating component of vehicle acceleration and the vehicle jerk have been shown experimentally to be the most important factors in passenger comfort in the FPFT region of operation. The control problem is formulated as follows: *steer the power-train elastic state, keeping the jerk bounded, i.e.,*

$$0 \leq j(t) \leq j_{\max} \quad (10)$$

to a point such that the application of the new requested value of transmitted torque produces an oscillating acceleration evolution $\tilde{a}(t)$ bounded above by a threshold of perception $\tilde{a}_{th} > 0$.

Introduce $v = u - u_R$ and the transformed state

$$\begin{bmatrix} x' \\ x \end{bmatrix} = N_{\zeta x} (\zeta + A^{-1}b u_R), \quad (11)$$

with $x' \in \mathbb{R}$, $x \in \mathbb{R}^2$ and $N_{\zeta x} \in \mathbb{R}^{3 \times 3}$ such that dynamics (1),(3),(4) are rewritten as follows

$$\begin{bmatrix} \dot{x}' \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & A_x \end{bmatrix} \begin{bmatrix} x' \\ x \end{bmatrix} + \begin{bmatrix} b_{x'} \\ b_x \end{bmatrix} v \quad (12)$$

$$\begin{bmatrix} \dot{a} \\ j \end{bmatrix} = \begin{bmatrix} c_{x'} & c_x \\ c_{x'}\lambda_1 & c_x A_x \end{bmatrix} \begin{bmatrix} x' \\ x \end{bmatrix} + \begin{bmatrix} -cA^{-1}b u_R \\ (cb)v \end{bmatrix} \quad (13)$$

with $A_x = \begin{bmatrix} \lambda_1 & 0 \\ 0 & A \end{bmatrix}$, $b_x = N_{\zeta x}b$ and $[c_{x'} \ c_x] = c N_{\zeta x}^{-1}$. Under constant control v in (12), the oscillating component of the acceleration can then be expressed as

$$\tilde{a} = c_x x. \quad (14)$$

Without loss of generality, let $N_{\zeta x}$ in (11) be chosen such that $\|c_x\| = \tilde{a}_{th}$. Consider the manifold

$$\mathcal{C}_x = \left\{ \begin{bmatrix} x' \\ x \end{bmatrix} \in \mathbb{R}^3 \mid x' \in \mathbb{R}, x = \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}, \theta \in [0, 2\pi) \right\} \quad (15)$$

Since the norm of $x(t)$ in (12) decreases over time when $v(t) = 0$, if at some time \bar{t} , the state has been driven to $x(\bar{t}) \in \mathcal{C}_x$, signal $v(t) = 0$, i.e. $u(t) = u_R$, keeps

the trajectory inside \mathcal{C}_x with acceleration $\tilde{a}(t)$ bounded above by the threshold value \tilde{a}_{th} , for all $t \geq \bar{t}$. Let

$$\mathcal{C}_{\zeta} = \left\{ \zeta \in \mathbb{R}^3 \mid \zeta = N_{\zeta \zeta} \begin{bmatrix} x' \\ x \end{bmatrix} - A^{-1}b u_R, \begin{bmatrix} x' \\ x \end{bmatrix} \in \mathcal{C}_x \right\} \quad (16)$$

with $N_{\zeta \zeta} = N_{\zeta x}^{-1}$ and \mathcal{C}_x as in (15). The optimal control problem is:

Problem 1 *Given the engine hybrid model \mathcal{M}_{4cyl} , for any ζ_0 not inside the region delimited by \mathcal{C}_{ζ} as in (16),*

$$\begin{aligned} & \min_{\substack{\alpha \in \mathcal{A}_{4cyl} \\ \gamma \in \mathcal{G}_{4cyl} \\ r \in \mathcal{R}_{4cyl}}} \int_0^T dt \\ & \text{subject to: } \begin{cases} \text{Dynamics of Hybrid Model } \mathcal{M}_{4cyl} \text{ with} \\ \mathcal{S}^o = (H, I, C, E), \\ \mathcal{z}(0) = [0, Gq_0, Gq_0, Gq_0]^T, \\ \zeta(0) = \zeta_0, \phi_c(0) = 0, p(0) = p_0, \\ \zeta(T) \in \mathcal{C}_{\zeta}, \\ 0 \leq j(t) \leq j_{\max} \text{ for all } t \in [0, T], \end{cases} \end{aligned}$$

with $(\zeta_0^T, 0)^T$ the power-train dynamics state value at the initial time and $p_0, q_0 = c(\omega_c(0), p_0)p_0$ the corresponding manifold pressure and mass of air.

3 Relaxed continuous-time problem

In this section a relaxation of Problem 1 to the continuous-time domain is defined and solved. The relaxed problem is concerned with comfort requirements, as specified in (10), for minimum time optimal trajectories of system (1) to manifold (16), assuming no constraint on torque signal. The solution is easily obtained by rewriting dynamics (1) with input j in place of u . From (1) and (4),

$$\dot{\zeta} = (I - b(cb)^{-1}c)A \zeta + (cb)^{-1}b j, \quad (17)$$

$$u = -(cb)^{-1}(cA)\zeta + (cb)^{-1}j. \quad (18)$$

System (17)-(18) corresponds to the inverse of system (1)-(4). For $a = c\zeta = 0$, hence for $j = 0$ (17) represents the zero-dynamics of system (1)-(4) which has a pole in the origin and two poles equal to the zeros of $c(sI - A)^{-1}b$. Since, c is proportional to the third row of A and the third entry in b is 0, pole $s = 0$ of (17) has multiplicity two. Let η denote the third real pole. The relaxed problem is easily solved in the transformed space $\xi = N_{\zeta \xi}(\zeta + A^{-1}b u_R)$ of the natural modes, where dynamics (17) is written as

$$\dot{\xi} = A_{\xi} \xi + b_{\xi} j \text{ with } A_{\xi} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \eta \\ 0 & 0 & 0 \end{bmatrix}. \quad (19)$$

Let \mathcal{J} be the class of functions $\mathbb{R}^+ \rightarrow [0, j_{\max}]$ and

$$\mathcal{C}_{\xi} = \left\{ \xi \in \mathbb{R}^3 \mid \xi = N_{\xi \xi} \begin{bmatrix} x' \\ x \end{bmatrix} \text{ with } \begin{bmatrix} x' \\ x \end{bmatrix} \in \mathcal{C}_x \right\} \quad (20)$$

with $N_{x\xi} = N_{\xi\xi}N_{x\xi}$ and C_x as in (15). Define the relaxed optimal control problem

Problem 2 For any ξ_0 not inside C_ξ as in (20)

$$\min_{j \in \mathcal{J}} \int_0^T dt$$

$$\text{subject to: } \begin{cases} \dot{\xi} = A_\xi \xi + b_\xi j \\ \xi(0) = \xi_0 \\ \xi(T) \in C_\xi \end{cases}$$

Proposition 1 Let $\hat{j}(t)$, with $t \in [0, T]$, be an optimal solution to Problem 2 and let $\hat{\xi}(t)$ be the corresponding minimum time trajectory to manifold C_ξ . Then, $\hat{j}(t) \in \{0, j_{\max}\}$ for all $t \in [0, T]$ and the final control value is

$$\hat{j}(T) = \begin{cases} 0 & \text{if } b_\xi^T Q \xi(T) > 0 \\ j_{\max} & \text{if } b_\xi^T Q \xi(T) < 0 \end{cases}, \quad (21)$$

where $Q = N_2^T N_2 + N_3^T N_3$, with N_2, N_3 the second and third row of $N_{x\xi}^{-1}$, respectively. Moreover there exist at most two times $t_1, t_2 \in [0, T]$ where a switching of $\hat{j}(t)$ takes place. Times t_1, t_2 are the solution to

$$[b_\xi^1 + b_\xi^2(T - t_1), b_\xi^2, b_\xi^3 e^{\eta(T-t_1)}] Q \xi(T). \quad (22)$$

The proof of Proposition 1, omitted here for lack of space, is based on Pontryagin's Principle [6]. Integrating backwards (19), from a given $\xi(T) = \bar{\xi} \in C_\xi$, with final control chosen according to (21) and switching instants obtained from (22), minimum time trajectories to C_ξ in the ξ state space are obtained. If (22) has no solutions in $(0, T)$ then \hat{j} as in (21) for $t \in [0, T]$, is optimal. Otherwise, in the backward integration of (19) \hat{j} switches at time t_1 , and, if two solutions to (22) in $(0, T)$ exist, at time t_2 . Points $\xi(t_1), \xi(t_2)$ belong to 2-dimensional surfaces which define a partition $S_\xi^0 \cup S_\xi^{j_{\max}}$ of the set of points controllable to C_ξ in time lower than or equal to T , so that the solution to Problem 2 is expressed as:

$$\hat{j}(\xi) = \begin{cases} 0 & \text{if } \xi \in S_\xi^0 \\ j_{\max} & \text{if } \xi \in S_\xi^{j_{\max}} \end{cases}. \quad (23)$$

From (18) and (23), mapping $S_\xi^0 \cup S_\xi^{j_{\max}}$ to the physical state space the minimum time torque is written as:

$$\hat{u}(\zeta) = \begin{cases} -\frac{cA}{cb} \zeta & \text{if } \zeta \in S_\zeta^0 \\ -\frac{cA}{cb} \zeta + \frac{j_{\max}}{cb} & \text{if } \zeta \in S_\zeta^{j_{\max}} \end{cases}. \quad (24)$$

4 Hybrid system control scheme

In this Section we propose feasible feedback laws for throttle angle $\alpha(t)$, mix composition $\gamma(k)$ and spark modulation $r(k)$ such that the torque u generated by

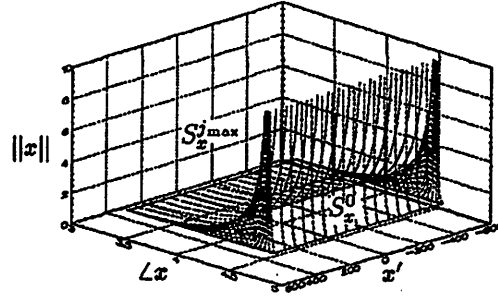


Figure 2: Partition $S_x^0, S_x^{j_{\max}}$, defining the minimum time jerk, in the (x', x) space represented in cylindric coordinate. Manifold C_x is the plane $\|x\| = 1$.

the hybrid model \mathcal{M}_{4cyl} tracks (24). The main difficulties are:

(a) the generated torque is limited to $\{0\} \cup [r_{\min} \gamma_{\min} q, \gamma_{\max} q]$, where the air mass q is subject to manifold pressure dynamics;

(b) torque generation is synchronized with the power-train dynamics;

(c) there is a delay between the time at which γ and r are set and the time at which the torque is generated. As discussed in Section 2, the spark modulation $r_m(k)$, applied at m -th cylinder which enters the C state at time t_k , affects torque $u(k+1)$ generated at time t_{k+1} , while the mass $q(k)$ of air loaded and the mix composition $\gamma_\ell(k)$ of the ℓ -th cylinder, which enters the H state, affect torque $u(k+2)$. Hence, feedbacks for r_m and γ_ℓ are expressed in terms of a prediction of the point reached by ζ respectively at times t_{k+1} and t_{k+2} , obtained by integration of (1). The torque generation process can be viewed as a MIMO system composed of two interconnected sub-systems, as depicted in Figure 3. The air mass evolution $q(t)$ is subject to manifold pressure dynamics (5), which is controlled by throttle angle α and depends on the crankshaft revolution speed ω_c . The torque to the crankshaft is provided by a DE system, whose evolution depends on $q(k)$, with two dimensional state and inputs $\gamma_\ell(k), r_m(k)$. The coupling between the DE system and pressure dynamics is given by the power-train dynamics (1), with input u and output ω_c . While the impact on the DE system of signal q is not negligible, the power-train impact on air dynamics is weak enough to allow a decentralized control (see [7]).

4.1 Pressure Dynamics Decentralized Control

The reference evolution \hat{q} for the air mass is obtained considering a rigid model for the power-train and solving the minimum time problem assuming $r_m = \gamma_\ell = 1$. Let τ_{drv}, M_v resp. denote the driveline transmission ratio and the vehicle equivalent mass. In the power-train rigid model, the force acting on the vehicle is $\frac{u}{R_w \tau_{drv}}$ and $j = (M_v R_w \tau_{drv})^{-1} \frac{du}{dt}$. Let $q_R = u_R / G$. The mini-

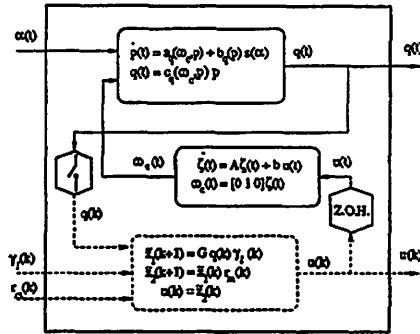


Figure 3: Torque generation model.

imum time air mass evolution is

$$\hat{q}(t) = \begin{cases} q(0) + \frac{j_{\max} M_y R_w T_{drv}}{G} t & \text{if } t < \frac{G(q_R - q(0))}{j_{\max} M_y R_w T_{drv}} \\ q_R & \text{if } t \geq \frac{G(q_R - q(0))}{j_{\max} M_y R_w T_{drv}} \end{cases} \quad (25)$$

A decentralized scheme for perfect tracking of reference trajectory (25) is presented (e.g. [4]). Rewrite (5) as

$$\dot{p} = a_q(\omega_c, p) p + w. \quad (26)$$

Consider $\hat{q}(t)$ as in (25) and introduce $\sigma(t) = q(t) - \hat{q}(t)$. From (25), $\sigma(0) = 0$. The equivalent control (see [8])

$$w_{eq} = -a_q p + \left(c_q + \frac{\partial c_q}{\partial p} p \right)^{-1} \left(\dot{q} - \frac{\partial c_q}{\partial \omega_c} p \dot{\omega}_c \right).$$

ensures $\sigma(t) = 0$ if $\sigma(0) = 0$. While \hat{q} is known, ω_c is not available. However, power-train weak coupling allows us to employ a robust decentralized scheme where nonlinearities are compensated. Given a value $\bar{\eta}_v$, $\bar{\omega}_c$ of volumetric efficiency and crankshaft speed, introduce $a_{q1} = -\frac{V_d}{4\pi V_c} \bar{\eta}_v \bar{\omega}_c$, $c_{q1} = \frac{V_d}{4R_q T_{atm}} \bar{\eta}_v$, and $\bar{a}_q(\omega_c, p) = a_q(\omega_c, p) - a_{q1}$, $\bar{c}_q(\omega_c, p) = c_q(\omega_c, p) - c_{q1}$. If $\bar{a}_q = \bar{c}_q = 0$, for any $W > 0$, the variable structure control

$$w_{usc} = -a_{q1} p + \frac{\dot{q}}{c_{q1}} - \frac{W}{c_{q1}} \operatorname{sign}(\sigma) \quad (27)$$

guarantees a sliding regime along the manifold $\sigma = 0$, during which perfect tracking of $\hat{q}(t)$ is achieved.

Proposition 2 Assume that $p, \omega_c, \dot{\omega}_c, \hat{q}$ satisfy $p \in [p_1, Patm]$, $\omega_c \in [\omega_1, \omega_2]$, $|\dot{\omega}_c| \leq \dot{\omega}_{max}$, $|\hat{q}| \leq \hat{q}_{max}$. Choose a $\bar{\eta}_v, \bar{\omega}$ and let $\bar{C}_1, \bar{C}_p, \bar{C}_{\omega_c}, \bar{A}_q$ be upper bounds on $\left| \frac{\bar{c}_q}{c_{q1}} \right|, \left| \frac{\partial \bar{c}_q}{\partial p} \right|, \left| \frac{\partial \bar{c}_q}{\partial \omega_c} \right|, |\bar{a}_q|$, respectively. Given $\epsilon > 0$, if $\bar{C}_1 + \bar{C}_p \frac{Patm}{c_{q1}} < 1$ the VSC (27) with

$$W = \frac{(\tilde{C}_1 + \tilde{C}_p \frac{patm}{c_{q1}}) \dot{q}_{max} + (\tilde{C}_{\omega_c} patm) \dot{\omega}_{max}}{1 - \tilde{C} - \tilde{C}_p \frac{patm}{c_{q1}}} + c_{q1} \tilde{A}_q patm + (1 - \tilde{C} - \tilde{C}_p \frac{patm}{c_{q1}})^{-1} \epsilon \quad (28)$$

guarantees a sliding motion on $\sigma = 0$, during which perfect tracking of $\hat{q}(t)$ as in (25) is achieved, robustly w.r.t. nonlinearities and power-train evolution.

To reduce the undesired chattering, typical in VSC, the sign function is replaced by $\frac{s}{\delta + |s|}$, where the smoothing parameter $\delta > 0$ is properly tuned so to maintain satisfactory tracking. The throttle angle feedback is

$$\alpha = s^{-1} \left(\frac{1}{b_q(p)} \left(-a_{q1} p + \frac{\dot{q}}{c_{q1}} - \frac{W}{c_{q1}} \frac{\sigma}{\delta + |\sigma|} \right) \right) \quad (29)$$

with $\sigma(t) = c_q(\omega_c, p) p - \hat{q}(t)$.

4.2 Torque Feedback Control

A feedback in terms of $\gamma(k)$ and $r(k)$ according to law (24) is presented in the sequel. Assuming that crankshaft speed does not change significantly, i.e. $t_{k+1} - t_k \approx \tau_k = \pi/\omega_c(t_k)$, since $u(t) = u(k)$ for $t \in [t_k, t_{k+1})$, dynamics (1),(3) are discretized as follows

$$\zeta(k+1) = \hat{A}\zeta(k) + \hat{b}u(k) \quad (30)$$

$$a(k) = c \zeta(k) \quad (31)$$

where $\hat{A} = e^{A\tau_k}$, $\hat{b} = (\hat{A} - I)A^{-1}b$. The jerk mean value in the time interval $[t_k, t_{k+1}]$ can be expressed as:

$$\frac{\int_{t_k}^{t_{k+1}} j dt}{t_{k+1} - t_k} \approx \frac{a(k+1) - a(k)}{\tau_k} = \frac{c(\hat{A} - I)}{\tau_k} \zeta(k) + \frac{c\hat{b}}{\tau_k} u(k).$$

Hence, the torque law $\hat{u}_d(\zeta(k))$ with

$$\hat{u}_d(\zeta) = (c\hat{b})^{-1}c(\hat{A} - I)\zeta + (c\hat{b})^{-1}\tau_k \hat{j}(\zeta), \quad (32)$$

where $\hat{j}(\zeta)$ is chosen according to (23), that is

$$\hat{j}(\zeta) = \begin{cases} 0 & \text{if } \zeta \in S_{\zeta}^0 \\ j_{\max} & \text{if } \zeta \in S_{\zeta}^{\max}, \end{cases} \quad (33)$$

produces a jerk with mean value $\hat{j}(\zeta)$. However, the jerk exhibits a ripple on its average value due to the fact that, being $u(t)$ piecewise-constant, between two samples the natural modes of dynamics (1) evolve. Due to this ripple, the jerk exceeds interval $[0, j_{\max}]$, and a more conservative feedback than (32) has to be devised.

Proposition 3 *Define*

$$\hat{u}_c(\zeta) = (cb)^{-1}(\hat{j}(\zeta) - cA\zeta)$$

$$\hat{u}_n(\zeta) = (cA\hat{b} + cb)^{-1}(\hat{j}(\zeta) - cA\hat{A}\zeta)$$

$$\hat{u}_m(\zeta) = (c(I + \tau_k A)b)^{-1}(j_{\max} - c(I + \tau_k A)A\zeta),$$

with $\hat{j}(\zeta)$ as in (33). Let $j_d(t)$, $j_c(t)$, $j_n(t)$ and $j_m(t)$ denote the jerk profiles under feedbacks $\hat{u}_d(\zeta(k))$, $\hat{u}_c(\zeta(k))$, $\hat{u}_n(\zeta(k))$ and $\hat{u}_m(\zeta(k))$ respectively, and assume that $\frac{d^2 j_d^2}{dt^2} \leq 0$, $\forall t \in [t_k, t_{k+1}]$. The feedback

$$\hat{u}(k) = \begin{cases} \max(\hat{u}_c(\zeta), \hat{u}_n(\zeta)) & \text{if } \zeta(k) \in S_\zeta^0 \\ \hat{u}_n(\zeta) \text{ if } j_d(t_{k+1}) > j_{\max} \wedge \frac{dj_d}{dt}(t_{k+1}) \geq 0 \\ \hat{u}_c(\zeta) \text{ if } j_d(t_k) > j_{\max} \wedge \frac{dj_d}{dt}(t_k) \leq 0 \\ \hat{u}_m(\zeta) \text{ otherwise} & \text{if } \zeta(k) \in S_\zeta^{j_{\max}} \end{cases} \quad (34)$$

produces a jerk profile $j(t) \in [0, j_{\max}]$.

Feedback laws for the spark modulation $r_m(k)$ and mix composition factor $\gamma_L(k)$ which generate the torque $\hat{u}(k)$ as in (34) are reported below.

```

 $\zeta_k = \zeta(t_k), \quad q_k = q(t_k)$       (power-train state and air mass)
 $r_k = \pi/[0 \ 1 \ 0] \zeta_k, \quad \hat{A} = e^{A\Delta t}, \quad \hat{b} = (A - I)^{-1}b$  (discrete model)
 $\zeta_{k+1} = \hat{A}\zeta_k + \hat{b}u_k^c(k)$       (future power-train state)

if  $\|N_{C_c}(\zeta_{k+1} + A^{-1}bu_R)\| > \beta$  then (exit condition: state  $\zeta$  in  $C_c$ )
     $\hat{u}_{k+1} = \hat{u}(k+1)$       (as in (34) evaluated at  $\zeta_{k+1}$ )
else
     $\hat{u}_{k+1} = u_R$       (target torque  $u_R$  applied)
endif

 $r = \hat{u}_{k+1}/s_1^c(k)$ 
if  $r < r_{\min}$  then  $r = r_{\min}$  elseif  $r > 1$  then  $r = r_{\max}$  endif
 $r_m(k) = r$       (spark advance control)

 $\zeta_{k+2} = \hat{A}\zeta_{k+1} + \hat{b}r_k s_1^c(k)$       (next power-train state)
if  $\zeta_{k+2} \in S_{C_c}^0$  then
     $\hat{j} = 0$ 
endif

if  $\|N_{C_c}(\zeta_{k+2} + A^{-1}bu_R)\| > \beta$  then (exit condition: state  $\zeta$  in  $C_c$ )
     $\hat{u}_{k+2} = \hat{u}(k+2)$       (as in (34) evaluated at  $\zeta_{k+2}$ )
else
     $\hat{u}_{k+2} = u_R$       (target torque  $u_R$  applied)
endif

 $\gamma = \hat{u}_{k+2}/(Qq_k)$ 
if  $\gamma < \gamma_{\min}$  then  $\gamma = \gamma_{\min}$  elseif  $\gamma > \gamma_{\max}$  then  $\gamma = \gamma_{\max}$  endif
 $\gamma_L(k) = \gamma$       (mix composition control)

 $s_1^c(k+1) = Qq_k \gamma$       (next potential torque)
 $s_2^c(k+1) = s_1^c(k) r$       (next predicted torque)
    
```

(35)

5 Simulation Results

The performance of the our hybrid control approach has been evaluated in a number of simulations. M_{4cyl} , with feedbacks α as in (29) and γ_L , r_m as in (35), has been captured in the Xmath/SystemBuild environment (by Integrated Systems Inc.). Initial conditions $\zeta(0)$ in the set $Z_0 = \{\zeta_0 = -A^{-1}b u_0, u_0 = 20, 21, \dots, 45 \text{ Nm}\}$, corresponding to power-train equilibrium points for $u = u_0$, were considered along with a target value $u_R = 100 \text{ Nm}$. The performances of the proposed control are evaluated by comparing, for all $\zeta_0 \in Z_0$, the optimal solution $T_{opt}^{CT}(\zeta_0)$ to the relaxed Problem 2 and the time $T^{M_{4cyl}}(\zeta_0)$ needed to steer ζ_0 to C_c in model M_{4cyl} under the proposed control. In fact, for the optimal solution $T^{M_{4cyl}}(\zeta_0)$ to Problem 1, we have

$$T_{opt}^{CT}(\zeta_0) \leq T_{opt}^{M_{4cyl}}(\zeta_0) \leq T^{M_{4cyl}}(\zeta_0).$$

As expected for larger u_0 , i.e. higher crankshaft speeds, $T^{M_{4cyl}}(\zeta_0)$ is closer to $T_{opt}^{CT}(\zeta_0)$ and, hence, to $T_{opt}^{M_{4cyl}}(\zeta_0)$ (see Figure 4).

6 Conclusions and future work

In this paper, we presented a novel approach to engine control for the Fast Positive Force Tracking problem, based on a hybrid model of the torque generation and of the power-train dynamics in a four-stroke engine. A control problem on this hybrid system is defined and solved using a sequence of approximations. The properties of the control law so obtained have been characterized, thus offering better confidence on the quality of the results with respect to commonly used heuristic approaches. In addition, since the control law is closed loop, expensive tuning processes can be avoided yielding a commercially appealing solution.

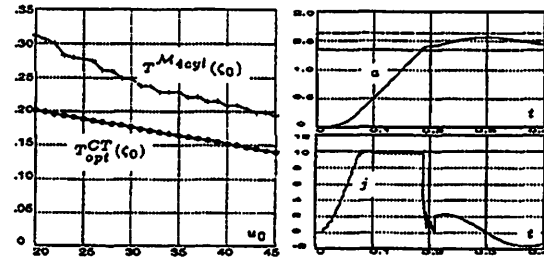


Figure 4: Relaxed optimal solution $T_{opt}^{CT}(\zeta_0)$ and hybrid solution $T^{M_{4cyl}}(\zeta_0)$ for $\zeta_0 \in Z_0$, vs. u_0 . (left); acceleration and jerk for $u_0 = 40 \text{ Nm}$ (right).

References

- [1] A. BALLUCHI, M. D. BENEDETTO, C. PINELLO, C. ROSSI, AND A. SANGIOVANNI-VINCENTELLI, *Cut-off in engine control: a hybrid system approach*, in 36th CDC, San Diego, CA, 1997.
- [2] —, *Hybrid control for automotive engine management: The cut-off case.*, in Hybrid Systems: Computation and Control, vol. 1386 of LNCS, Springer-Verlag, London, 1998.
- [3] L. BENVENUTI, M. D. BENEDETTO, C. ROSSI, AND A. SANGIOVANNI-VINCENTELLI, *Injector characteristics estimation for spark ignition engines*, in this proceedings, 1998.
- [4] Y. CHEN, G. LEITMANN, AND Z. XIONG, *Robust control design for interconnected systems with time-varying uncertainties*, Inter. J. Contr., 54, pp. 1119–1142.
- [5] A. S. MORSE, ed., *Control using logic-based switching*, vol. 222 of LNCIS, Springer-Verlag, London, U.K., 1997.
- [6] L. S. PONTRYAGIN, V. G. BOLTYANSKY, R. V. GAMKRELIDZE, AND E. F. MISCENKO, *The Mathematical Theory of Optimal Processes*, Wiley, New York, N.Y., 1962.
- [7] H. H. ROSENBROCK, *Computer-Aided Control System Design*, Academic Press, London, U.K., 1974.
- [8] V. UTKIN, *Variable structure systems with sliding modes: a survey*, IEEE Trans. on Aut. Contr., 22 (1977), pp. 212–222.

Controller Synthesis for Vision Guided Landing of a UAV

EECS 291e Course Project, Fall 1999

Omid Shakernia
omids@eecs.berkeley.edu

1 Introduction

A hybrid systems formulation of the problem of vision guided landing of an Unmanned Aerial Vehicle (UAV) is proposed. An unmanned aerial vehicle uses computer vision as a sensor in the feedback loop in an attempt to land onto a moving target. Vision guided landing is naturally posed as a controller synthesis problem of finding the least restrictive controller to keep the landing target within the field of view of the camera. The structure of the problem suggests a possible alternative to solving the Hamilton-Jacobi PDE in the controller synthesis procedure.

The outline of this report is as follows. We first give the motivating example of why we want to do controller synthesis. Then we review the controller synthesis formulation. Then we discuss how the controller synthesis formulation can be simplified for our class of systems. The analysis suggests a promising direction for future research of a class of systems where the solution to the Hamilton-Jacobi PDE is not necessary.

2 Problem Formulation

Our goal is to use computer vision based controller to land an Unmanned Aerial Vehicle onto a landing target. A camera on-board the UAV is used to measure image correspondences and optical flow of the landing target. These measurements are used in a “structure from motion” algorithm which recovers the UAV relative position and velocity relative to the landing target. The camera has limited field of view, thus maintaining view of target imposes constraint set of configurations where the landing target is visible.

Given a model of the UAV dynamics and the on-board camera, our goal is to compute the maximal controlled invariant subset of the visibility region of the UAV and find the least restrictive controller that maintains visibility of the landing target. The maximal controlled invariant set is important because it allows us to determine from which configurations it is feasible to initiate the landing maneuver. The least restrictive controller is important because it can be composed with other controllers which can try to meet other design considerations such as efficiency with respect to fuel, time, etc.

The vision sensor imposes a natural safety constraint, which transforms the control design problem of a purely continuous system into an inherent hybrid systems problem. Most visual servoing controllers ignore the visibility constraint and assume that if the initial conditions are “small

enough,” then visibility maintained throughout a trajectory generated by the controller. This assumption may be valid for a kinematic model of a system such as a 6DOF robot manipulator arm, but for a dynamic model of a UAV with actuator saturation constraints, this is a dangerous assumption, since the UAV will not be able to change directions arbitrarily fast.

We give a simple example of why controller synthesis is necessary for landing. In this simulation shown in figure 1, a simple double integrator model of the UAV is assumed, and a standard PD controller is applied for the purpose of landing. The inverted triangle with vertex at the origin corresponds to the visibility region of the UAV: when the UAV is within the region, the landing target is visible from the on board camera. This simulation on the left shows the trajectory generated by the PD controller when the inputs saturated, and the simulation on the right shows the trajectory under unbounded inputs. This simulation gives a clear example of why we need to do controller synthesis.

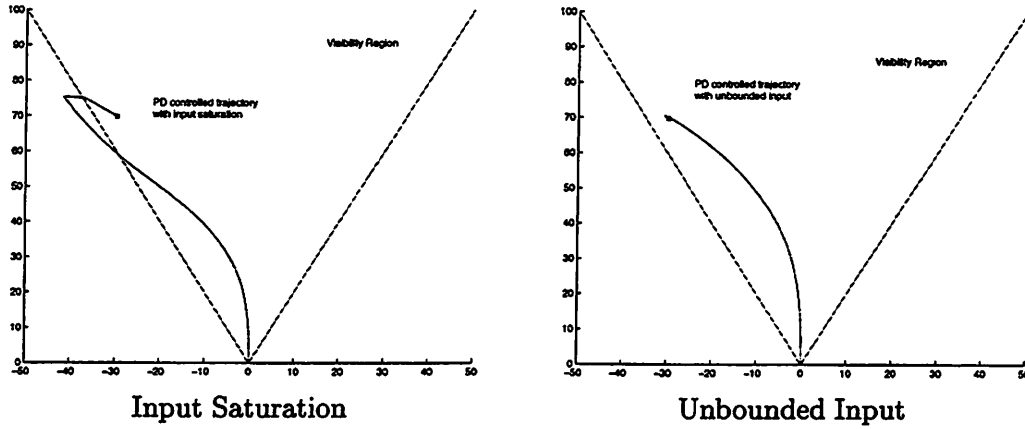


Figure 1: Landing with a PD controller at the same initial conditions with and without input saturation

3 Controller synthesis background

In this section, we briefly review the hybrid systems notation and controller synthesis formulation as given in [4].

3.1 Basic Notation

Consider a Hybrid Automaton $H = (X, U \cup D, I, f, E)$ described by

- State space: X (continous & discrete states)
- Input spaces: U controls, D disturbances
- Initial states: $I \subset X$
- Continous evolution: f vector field
- Discrete transitions: E transition relation

We will use the following notation, which is consistent with [4].

- $\chi = (\tau, x, (u, d))$ is an execution of H
- \mathcal{H} is the set of all executions
- A feedback controller is a mapping $g : \mathbf{X} \rightarrow 2^U$
- H_g is the closed loop automaton
- \mathcal{H}_g is set of closed loop executions

Given $F \subset \mathbf{X}$ define an *invariance property* $\Box F$ by

$$\Box F(\chi) = \begin{cases} \text{True} & \text{if } \forall t \in \tau \ x(t) \in F \\ \text{False} & \text{otherwise} \end{cases}$$

F is typically the “safety set” that the hybrid automaton tries to maintain. A feedback controller g solves the controller synthesis problem $(H, \Box F)$ if and only if $\Box F(\chi) = \text{True} \ \forall \chi \in \mathcal{H}_g$. A set $W \subset \mathbf{X}$ is *controlled invariant* if $\exists g$ that solves $(H, \Box W)$ when $I = W$. Feedback controllers that solve the synthesis problem can be partially ordered by the relation:

$$g_1 \preceq g_2 \iff g_1(x) \subseteq g_2(x) \ \forall x \in \mathbf{X}$$

A controller that solves $(H, \Box F)$ is *least restrictive* if it is a maximal element in this partial order. The controller synthesis problem can be stated as follows: given a safety specification $\Box F$, compute the maximal controlled invariant $W^* \subseteq F$, and the least restrictive controller that renders W^* invariant.

3.2 Game Theoretic Approach

A game theoretic approach is applied towards the controller synthesis problem. The controller plays a game against the disturbance in order to find the maximal controlled invariant subset of the safety set. It is assumed that the disturbance d does it’s best to drive the state of the system outside the safety set F .

For $F \subset \mathbf{X}$ define a cost $J : \mathcal{H} \rightarrow \mathbb{R}$

$$J(\chi) = \begin{cases} 1 & \text{if } \forall t \in \tau, \ x(t) \in F \\ 0 & \text{otherwise} \end{cases}$$

The control and disturbance play zero sum game over J . The controller “wins” if maintains safety specification, *i.e.* keeps $J = 1$, and the disturbance plays optimally against controller to try to drive $J = 0$. Define max-min solution $J^* : \mathbf{X} \rightarrow \mathbb{R}$ by

$$J^*(x^0) = \max_g \min_d \left(\min_{\chi=(\tau, x, (u, d)) \in \mathcal{H}_g} J(\chi) \right)$$

Define set $W^* \subseteq F$ by

$$W^* = \{x^0 \in F : J^*(x^0) = 1\}$$

A proposition proven in [4] shows that W^* is the unique maximal controlled invariant subset of F .

Our interest for vision guided landing is the controller synthesis for a purely dynamical system. When H is a purely dynamical system, we have a *pursuit-evasion* game [1] between the control and disturbance. The controller is “captured” by the disturbance if it is driven out of the safety set. In this case, the hybrid system is simplified to

$$\dot{x} = f(x, u, d), \quad x \in \mathbf{X}, \quad (u, d) \in \mathbf{U} \times \mathbf{D}$$

Assume the safe set defined by $F = \{x \in \mathbf{X} : l(x) \geq 0\}$ where $l : \mathbb{R}^n \rightarrow \mathbb{R}$ differentiable with $\frac{\partial l}{\partial x}(x) \neq 0$ on ∂F . Introduce a *value function*

$$\begin{aligned} J(x^0, u, d, t) &: \mathbb{R}^n \times \mathbf{U} \times \mathbf{D} \times \mathbb{R} \rightarrow \mathbb{R} \\ J(x^0, u, d, 0) &= l(x^0) \end{aligned}$$

Define the max-min solution

$$J^*(x, t) = \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} J(x, u(\cdot), d(\cdot), t)$$

Notice that the approach is conservative in that the advantage is always given to the disturbance. Now introduce the Hamiltonian

$$\begin{aligned} H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbf{U} \times \mathbf{D} &\rightarrow \mathbb{R} \\ H(x, p, u, d) &= p^T f(x, u, d) \end{aligned}$$

Then the optimal Hamiltonian is given by

$$H^*(x, p) = \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} H(x, p, u, d)$$

If the optimal value function is $J^*(x, t)$ is differentiable, we may solve the (modified) *Hamilton-Jacobi* PDE:

$$\begin{aligned} J^*(x, 0) &= l(x) \\ -\frac{\partial J^*(x, t)}{\partial t} &= \min \left\{ 0, H^* \left(x, \frac{\partial J^*(x, t)}{\partial x} \right)^T \right\} \end{aligned}$$

If $J^*(x, t)$ converges as $t \rightarrow -\infty$ to a unique $J^*(x)$, may define $W^* = \{x \in F : J^*(x) \geq 0\}$. W^* is maximal controlled invariant subset of F . The least restrictive controller that renders W^* invariant is given by

$$g(x) = \begin{cases} \{u \in \mathbf{U} : \min_{d \in \mathbf{D}} \frac{\partial J^*(x)}{\partial x} f(x, u, d) \geq 0\} & \text{if } x \in \partial W^* \\ \mathbf{U} & \text{if } x \in (W^*)^\circ \cup (W^*)^c \end{cases}$$

In general, finding the of maximal controlled invariant subset and least restrictive controller for a given system and safety set are very challenging due to the computational issues associated with solving the Hamilton-Jacobi PDE. In particular when the PDE has a non-smooth right hand side, the solution may exhibit discontinuities or shocks. Also, numerical techniques which rely on discretizing the state space become intractable in \mathbb{R}^n for $n > 3$.

4 Synthesis for landing

Consider a (very) simplified dynamic model of an Unmanned Aerial Vehicle:

$$\dot{x} = Ax + Bu, \quad A = \begin{bmatrix} 0 & I_{3 \times 3} \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ I_{3 \times 3} \end{bmatrix} \quad (1)$$

where $\mathbf{X} = \mathbb{R}^6$ and the input space $u \in \mathbf{U} \subset \mathbb{R}^3$ is a compact rectangle. This model is a simplified version of a valid model for the feedback linearized closed loop dynamics of a helicopter [3]. Before considering this case, we look at an even simpler model in order to build a feel for how control synthesis will work for linear systems of this class.

4.1 Vertical lander

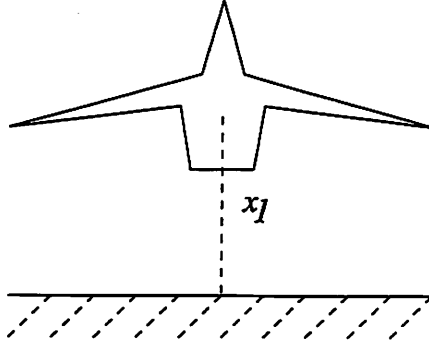


Figure 2: Vertical Lander

The “vertical lander” is a version of the above problem when UAV is constrained to only move vertically. Here, maintaining visibility of the landing pad is equivalent to not going below ground. The state is given by $x = (x_1, x_2)^T \in \mathbf{X} = \mathbb{R}^2$ and the input is $u \in \mathbf{U} = [-1, 1]$. The state equations $\dot{x} = Ax + Bu$ become

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

The safe set is given by $F \triangleq \{x \in \mathbb{R}^2 : l(x) = x_1 \geq 0\}$. Notice that since the input is bounded, the UAV can not change directions arbitrarily fast. Thus if $x \in \partial F$, then x will remain in F if and only if $\dot{l}(x) = \frac{\partial l}{\partial x}(Ax + Bu) \geq 0$.

Suppose we are given a system governed by the vector field $\dot{x} = f(x) + g(x)u$, where $f(x) = Ax$ and $g(x) = B$, and a value function $y = h(x)$. Further, suppose the system has relative degree is γ , that is $L_g L_f^i h(x) \equiv 0$ for $i = 0, \dots, \gamma - 2$, and $L_g L_f^{\gamma-1} h(x) \neq 0$. The Maximum Principle [2] states that the optimal control u^* to maximize the value function h is just:

$$u^* = \arg \max_{u \in \mathbf{U}} \left\{ L_f^\gamma h(x) + L_g L_f^{\gamma-1} h(x) u \right\}$$

For the case of linear systems, this means that the optimal control will always be switching among points on the boundary of the input set \mathbf{U} . This is known as “bang-bang” control.

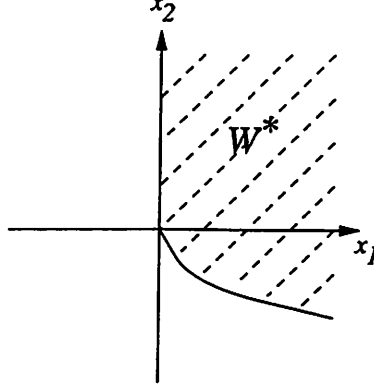


Figure 3: Maximal controlled invariant set for vertical lander

Applying the Maximum Principle to the vertical lander the optimal control for maximizing the $l(x)$ is $u^* \equiv 1$. This is the best the controller can do to maintain visibility of the landing target. With this optimal control and the method described in section 3 for computing the least restrictive controller is, it is easy to show the following:

Proposition 1 *For the vertical lander, the maximal controlled invariant subset of $F = \{x \in \mathbb{R}^2 : l(x) = x_1 \geq 0\}$ is given by:*

$$W^* = \left\{ x = (x_1, x_2)^T \in X : (l(x) \geq 0 \wedge \dot{l}(x) \geq 0) \vee x_1 \geq \frac{1}{2}x_2^2 \right\}$$

The least restrictive controller that renders W^ invariant is given by:*

$$g(x) = \begin{cases} [0, 1] & \text{if } l(x) = \dot{l}(x) = 0 \\ 1 & \text{if } x_1 = \frac{1}{2}x_2^2 \\ U & \text{otherwise} \end{cases}$$

Now we give an alternative method of computing W^* . According to the controller synthesis method of section 3, the maximal controlled invariant $W^* \subseteq F$ can be written as:

$$W^* = \{x_0 \in F : \exists u(\cdot) \in \mathcal{U} \forall d(\cdot) \in \mathcal{D} \ x(t) \in F \ \forall t \in \tau\}$$

For the vertical lander, we can find W^* by applying the optimal control $u^* = 1$, and notice that we get a polynomial flow

$$\begin{aligned} x_0 &= (x_1(0), x_2(0)) \in X \\ x_1(t) &= x_1(0) + x_2(0)t + \frac{1}{2}t^2 \\ x_2(t) &= x_2(0) + t \end{aligned}$$

Thus, we may define first a order formula in $\text{OF}(\mathbb{R})$

$$\psi(x_1, x_2) = \forall t : t \geq 0 \wedge 2x_1 + 2x_2t + t^2 \geq 0$$

and simply write $W^* \subseteq F$ in terms of ψ :

$$W^* = \{(x_1, x_2) \in F : \psi(x_1, x_2)\}$$

4.2 3D Lander

For the 3D lander described in equation (1), consider the input space $\mathbf{U} = [-1, 1] \times [-1, 1] \times [-1, 1]$. Suppose the visibility region is an inverted pyramid at origin of Euclidean space (the landing target) described by

$$F = \{x \in \mathbb{R}^6 : \bigwedge_{i=1}^4 l_i(x) \geq 0\}$$

$$l_1(x) = x_1 + x_3, \quad l_2(x) = -x_1 + x_3,$$

$$l_3(x) = x_2 + x_3, \quad l_4(x) = -x_2 + x_3,$$

Notice that for this case, a controller that ensures $l_1(x) \geq 0$ will not constrain u_2 . Also, for each boundary, the optimal control for u_3 is always $u_3^* = 1$. Thus controllers for u_1, u_2 can be decoupled and we can study planar lander.

4.3 Planar Lander

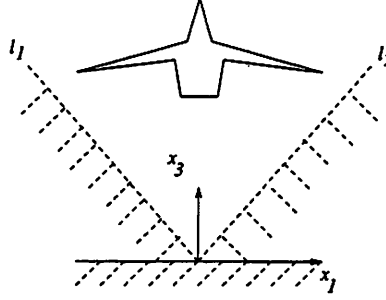


Figure 4: Planar Lander

Consider the vertical and horizontal components of the UAV:

$$x = (x_1, x_3, v_1, v_3) \in \mathbf{X} = \mathbb{R}^4$$

$$u = (u_1, u_3) \in \mathbf{U} = [-1, 1] \times [-1, 1]$$

The state equations are $\dot{x}_i = v_i, \dot{v}_i = u_i$ for $i = 1, 3$. Let the visibility region be given by $F = F_1 \cap F_2$, with $F_i = \{x \in \mathbf{X} : l_i(x) \geq 0\}$, $l_1(x) = x_1 + x_3$, $l_2(x) = -x_1 + x_3$. We can divide the visibility region two boundaries and solve for the maximal controlled invariant subset of each boundary as we did for the vertical lander. Define

$$W_i^* = \{x \in F_i : \exists u(\cdot) \in \mathbf{U} \ x(t) \in F \ \forall t \geq 0\}$$

By the Maximum Principle, for W_1^* , “ $\exists u$ ” becomes $(u_1, u_3) = (1, 1)^T$, and for W_2^* , “ $\exists u$ ” becomes $(u_1, u_3) = (-1, 1)^T$. Then we may compute $W_i^* = \{x \in F_i : \psi_i(x)\}$, where

$$\psi_1(x) = \forall t : t \geq 0 \wedge (x_3 + x_1) + (v_3 + v_1)t + t^2 \geq 0$$

$$\psi_2(x) = \forall t : t \geq 0 \wedge (x_3 - x_1) + (v_3 - v_1)t + t^2 \geq 0$$

Analogously to the vertical lander, we may compute the maximal controlled invariant subsets of F_i to be:

$$W_1^* = \{x \in F_1 : (l_1(x) \geq 0 \wedge \dot{l}_1(x) \geq 0) \vee (x_3 + x_1) \geq \frac{1}{4}(v_3 + v_1)^2\}$$

$$W_2^* = \{x \in F_2 : (l_2(x) \geq 0 \wedge \dot{l}_2(x) \geq 0) \vee (x_3 - x_1) \geq \frac{1}{4}(v_3 - v_1)^2\}$$

and the least restrictive controllers for W_i^* to be:

$$g_1(x) = \begin{cases} [0, 1] \times [0, 1] & \text{if } l_1(x) = \dot{l}_1(x) = 0 \\ (1, 1)^T & \text{if } (x_3 + x_1) = \frac{1}{4}(v_3 + v_1)^2 \\ \mathbf{U} & \text{otherwise} \end{cases}$$

$$g_2(x) = \begin{cases} [-1, 0] \times [0, 1] & \text{if } l_2(x) = \dot{l}_2(x) = 0 \\ (-1, 1)^T & \text{if } (x_3 - x_1) = \frac{1}{4}(v_3 - v_1)^2 \\ \mathbf{U} & \text{otherwise} \end{cases}$$

In general, if a safe set is given as an intersection of basic safe sets, the maximal controlled invariant subset of the safe set will be a strict subset of the intersection of the maximal controlled invariant subsets of the base sets.

Notice that there is a set $Y \subset \partial W_1^* \cap \partial W_2^*$ given by $Y = \{x \in W_1^* \cap W_2^* : g_1(x) \cap g_2(x) = \emptyset\}$ where the allowable controls conflict. That is, for all states in Y , there exist a control to keep the state inside W_1^* and a control to keep the state inside W_2^* , but no control to keep the state inside both W_1^* and W_2^* . Thus it must be that $Y \cap W^* = \emptyset$, and $W^* \subset W_1^* \cap W_2^* \subset F$.

Now, safety set is $W_1^* \cap W_2^* \setminus Y$, and we can iterate the procedure. Notice that the safety set is still semialgebraic. Since the optimal controls lead to polynomial flows, all sets will be definable in $\text{OF}(\mathcal{R})$. In general, this procedure may not terminate, but it suggests a possible semidecidability result.

5 Directions for Future Work

The example described in this report suggests a possible semidecidable controller synthesis method. In future work, we will attempt to use the Maximum Principle and bang-bang controls to link the controller synthesis method of Lygeros *et al* [4] with results of Pappas [5]. Consider a dynamical system of the class

$$\dot{x} = Ax + Bu + Ed$$

with $x \in \mathbf{X} = \mathbb{R}^n$, and $u \in \mathbf{U} \subset \mathbb{R}^{n_u}$, $d \in \mathbf{D} \subset \mathbb{R}^{n_d}$ where \mathbf{U}, \mathbf{D} are compact rectangles and $A \in \mathbb{Q}^{n \times n}$ is nilpotent and $B \in \mathbb{Q}^{n \times n_u}$, $E \in \mathbb{Q}^{n \times n_d}$. Consider a safety set $F = \{x \in \mathbf{X} : l(x) \geq 0\}$ where $l(x)$ is a polynomial.

The controls and disturbances play a game over the cost $l(x)$. Suppose the relative degree between both the controls and disturbance to the cost is γ . That is, $l^\gamma(x)$ is the first Lie derivative of $l(\cdot)$ along the vector field that is a function of u and d . The optimal controls and disturbances can be found as:

$$u^* = \arg \max_{u \in \mathbf{U}} \min_{d \in \mathbf{D}} \{l^\gamma(x)\}$$

$$d^* = \arg \min_{d \in \mathbf{D}} \max_{u \in \mathbf{U}} \{l^\gamma(x)\}$$

In general, this may not be a saddle solution. By the Maximum Principle, the optimal controls and disturbances will be piecewise constant on $\partial\mathbf{U}, \partial\mathbf{D}$. They will also be feedback solutions. From these optimal controls we may construct a hybrid system with $2^{n_u} \cdot 2^{n_d}$ discrete states, with guards and invariants governed by the optimal controls, and the identity as the reset map. Within each state q_i , the controls and disturbances $u_{q_i}^*, d_{q_i}^*$ are constant. This implies that within each discrete state the flow is given by

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}d\tau(Bu_{q_i}^* + Ed_{q_i}^*)$$

Note that since A is nilpotent we have

$$e^{At} = \sum_{k=0}^{n-1} \frac{t^k}{k!} A^k$$

This implies that within each discrete state, the flow is *polynomial*. Recall that with a polynomial flow, if a given set Y is definable in $\text{OF}(\mathcal{R})$ then $\text{Pre}_\tau(Y)$ is definable in $\text{OF}(\mathcal{R})$. Thus, by quantifier elimination, one can analytically compute the reach sets of each discrete state. Thus, the problem of finding the maximal controlled invariant subset of a safe set can be converted to a reach set computation that can be solved analytically. This suggests a possible semidecidability results for this class of linear systems, which we will pursue in further research.

References

- [1] T. Başar and G.J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, 2nd edition edition, 1995.
- [2] A.E. Bryson and Y. Ho. *Applied Optimal Control*. Hemisphere Publishing Corporation, 1975.
- [3] T.J. Koo and S.S. Sastry. Differential flatness based full authority helicopter control design. In *Submitted to 1999 Conference on Decision and Control*, 1999.
- [4] J. Lygeros, C. Tomlin, and S.S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 1999.
- [5] G.J. Pappas. *Hybrid Systems: Computation and Abstraction*. PhD thesis, UC Berkeley, 1998.

Implementation of hybrid phenomena in the HCC simulation language

Adam Sweet
EE 291E
May 17, 1999

Introduction

Traditionally, systems have been modeled as either continuous systems, described by continuous differential equations, or discrete systems, described by a set of states and the set of transitions between the states. However, much recent interest has developed for hybrid systems, which combine elements of both the traditional continuous systems and discrete systems. As hybrid systems become a better-developed area of research, there is an increasing interest in developing tools to accurately simulate the hybrid systems. Hybrid systems have several unique characteristics that make it difficult for standard tools to accurately simulate the systems. This has led to the creation of many new hybrid simulators, each with its own capabilities and ability to model these characteristics of hybrid systems. This report will give a description of the characteristics of hybrid systems and discuss their implementation into one particular simulation program, HCC.

Development of HCC

HCC was developed out of concurrent constraint programming; the name HCC stands for Hybrid Concurrent Constraint. Basically, HCC took concurrent constraint programming and introduced differential equations to extend it across time. The basic flow of HCC is given below in Figure 1. HCC begins by reading in a text file created by the user. It then parses the constraints and differential equations given there into memory. The execution of the simulation progresses as a series of point phases and interval phases. It begins with a point phase, where the initial conditions and constraints are solved at an instant in time. Next, HCC begins an interval phase, during which time progresses. The differential equations given in the file are integrated, using one of

several numerical integration methods. Currently, the available methods are Euler, 4th order Runge-Kutta, and 4th-5th order Runge-Kutta with adaptive stepsize.

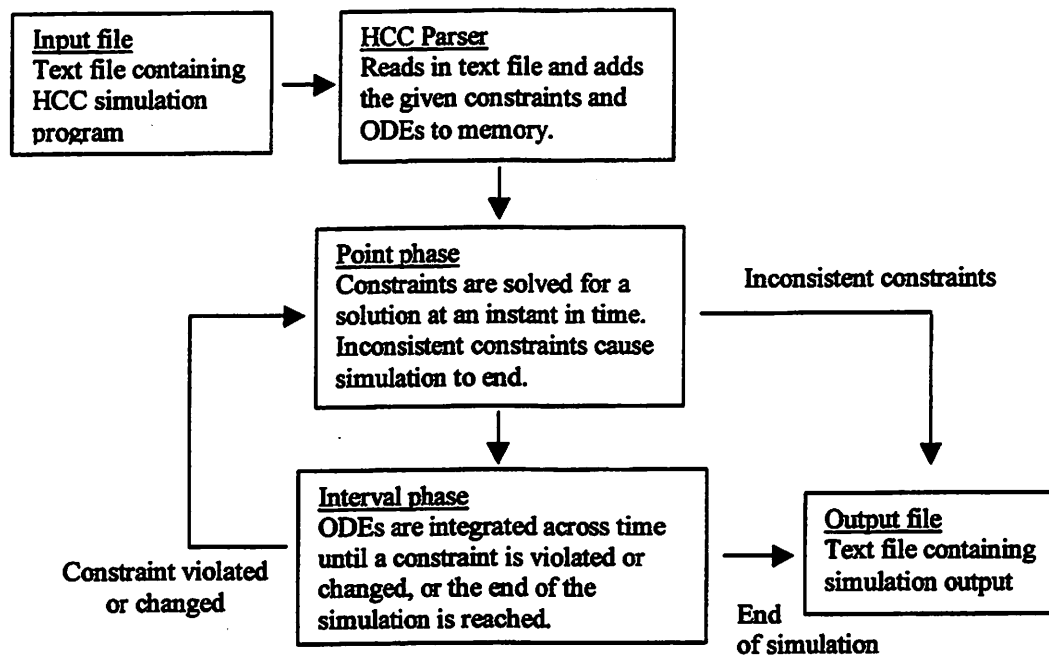


Figure 1: Basic flow of HCC

The simulation remains in the interval phase until a constraint is violated or changed. This can happen as a result of either the variables reaching values that directly conflict with given constraints, or by the variables reaching values that trigger the simulation to change the equations and constraints. The program then reenters a point phase and attempts to solve the new constraints for a solution. This process is repeated until either a valid solution cannot be found, or the end of the simulation is reached.

Characteristics of Hybrid Systems

There are several general characteristics unique to hybrid systems that must be represented in any hybrid simulator¹. They are:

1) Time events

An event could be generated within the simulation by time crossing a threshold value. This event needs to be generated by the simulation at the correct time.

2) State events

- Detection

If a condition for generating an event depends on the system entering a particular state, the simulation should detect when the system has crossed into that state.

- Location

If a state event is detected, the simulation also should detect the time at which that event occurred.

3) Reinitialization

The simulation should have an ability to reinitialize the state of the system when an event is generated. There are two ways to do this:

- Explicit

Explicit reinitialization is used when the user specifies the reinitialization function explicitly.

- Integration of balance laws

Reinitialization of the state of the system could also be done by automatically integrating a balance law from physics, such as conservation of momentum.

4) Dirac pulses

Dirac pulses are most commonly used to represent collisions in systems, but could be used in other circumstances as well.

5) Event iteration

¹ Source: Mosterman, Pieter J. "An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages", HSCC '99, LNCS 1569, pp. 165-177, 1999.

It is possible for events in the simulation to instantly generate additional events. There are two possibilities for updating the state vector when this happens:

- **Update**

In an updating mode, the state vector is changed by each event as the events propagate.

- **Invariant**

In an invariant mode, the state vector is held constant until events stop.

6) Simulation model events

- **Add/remove program blocks**

Some events can require that blocks of the simulation be added or removed. This ability should be reflected in the simulation.

- **Resort equations**

When the blocks are added or removed, it can often change the computational dependencies of the equations, and the equations must be resorted.

- **Resolve equations**

Also, when the equations are changed, it can result in a larger set of equations that will require index reduction to be dealt with effectively.

7) Chattering

If the differential equations for two separate states of a hybrid are both driving the system back to the boundary between states, chattering between the states will occur. This will at least slow down a simulation.

Implementation into HCC

Many but not all of the characteristics listed above are implemented into HCC. The following describes the method of implementation of the characteristics.

Time events:

Time events are dealt with by simply ending an interval phase and beginning a new point phase when the event occurs. If the simulation is using adaptive stepsize integration, the integration is simply halted, as shown in Figure 2:

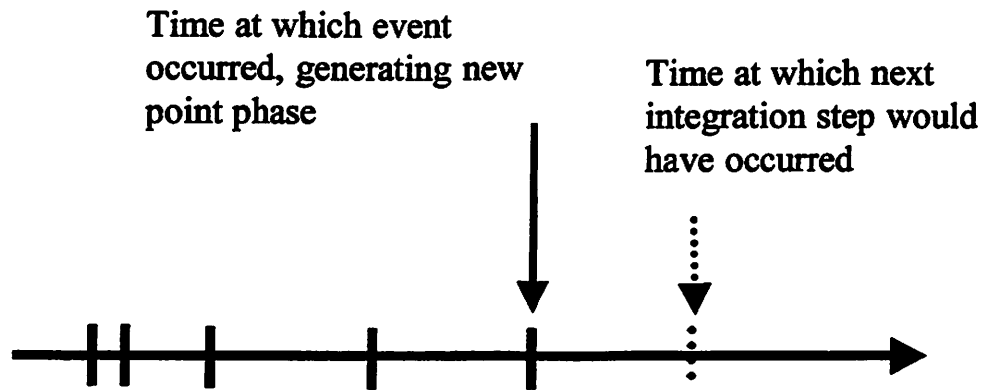


Figure 2: How time events are handled in HCC

Including time events in HCC is usually accomplished with the command, *wait x do A*.

State events:

As all statements are represented in HCC as constraints, state events are generated by these events being violated or changed. HCC will detect these events, and it will backtrack to detect when the event occurred (to the best of its resolution). The time at which the constraint was violated is found by finding the zeros of a cubic spline interpolation to the constraint function. The process is iterated until the time location of the event is found. This is depicted in Figure 3:

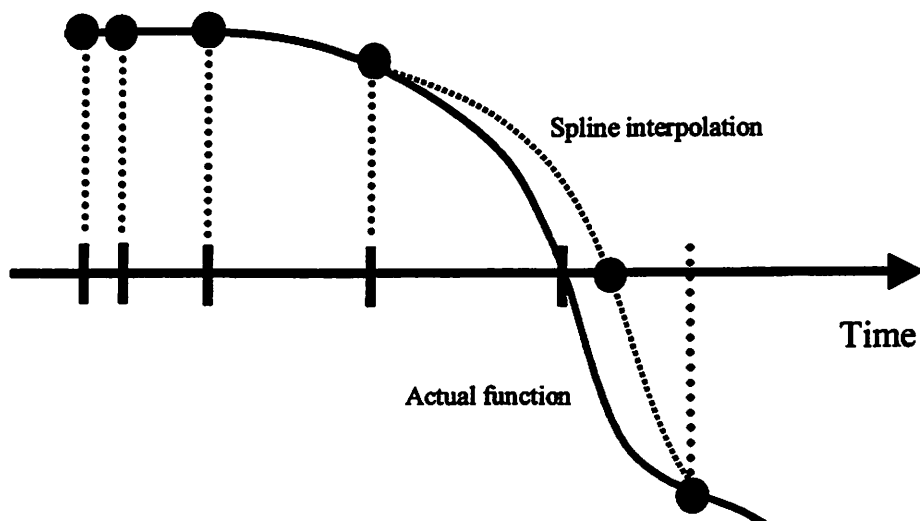


Figure 3: How state events are detected and localized in HCC

Reinitialization:

HCC only supports explicit reinitialization of the state variables. Automatic integration of balance laws is not supported. When collisions occur, the laws of conservation of momentum and energy must be explicitly stated in the model formulation. An example of reinitialization is the Newton's cradle, shown below in Figure 4:

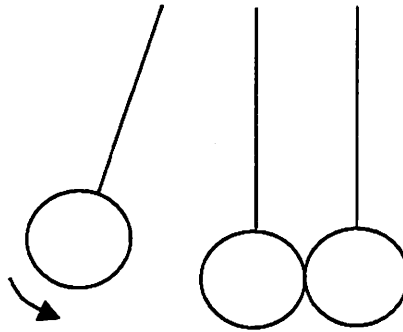


Figure 4: Newton's cradle

After the ball on the left is released, it swings down and impacts on the middle ball, generating a state event. This event is detected, and the state vector is reinitialized so that the momentum of the first ball is transferred to the second. Again, this reset of the state must be specified explicitly in the simulation.

Dirac pulses:

Dirac pulses are not currently implemented in HCC. The reason is that Dirac pulses are not yet mathematically rigorous objects. They are commonly used in engineering to represent collisions, where a significant change of momentum occurs in an insignificant amount of time. Thus, they could be used in simulating the Newton's cradle example of Figure 4.

Event iteration:

HCC supports state updating event iteration, not state invariant event iteration. Again, the Newton's cradle is a good example of the state updating event iteration. After

the first ball has swung down and collided with the second ball, it generated a state event, and the state of the system was updated to transfer the momentum to the second ball. However, as the second ball was touching the third ball, it will instantly collide and transfer its momentum to the third ball. HCC implements this by using the construct *next()*, which introduces another point phase immediately following the current point phase. When no more discrete events are needed, then HCC will finally begin the next interval phase.

The HCC output of the Newton's cradle example is given below in Figure 5. Note that it gives the expected results: Ball 1 is initially pulled away and then released. It travels down to collide with Ball 2. Ball 2 instantly collides with Ball 3, transferring the momentum to it. Therefore, Ball 2 does not move, and Ball 3 then begins to move continuously. The model does not include damping, so the process will be repeated forever.

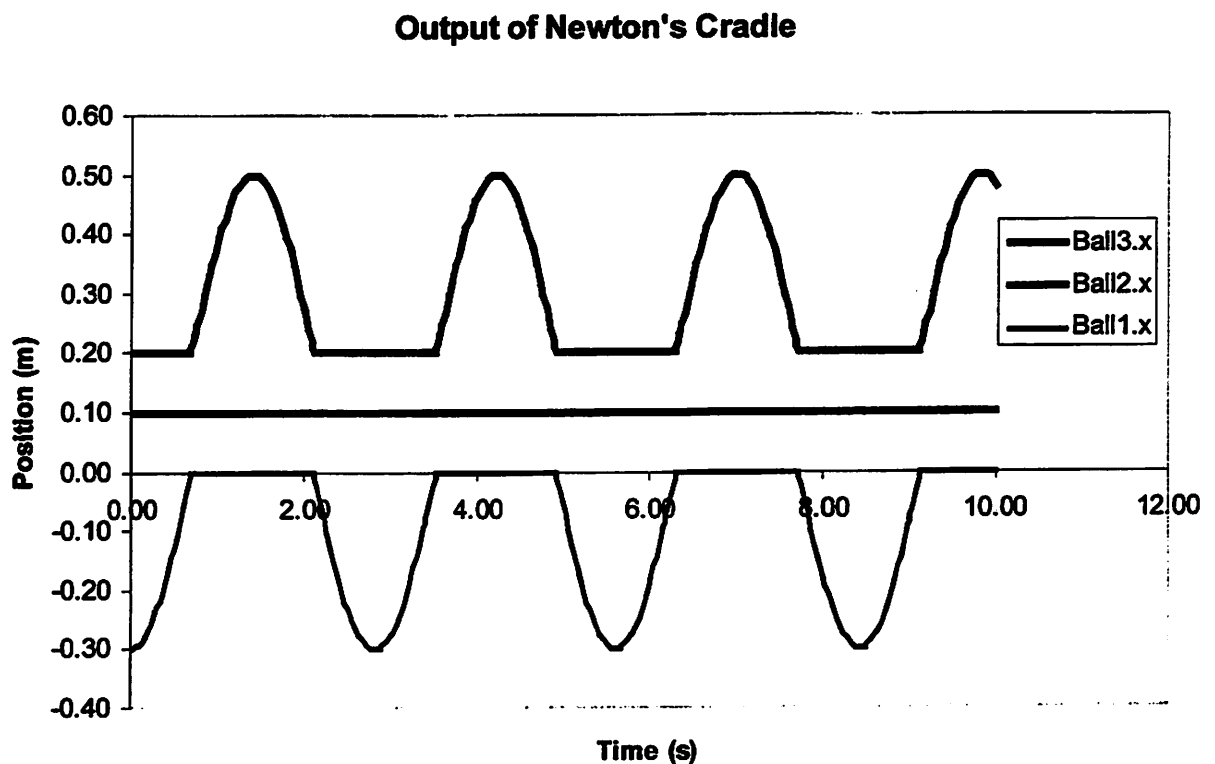


Figure 5: HCC results of the Newton's Cradle example

HCC does not in general support state invariant event iteration. It will not determine automatically whether the state vector should be held invariant or updated as the events progress. Special cases could be made explicitly, with the reinitialization for the event not occurring if there will be additional events generated. However, these of course must be considered on a case-by-case basis.

Simulation model events:

HCC does support dynamic addition and removal of program blocks. To illustrate this, consider the cat and mouse example plotted in Figure 6:

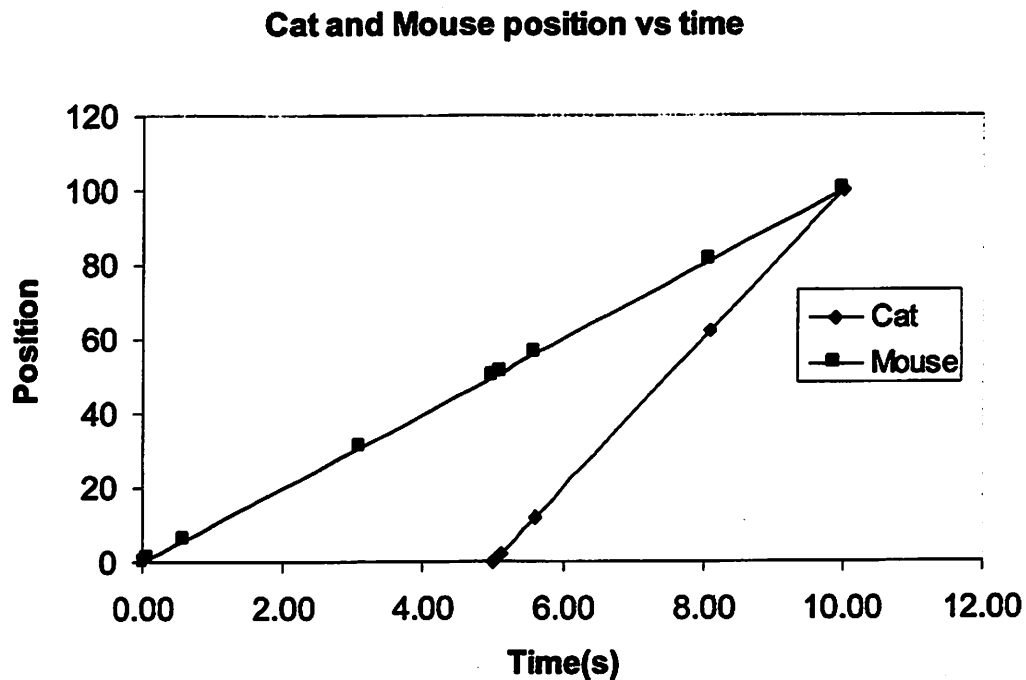


Figure 6: Cat and mouse example, showing the addition and removal of blocks

The simulation is begun with a mouse at position = 0. When the mouse reaches position 50, a cat object is created in the simulation. The cat chases after the mouse, and when it reaches the mouse, the mouse object is removed from the simulation. This is a simple example, but the same structure would be used for more complicated simulations, such as cars coming onto and leaving an automated highway system.

With the addition and removal of equations from a simulation, the computational dependencies within the equations might also change. HCC does support resorting of equations, as in every point phase the constraints are solved for a consistent solution. However, it does not perform index reduction on the new set of equations.

Chattering:

Chattering is not supported by HCC. If a system is chattering, the simulation can only progress if the simulator overshoots the time location of the state event by a small error. Then, the system will proceed in a zig-zag fashion along the boundary between the states. This process is much slower than if the simulation progressed directly along the boundary. However, HCC cannot simulate systems with chatter, because it attempts to find the exact time at which a state event was generated. In the best case, HCC will begin an infinite loop where the system is only generating successive point phases, and not progressing in time. In the worst case, HCC will crash. Either way, chattering of systems must be avoided in HCC.

HCC in comparison with other simulators:

A table with a comparison between HCC and many other existing hybrid simulation programs is listed in Table 1. As evident from the table, HCC has all of the characteristics of most hybrid simulation programs. There is one additional note about the table. This list of characteristics was first made in a paper from LDR Oberpfaffenhofen, from Wessling, Germany. Hybersim is the language that fulfills most of the characteristics listed here: it was developed at the same university. It could therefore be possible that the author of the original paper was biased when creating the list of desired characteristics. Many of the hybrid characteristics listed above are not strictly necessary to implement in a simulator. If a system has a characteristic not supported by a simulator, it could often be modeled in a slightly different way to take into account the capabilities of the hybrid simulator used.

Table 1: HCC in comparison to other hybrid simulation languages²:

		χ	A B A C U S	B a S i P	D O R s	D y m o l a	g P R O M S	H y b r i m	O m b r a	S H I F T	S I M U L I N K	2 O S I M	H C C
State events	Detection	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Location	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
Time events		✓	✓		✓	✓	✓		✓		✓	✓	✓
Simulation model	Add / remove	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
	Resort	✓	✓		✓	✓	✓	✓	✓				✓
	Resolve							✓					
Reinitialization	explicit	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	integration							✓					
Event iteration	invariant							✓					
	update	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓
Dirac pulses								✓					
chattering													

Additional advantages of HCC

There are characteristics of HCC that were not mentioned in the above discussion. The first is that HCC is a declarative language, as well as an object-oriented language. This gives the user a great amount of flexibility. Also, every variable in HCC is actually represented as an interval. This gives HCC the ability to do linear and nonlinear optimization problems, as well as modeling physical systems. The interval nature of HCC also makes it possible to do differential inclusions in the models. A simple differential inclusion was simulated to create Figure 7:

² Source: Mosterman, Pieter J. "An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages", HSCC '99, LNCS 1569, pp. 165-177, 1999.
HCC column added for current report.

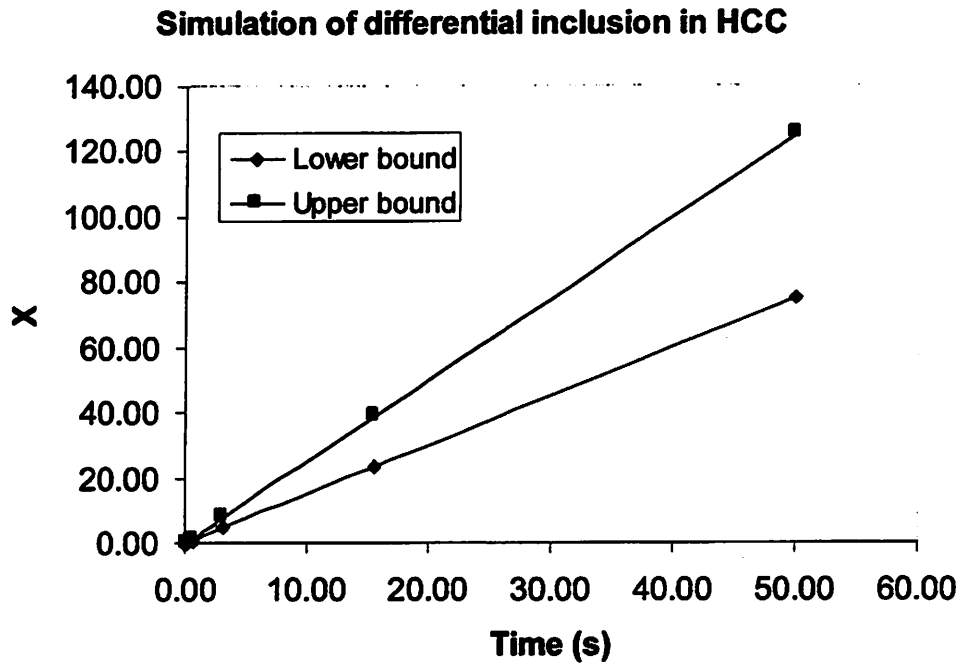


Figure 7: Results of differential inclusion $x' = [1.5, 2.5]$

The differential inclusion was a simple one. There are limitations on HCC's ability to produce differential inclusions. When the system has several nonlinear, coupled differential inclusions, the variable intervals become arbitrarily large. Also, care must be taken when using the same interval variables in constraints with non-interval variables.

Conclusion

HCC includes the ability to model most of the potential characteristics of hybrid systems. Many of the characteristics that are not implemented into HCC can be worked around by modifying the model of the system. While HCC is a relatively new hybrid system simulation language, it measures well against the most hybrid simulation systems available today. In addition, it includes some additional characteristics that are not included in most other simulation programs. All in all, HCC is a well-developed language to use in modeling hybrid systems.

```
// HCC code for Cat and Mouse example
```

```
always Mouse = (vel)[pos]{  
  pos = 0,  
  hence pos' = vel,  
  sample(pos)  
},
```

```
always Cat = (vel)[pos]{  
  pos = 0,  
  hence pos' = vel,  
  sample(pos)  
},
```

```
Arbiter = (C, M){  
  always if prev(C.pos) = prev(M.pos) then Wincat,  
  always if prev(M.pos) = 120 then Winmouse  
},
```

```
do {  
  Mouse(M, 10),  
  when (M.pos = 50) do Cat(C, 20)  
} watching (Wincat || Winmouse),  
Arbiter(C,M)
```

```

// HCC code for Newton's Cradle

#define      radius  0.05
#define SAMPLE_STYLE  1
#define ROUNDOFF 10

Ball = (xinit, xcenter_init, stiffness)
  [x, v, center, k, Change, Reset] {

  x = xinit,
  v = 0.0,
  StoreNumVal(center, xcenter_init),
  StoreNumVal(k, stiffness),

  always {

    unless Change then v' = k*(center - x),
    x' = v

  },

/*  Reset = (new_xvalue) {
    Change,
    x = new_xvalue,
    v = 0.0
  },
*/
  sample(x, v)
},

Collision = () {

  always forall Ball(A) do forall Ball(B) do
  if (A != B) then //not the same ball
  if ((A.x - B.x)^2 = 4*radius^2 ) then
  {
    if ( (A.x < B.x) && (prev(A.v) > prev(B.v)) ) then
    {
      A.Change, B.Change,
      A.v = prev(B.v),
      B.v = prev(A.v),
      forall Ball(C) do
      if (C != A && C != B) then {
        if ((A.x - C.x)^2 = 4*radius^2 &&
          (A.x > C.x) && (A.v < C.v)) then
          next Bounce,
        if ((B.x - C.x)^2 = 4*radius^2 &&
          (B.x < C.x) && (B.v > C.v)) then
          next Bounce
      }
    }
  }

},

Ball(Ball1, -0.3, 0.0, 5),
Ball(Ball2, 0.10, 0.10, 5),

```

```
Ball(Ball3, 0.20, 0.20, 5),  
Collision(),
```

```
always {  
  SetNumVal = (var, value) {  
    var = value,  
    do hence var = value  
    watching var != value  
  },  
  SetVal = (Var, Value) {  
    Var = Value,  
    do hence Var = Value  
    watching Var != Value  
  },  
  StoreNumVal = (var, value) {  
    evalexpr temp = value in {  
      realvar(temp),  
      SetNumVal(var, temp)  
    }  
  }  
}
```

FLIGHT ENVELOPE PROTECTION CONTROLLER SYNTHESIS USING FINITE ELEMENT METHOD EE291E CLASS PROJECT REPORT

BRUNO SINOPOLI AND T.JOHN KOO

1. INTRODUCTION

Aerial vehicles are able to perform complex maneuvers in the air by composition of elementary flight modes. Flight modes represent different modes of operation of the Aerial Vehicle and correspond to controlling different variables of the vehicle dynamics [5]. At a higher level composition of complex behaviors, ability to react to the information coming from the environment by changing behavior is essential to successfully achieve any kind of mission. The complex interaction between continuous dynamics and discrete logic can be modeled by using hierarchical structure within the hybrid systems framework. Flight Management Systems (FMS) were first proposed for smart air-crafts in future Air Traffic Management Systems (ATMS)[5] for decentralized air traffic control. FMS are responsible for

- planning of the flight path, generating a proper sequence of flight modes, calculating a feasible trajectory and regulating an Unmanned Aerial Vehicle (UAV) along the nominal trajectory.
- switching among different modes of operation to handle situations like conflict resolution among UAVs, obstacle avoidance, and flight envelope protection.

An FMS operates in a mission critical environment, where reliability and safety are more important criteria than performance. Conflict resolution, obstacle avoidance need to be guaranteed They have a higher priority than performance optimality criteria. From here comes the need for a supervisory control that can switch to a different operation mode to guarantee safety. Protecting the flight envelope is one of the important flight mode needs to be switched from one to another in order to cope with the situation.

Using the Hybrid System formalism, safety properties can be easily described and a procedure for synthesis of a safety controller which makes use of optimal control techniques can be derived [1].

The aim of this project is to protect the flight envelope for an helicopter in forward flight. Previous work has been conducted by Lygeros et al. [1]. A flight envelope control for aircraft control has been synthesized using optimal control techniques. In particular the problem for a simple two dimensional system of the aircraft has been solved by using Hamilton-Jacobi partial differential equation and solving it analytically. Aside from particular cases an analytical solution cannot be found. In this project we will also address this problem by proposing the use of Finite Element Theory for computing a numerical solution to the Hamilton-Jacobi equation.

Section 2 will formulate the problem of flight envelope controller synthesis for an helicopter. Section 3 will introduce the Finite Element Method. In section 4 we will give conclusions and perspective for future work.

2. HELICOPTER MODEL

In order to illustrate the flight envelope protection, we adopt a planar helicopter model. Only the motion along longitudinal and vertical axes is considered. The x, z -axes of the spatial frame are pointing to north and down directions. The body x -axis is defined from the center of gravity to the nose of the helicopter, and body z -axis is pointing down from the center of gravity. The state vector is defined as $x = [p_x, \dot{p}_x, p_z, \dot{p}_z, \theta, \dot{\theta}]^T \in \mathbb{R}^6$. The inputs vector is defined as $u = [T_M, a]^T \in \mathbb{R}^2$. The equations of motion can be expressed as:

$$(2.1) \quad \begin{bmatrix} \ddot{p}_x \\ \ddot{p}_z \end{bmatrix} = \frac{1}{m} R(\theta) \left[R^T(\alpha) \begin{bmatrix} -D \\ 0 \end{bmatrix} + \begin{bmatrix} -T_M \sin a \\ -T_M \cos a \end{bmatrix} \right] + \begin{bmatrix} 0 \\ g \end{bmatrix}$$

$$(2.2) \quad \ddot{\theta} = \frac{1}{I_y} (M_M a + h_M T_M \sin a)$$

where $\alpha = \theta - \gamma$ is the angle of attack, $\gamma = \tan^{-1}(\dot{p}_z/\dot{p}_x)$ is the flight path angle, $D = \frac{f_p V^2}{2}$ is the drag force. The output equation is the following:

$$(2.3) \quad y_1 = V$$

$$(2.4) \quad y_2 = p_z$$

where V represents the ground speed $V = \sqrt{\dot{p}_x^2 + \dot{p}_z^2}$ and flight path angle as $\gamma = \tan^{-1}(\dot{p}_z/\dot{p}_x)$.

The system, as shown above, is six dimensional. As a first approximation, we will consider the dynamics of θ with respect to the control a fast enough to consider θ itself as an input. After this assumption the system will still have four states.

The dynamic equations become:

$$(2.5) \quad \begin{bmatrix} \ddot{p}_x \\ \ddot{p}_z \end{bmatrix} = \frac{1}{m} R(\theta) \left[R^T(\alpha) \begin{bmatrix} -D \\ 0 \end{bmatrix} + \begin{bmatrix} -T_M \sin \theta \\ -T_M \cos \theta \end{bmatrix} \right] + \begin{bmatrix} 0 \\ g \end{bmatrix}$$

The Flight Management System (FMS) is responsible for planning and controlling the operation of the UAV, which is equipped with a detector for the detection and investigation of objects of interest. The FMS consists of four layers, the strategic, tactical, and trajectory planners, and the regulation layer, as described in Figure 1[4].

The **Strategic Planner** is concerned with the planning and execution of the central UAV mission. It designs a coarse, self-optimal trajectory, which is stored in form of a sequence way-points. gives a list of way points that are planned for the mission. This layer also takes care of the transition between the points, by acknowledging the completion of a subtask and scheduling the next one.

The **Tactical Planner** is responsible for the coordination and execution of behaviors, and is able to overrule the behavior proposed by the strategic planner, in case of safety critical situations such as collision avoidance. Tactical planner functions as a supervisory control. Safety must be guaranteed at this level. Conflict resolution is the mode with higher priority, followed by the Flight Envelope Protection mode and finally the goal of the mission.

The **Trajectory Planner** uses a detailed dynamic model, sensory input, and the output trajectory, to design a full state and nominal input trajectory for the UAV, and the sequence of *flight modes* necessary to execute the dynamic plan. The trajectory planner, given the information about the type of flight mode chosen by the tactical planner, executes it choosing the corresponding outputs

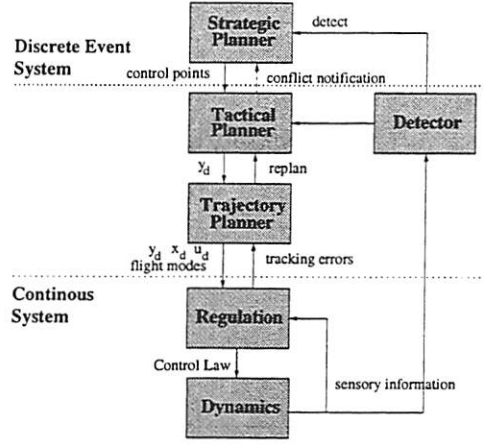


FIGURE 1. System Architecture

and the appropriate controllers. Several types of trajectories can interpolate the way points. Each basic flight mode has then associated with it a particular combination of control actions. Each time an elementary flight mode becomes active the corresponding type of controls are requested to be used at the regulation layer.

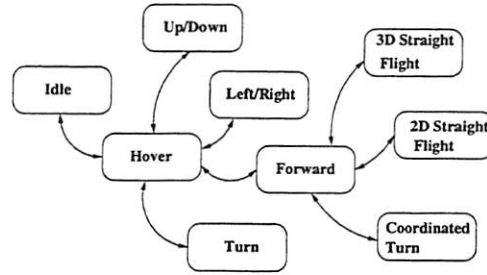


FIGURE 2. Transitions diagram

The **Regulation Layer**, together with the plant, represents the continuous part of the system. The regulation layer has access to sensory information about the actual state of the UAV to generate control signal to regulate it on the given trajectory. It consists of the so-called auto-pilot, which receives the flight mode and computes the input signals to the vehicle according to the system state.

The **Detector** has limited range of detection capability. This block constitutes the connection between the helicopter and the environment. It is crucial for investigation mission, and for obstacle detection.

For envelope protection purposes the unsafe set is shown in figure 3:

The figure shows a qualitative representation of the unsafe sets. The dimension of the region depends on the parameters of the helicopter. A very detailed description of the flight envelope for an helicopter can be found in [3]. The two regions shown above are the unsafe sets. In case of engine failure there are control actions that allow the helicopter to safely perform an emergency landing. In the vertical set it is impossible to demonstrate safe auto-rotative landings. The horizontal set (high speed portion of the unsafe set) is simply a warning that a power failure at high speed close to the ground is a dangerous situation. It is therefore an empirical curve designed based upon what is considered to be

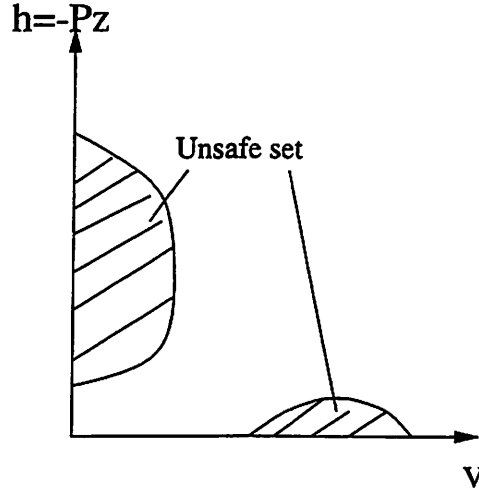


FIGURE 3. Flight Envelope

prudent. non analytical method has been developed for constructing this portion of the diagram. In our approach we will omit it.

3. FLIGHT ENVELOPE PROTECTION CONTROLLER SYNTHESIS

For a formal treatment of controller synthesis for safety specification, please refer to[1]. The safe set can be described as a first order approximation by the following set:

$$(3.1) \quad F = \{x \in \mathbb{R}^3 \mid V \geq 0, V^2 + (h - h_0)^2 - r^2 \geq 0, V < V_{max}, h < h_{max}\}$$

By setting bounds on maximal velocity and height F is a compact set. The state p_x is not included in the specification of the safe set and all the other states are decoupled from it. For the synthesis of our controller we will consider a third order system derived from 2.5 by dropping the state p_x . We are therefore left with a three dimensional optimal control problem. In the rest of the section we will formulate the flight envelope controller synthesis as an optimal control problem. Consider the system (2.5) over the time interval $[t, 0]$, where $t < 0$. The value function is given by

$$(3.2) \quad J(x, u(\cdot), t) : \mathbb{R}^n \times \mathbb{U} \times \mathbb{R}_- \rightarrow \mathbb{R}$$

The value function represents the cost of the trajectory. Notice that the cost depends only on the final state, i.e. there is no running cost. Optimal cost is given by:

$$(3.3) \quad J^*(x, t) = \max_{u(\cdot) \in \mathbb{U}} J(x, u(\cdot), t)$$

The optimal Hamiltonian is given by:

$$(3.4) \quad H^*(x, p) = \max_{u \in \mathbb{U}} p^T f(x, u)$$

where $f(x, u)$ is the vector field described in equation 2.5.

The evolution of $J^*(x, t)$ follows the Hamilton-Jacobi equation.

$$(3.5) \quad \begin{aligned} -\frac{\partial J^*(x, t)}{\partial t} &= \min\{0, H^*(x, \frac{\partial J^*(x, t)}{\partial x})\} \\ J^*(x, 0) &= J(x) \end{aligned}$$

The equation in the form of 3.5 is not solvable analytically. A numerical method is needed in this case. In the next section we will propose the use of Finite Element Method (FEM) for numerically evaluating the solution to the Hamilton-Jacobi equation.

4. FINITE ELEMENT APPROACH

The Finite Element Method represents a general tool for numerical solution of partial differential equations. Finite dimensional approximations are very important from a computational point of view, since they allow for numerical solution to problems in the continuum otherwise intractable. Several methods exist. In this section we will propose FEM approach by using the method of weighted residuals in the Galerkin formulation. For more details on the method please refer to [2]. Never before Finite Element Method had been applied to the solution of an optimal control problem in the Hamilton-Jacobi PDE formulation. In order to validate the approach we will face the problem of aircraft flight envelope protection controller synthesis solved by Lygeros et al. in [1]. An analytical solution is given there. In this context we can make a comparison between the real solution and the approximated one for the purpose of evaluating the novel approach. A detailed description of flight envelope protection controller synthesis for the aircraft case can be found in [1].

In the use of FEM the basic idea consists in partitioning the state space with finitely many elements, solving the PDE for each element and then assemble each element in the mesh by specifying the appropriate boundary conditions. The flight envelope for the aircraft case is a rectangle. Differently from [1] we will introduce a disturbance $d(x, t)$, which is attributable to the approximation error. For simplicity we will consider it constant for the rest of the discussion. We will play a game with the approximation error.

$$(4.1) \quad J^*(x, t) = \max_{u(\cdot) \in U} \min_{d(\cdot) \in D} [\hat{J}(x, u(\cdot), t) + d(\cdot)]$$

The Hamilton-Jacobi PDE then becomes:

$$(4.2) \quad -\frac{\partial \hat{J}^*(x, t)}{\partial t} = \min\{0, \max_{u(\cdot) \in U} \min_{d(\cdot) \in D} \left\{ \frac{\partial \hat{J}^*(x, t)}{\partial x} f(x, u(\cdot)) + \delta(d(\cdot)) \right\}\}$$

We will use a master element of the type depicted in figure 4.

Next we will define an isoparametric mapping between the (ξ, η) and the (x, y) coordinates in the following way. Given the coordinates of the four corner points on the element (\hat{x}_i, \hat{y}_i) , with $i = 1..4$, all the other points are computed through the following transformation:

$$(4.3) \quad \begin{aligned} x &= \sum_{i=1}^4 N_i \hat{x}_i \\ y &= \sum_{i=1}^4 N_i \hat{y}_i \end{aligned}$$

Notice that the N_i are of the following form:

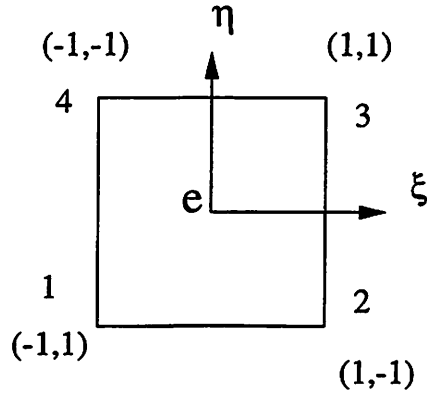


FIGURE 4. Master Element

$$\begin{aligned}
 (4.4) \quad N_1(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 - \eta) \\
 N_2(\xi, \eta) &= \frac{1}{4}(1 + \xi)(1 - \eta) \\
 N_3(\xi, \eta) &= \frac{1}{4}(1 + \xi)(1 + \eta) \\
 N_4(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 + \eta)
 \end{aligned}$$

Also $\sum_{i=1}^4 N_i = 1, \forall (\xi, \eta) \in [-1, 1] \times [-1, 1]$.

With this construction the knowledge of the function J at the edges will be sufficient to know the function in every point in the element:

$$(4.5) \quad J_e(x, t) = \sum_1 N_k(\xi, \eta) \hat{J}_k(t)$$

The weighted residual formulation of the differential equation $F(x, t)$ is the following:

$$(4.6) \quad \int_0^T w_t(t) \left[\int_{\Omega_e} w(x) F(x, t) dx \right] dt = 0, \forall w_t, w$$

Let's choose the weighting functions $w(x)$ to be:

$$(4.7) \quad w(x) = \sum_k N_k(\xi, \eta) w_k,$$

where w_k , are arbitrary constants that give each node a weight.

Notice that:

$$(4.8) \quad \frac{\partial \hat{J}_e}{\partial x} = [\hat{J}_1(t) \cdots \hat{J}_K(t)] B^T,$$

$$(4.9) \quad \frac{\partial \hat{J}_e}{\partial t} = \sum_{k=1}^4 N_k(\xi, \eta) \dot{J}_k(t)$$

where $B \in R^{K \times 2}$, $B = [N_{ij}]$ with $N_{ij} = \frac{\partial N_i}{\partial x_j}$, $i = 1, 2$, $j = 1, \dots, 4$. Let's consider first the inner integral of equation 4.6.

$$(4.10) \quad [\omega_1 \cdots \omega_4] \int_{\Omega_e} \begin{bmatrix} N_1 \\ \vdots \\ N_4 \end{bmatrix} \left([N_1 \cdots N_4] \begin{bmatrix} \dot{J}_1 \\ \vdots \\ \dot{J}_4 \end{bmatrix} + \min \left(0, \max_{u \in U} \left\{ \frac{\partial \hat{J}_e}{\partial x} f(x, u) + \bar{\delta} \right\} \right) \right) dx$$

As we said above the weighting functions are arbitrary. We will use the same weighting factor for each node, i.e. we will set the w_i to 1. We will also assume $\bar{\delta}$ constant. Substituting equations 4.8 and 4.9 into 4.10 yields:

$$(4.11) \quad [1 \cdot 1] \left(\int_{\Omega_e} \begin{bmatrix} N_1 \\ \vdots \\ N_4 \end{bmatrix} [N_1 \cdot N_4] |\mathcal{J}| d\xi d\eta \right) \begin{bmatrix} \dot{J}_1 \\ \vdots \\ \dot{J}_4 \end{bmatrix} + \int_{\Omega_e} \begin{bmatrix} N_1 \\ \vdots \\ N_4 \end{bmatrix} \min \left(-\bar{\delta}, \max_{u \in U} \left\{ \widehat{f^T}(\xi, \eta, u) B \begin{bmatrix} \dot{J}_1 \\ \vdots \\ \dot{J}_4 \end{bmatrix} \right\} \right) |\mathcal{J}| d\xi d\eta$$

where $|\mathcal{J}|$ is the determinant of the jacobian of the isoparametric transformation. Applying Gaussian quadrature the integrals can be solved. After integration we are left with the following integro-differential equation:

$$(4.12) \quad \int_0^{-T} w_t \left(\Gamma \begin{bmatrix} \dot{J}_1 \\ \vdots \\ \dot{J}_4 \end{bmatrix} + \Lambda \begin{bmatrix} \dot{J}_1 \\ \vdots \\ \dot{J}_4 \end{bmatrix} \right) dt = 0$$

After solving the equation above for each element in the mesh we need to assembly in order to satisfy the conditions at the boundary of each component. Each node has a local and a global index. A good assembly process will provide a sparse matrix with small bandwidth, with non zero element around the diagonal. Such a configuration will be fundamental for fast implementation.

5. CONCLUSIONS

In this project we presented a novel approach to solve optimal control theory using FEM as numerical tool. The formulation is intended to deal with those problems for which an analytical solution cannot be found. The approximation error is treated as a disturbance in order to have a conservative approximation of the safe set. The ultimate goal is to design a flight envelope protection controller for a two dimensional helicopter model. In future work we will implement the proposed formulation and apply it to the aircraft case. Results will be compared with the analytical solution for validation purposes.

ACKNOWLEDGMENT

Research is supported by grants (ARO)DAAH04-96-1-0341, (DARPA)F33615-98-C-3614, (ONR)N00014-97-1-0946. The authors would like to acknowledge Kostas Adam and John Lygeros for their invaluable advice.

REFERENCES

- [1] John Lygeros, Claire Tomlin, and Shankar Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, April, 1999.
- [2] P Papadopoulos. *ME 280A Class Notes*. Berkeley, 1997.
- [3] Raymond W. Prouty. *Helicopter Performance, Stability, and Control*. Krieger Publishing Co., Inc., 1995.
- [4] T.J.Koo, B. Sinopoli, A. Sangiovanni-Vincentelli, and S. Sastry. A formal approach to reactive system design: A uav flight management system design example. In *Proceedings of the CACSD*, 1999.
- [5] C. Tomlin, G. Pappas, J. Lygeros, D. Godbole, and S. Sastry. A next generation architecture for air traffic management systems. In Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems IV*. Springer Verlag, New York, 1997.

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES, UNIVERSITY OF CALIFORNIA AT BERKELEY, BERKELEY, CA 94720, TEL. (510) 643-5806, FAX (510) 642-1341

E-mail address: sinopoli@eecs.berkeley.edu

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES, UNIVERSITY OF CALIFORNIA AT BERKELEY, BERKELEY, CA 94720, TEL. (510) 643-5798, FAX (510) 642-1341

E-mail address: koo@eecs.berkeley.edu

MULTIPLE-AGENT PROBABILISTIC PURSUIT-EVASION GAMES IN 2.5D

Shahid Rashid Hyoun Jin Kim

Dept. Electrical Eng. & Comp. Science
University of California at Berkeley
275M Cory Hall #1770, Berkeley, CA 94720-1770
Phone: (510) 643-5190, Fax: (510) 642-1341
{rashid,jin}@eecs.berkeley.edu

May 16, 1999

1 Introduction

This project addresses the problem of controlling a swarm of autonomous agents in the pursuit of multiple evaders in 2.5D. To this effect we develop a probabilistic framework for pursuit-evasion games involving multiple ground agents and an eye-in-the-sky (e.g. an aerial reconnaissance vehicle). The problem is nondeterministic because the motions of the pursuers/evaders and the devices they use to sense their surroundings require probabilistic models. It is also assumed that when the pursuit starts the map of the region is unknown. A probabilistic framework for pursuit-evasion games avoids the conservativeness of deterministic worst-case approaches.

Pursuit-evasion games[1, 2] arise in numerous situations. Typical examples are search and rescue operations, localization of (possibly moving) parts in a warehouse, search and capture missions, etc.

Deterministic pursuit-evasion games on finite graphs have been well studied [3, 4]. In these games, the region in which the pursuit takes place is abstracted to be a finite collection of nodes and the allowed motions for the pursuers and evaders are represented by edges connecting the nodes. An evader is “captured” if he and one of the pursuers occupy the same node. A question often studied within the context of pursuit-evasion games on graphs is the computation of the search number $s(G)$ of a given graph G . By the “search number” it is meant the smallest number of pursuers needed to capture a single evader in finite time, regardless of how the evader decides to move. It turns out that determining if $s(G)$ is smaller than a given constant is NP-hard [4, 5]. Pursuit-evasion games on graphs have been limited to worst-case motions of the evaders.

The pursuit-evasion games were extended to known polygonal environments [6] and simply-connected, smooth-curved, two-dimensional environment [7], based on the principle that for every graph G there is a polygonal free space F such that game on G is equivalent to the game on F , when “capture” is defined as having an evader in one of the lines of visibility of pursuers.

Until recently the literature on pursuit-evasion games always assumed the region on which the pursuit takes place (be it a finite graph or a continuous terrain) is known. When the region is unknown *a priori* a “map-learning” phase is often proposed to precede the pursuit. However, systematic map learning is time consuming and computationally hard, even for simple two-dimensional rectilinear environments with each side of the obstacles parallel to one of the coordinate axis [8]. In practice, map learning is further complicated by the fact that the sensors used to acquire the data upon which the map is built are not accurate. A probabilistic framework is also natural to take into account the fact that sensor information is not precise and that only an inaccurate *a priori* map of the terrain may be known. Recent work [9] combined exploration (or map-learning) and pursuit in a single problem in a probabilistic framework to avoid the conservativeness inherent to worst-case assumptions on the motion of the evader. It was proven that in pursuit-evasion games with partial observations and obstacles under mild assumptions, there exists a pursuit policy with certain “persistency” which guarantees to find an evader in finite time with probability one, or the expected time needed to find the evader is finite. Thus the answer to reachability question (can the fleet of pursuers catch the random evader?) is *almost sure*[10].

Here we expand this previous work into 2.5-dimensional games with multiple evaders.

2 Rules of the Game

In this section we describe a specific pursuit-evasion game with partial observations and obstacles to which the pursuit policies can be applied. In this game the pursuit takes place in a rectangular two-dimensional grid with n_c square cells numbered from 1 to n_c . We say that two distinct cells $x_1, x_2 \in \mathcal{X} \triangleq \{1, 2, \dots, n_c\}$ are *adjacent* if they share one side or one corner. In the sequel we denote by $\mathcal{A}(x) \subset \mathcal{X}$ the set of cells adjacent to some cell $x \in \mathcal{X}$. Each $\mathcal{A}(x)$ will have, at most, 8 elements. The position of obstacles are represented by *obstacle map* $\mathbf{m} : \mathcal{X} \rightarrow \{0, 1\}$ that takes the value 1 precisely at those cells that contain an obstacle. The motion of the ground pursuers and evaders is constrained in that each can only remain in the same cell or move to an empty cell adjacent to its present position. This means that if at a time $t \in \mathcal{T}$ the pursuers are positioned in the cells $v \triangleq \{v_1, v_2, \dots, v_{n_p}\} \in \mathcal{U}$, then the subset of \mathcal{U} consisting of those lists of cells to which the pursuers could move at time $t + 1$, were these cells empty, is given by

$$\mathcal{U}(v) \triangleq \{\{\bar{v}_1, \bar{v}_2, \dots, \bar{v}_{n_p}\} \in \mathcal{U} : \bar{v}_i \in \{v_i\} \cup \mathcal{A}(v_i)\}.$$

Each pursuer is capable of determining its current position and sensing the cells adjacent to the one it occupies for obstacles/evaders. The pursuers are also able to distinguish between the enemies in the game. Each measurement $\mathbf{y}(t)$, $t \in \mathcal{T}$, therefore is a triple $\{\mathbf{v}(t), \mathbf{e}(t), \mathbf{o}(t)\}$ where $\mathbf{v}(t) \in \mathcal{U}$ denotes the measured positions of the pursuers, $\mathbf{e}(t) \subset \mathcal{X}$ a set of cells where each evader was detected, and $\mathbf{o}(t) \subset \mathcal{X}$ a set of cells where obstacles were detected. For this game we then have $\mathcal{Y} \triangleq \mathcal{U} \times 2^{\mathcal{X}} \times 2^{\mathcal{X}}$. For simplicity, we shall assume that $\mathbf{v}(t)$ reflect accurate measurements and therefore $\mathbf{v}(t) = \mathbf{x}(t)$, $t \in \mathcal{T}$. We also assume that the detection of evaders is perfect for the cells in which the pursuers are located, but not for adjacent ones. To account for imperfect sensors for evader detection, we introduce the *probability of false positives* and q the *probability of false negatives*: the probability $p \in [0, 1]$ of a pursuer detecting an evader in a cell adjacent to its current position, given that none was there, and the probability $q \in [0, 1]$ of not detecting an evader, given that it was there. For calculational simplicity we shall assume that the sensors used for obstacle detection are perfect in that $\mathbf{o}(t)$ contains precisely those cells adjacent to the pursuers that contain an obstacle.

The observations and movements of the eye-in-the-sky are somewhat different from those of the pursuers. Most importantly, the eye-in-the-sky cannot 'capture' an evader. The eye-in-the-sky is allowed to occupy the same cell as an obstacle (think of this as flying over the obstacle). We also allow the eye-in-the-sky to have a larger set of points over which it can detect evaders, $\mathcal{A}(\mathcal{A}(x))$ (basically a region of radius two centered around its present location) and assume that it has perfect sensors for the detection of enemies. At the same time, we decided to prevent the eye-in-the-sky from making any observations in cells adjacent to obstacles, $\mathcal{A}(\omega)$, where ω is the set of obstacle positions (think of this as a situation where the evader is hiding under a tree top). Since the eye-in-the-sky cannot accurately pinpoint obstacle locations, we do not allow it to update the obstacle map. Refer to the measurements made by the eye-in-the-sky as $\mathbf{a}(t)$, $t \in \mathcal{T}$, which is a double $\{\mathbf{v}_a(t), \mathbf{e}_a(t)\}$ where $\mathbf{v}_a(t) \in \mathcal{U}$ denotes the measured position of the eye-in-the-sky and $\mathbf{e}_a(t) \subset \mathcal{X}$ a set of cells where each evader was detected. As with the pursuers assume that $\mathbf{v}_a(t)$ reflect accurate measurements and therefore $\mathbf{v}_a(t) = \mathbf{x}(t)$, $t \in \mathcal{T}$. As long as the eye-in-the-sky does not detect an enemy, it maintains a motion in which it sweeps out the entire map. Once an enemy is sighted, the eye-in-the-sky uses the simple pursuit strategy of moving towards the location at which the enemy was detected, until the enemy is no longer detected (captured by a pursuer or hiding next to an obstacle) at which time it returns to its 'sweeping' strategy.

Pursuers assume a Markov model for the motion of the evaders. The model is completely determined by a scalar parameter $\rho \in [0, 1/8]$ that represents the probability of the evader moving from its present position to an adjacent cell with no obstacles. This probability is independent of the specific pursuit policy being used. This means that, for each $x, \bar{x} \in \mathcal{X}$, $n \in \{0, 1, \dots, 8\}$,

$$P(\mathbf{x}_e(t+1) = x \mid \mathbf{x}_e(t) = \bar{x}, \mathbf{m}(x) = 0, \mathbf{n}_0(\bar{x}) = n) = \begin{cases} \rho & x \in \mathcal{A}(\bar{x}) \text{ and } \mathbf{m}(x) = 0 \\ 1 - (|\mathcal{A}(\bar{x})| - n)\rho & x = \bar{x} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $|\mathcal{A}(x)| \in \{3, 5, 8\}$ denotes the number of cells adjacent to x and $\mathbf{n}_o(\bar{x}) \in \{0, 1, \dots, 8\}$ the number of obstacles in $\mathcal{A}(\bar{x})$. This models a situation in which the moving evaders are not actively avoiding detection.

3 Updating Probability

We describe next how to compute the (conditional) posterior probability $p_{e(i)}(x, Y_t)$ of the evader i being in cell x at time $t + 1$, given the measurements $\mathbf{Y}_t = Y_t$ taken up to time $t \triangleq |Y_t|$. These steps are repeated for each enemy (left) in the game, to update the unique probabilistic location map of each enemy. As each of these maps are developed independently, the subscripts for the particular enemy are omitted for the rest of the discussion. The derivations below are independent of the specific pursuit policy \bar{g} in use and we therefore drop the subscript \bar{g} in the probability measure. For computational efficiency $p_e(x, Y_t)$ is computed recursively in two steps:

1. A *measurement step* The probability $P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t)$ of the evader being in cell x at time t , given the measurements $\mathbf{Y}_t = Y_t$ taken up to t , is computed based on the probability $p_e(x, Y_{t-1}) \triangleq P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_{t-1} = Y_{t-1})$ of the evader being in cell x at time t , given the measurements $\mathbf{Y}_{t-1} = Y_{t-1}$ taken up to $t - 1$, and the last measurement $y(t)$ in the sequence Y_t . The sensor model is used in this step. we use Bayes' rule repeatedly to write

$$P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t) \approx \alpha p_e(x, Y_{t-1}) \begin{cases} 0 & x \in o(t) \cup v(t) \setminus e(t) \text{ or } \exists \bar{x} \neq x : \bar{x} \in v(t) \cap e(t) \\ p^{k_1}(1-p)^{k_2}q^{k_3}(1-q)^{k_4} & \text{otherwise} \end{cases} \quad (2)$$

where α is a normalizing constant, chosen so that $\sum_{x \in \mathcal{X}} P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t) = 1$. And k_1 is the number of pursuers that reported in $e(t)$ seeing the evader at cells other than x that are adjacent to their one (false positives); k_2 is the number of pursuers that reported in $e(t)$ not seeing the evader at cells other than x that are adjacent to their one (true negatives); k_3 is the number of pursuers that reported in $e(t)$ not seeing the evader at the cell x adjacent to their one (false negatives); and k_4 is the number of pursuers that reported in $e(t)$ seeing the evader at the cell x adjacent to their one (true positives). $k_1 + k_2 + k_3 + k_4$ must be equal to the number of cells adjacent to any of the pursuers positions in $v(t)$.

2. An *eye-in-the-sky update step* was added here for the cases in which an eye-in-the-sky was part of the game. This update was purposely put after the pursuers measurement step to give the eye-in-the-sky's observations higher authority.

$$P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t) = \alpha P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t) \begin{cases} 1 & x \in e_a(t) \\ 0 & x \in \mathcal{A}(\mathcal{A}(v_a(t))) \setminus \mathcal{A}(\omega) \text{ if not using greedy strategy} \end{cases} \quad (3)$$

where α is again a normalizing constant chosen as above. ¹ Furthermore, if $e_a(t) \neq \emptyset$

$$P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t) = \alpha P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t) \begin{cases} 1 & x \in e_a(t) \\ 0 & x \in \mathcal{X} \setminus e_a(t) \text{ if not using greedy strategy} \end{cases} \quad (4)$$

- if an enemy is found, the position in which it was found is set to have probability one, and the rest of the map has probability zero (if not using the greedy strategy²).

¹ this is a blatant abuse of notation and convention done for simplicity - think of these update equations in the sense of a computer program

² to be described in the following section

3. A *prediction step* The probability $p_e(x, Y_t) \triangleq P(\mathbf{x}_e(t+1) = x \mid \mathbf{Y}_t = Y_t)$ of the evader being in cell x at time $t+1$, given the measurements $\mathbf{Y}_t = Y_t$ taken up to time t , is then computed from $P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t)$. The evader's motion model is used in this step.

$$\begin{aligned} p_e(x, Y_t) &= \sum_{\bar{x} \in \{x\} \cup \mathcal{A}(x)} P(\mathbf{x}_e(t+1) = x, \mathbf{x}_e(t) = \bar{x}, \mathbf{m}(x) = 0 \mid \mathbf{Y}_t = Y_t) \\ &\approx P(\mathbf{x}_e(t+1) = x \mid \mathbf{x}_e(t) = x, \mathbf{Y}_t = Y_t) P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t) \\ &\quad + \sum_{\bar{x} \in \mathcal{A}(x)} P(\mathbf{x}_e(t+1) = x \mid \mathbf{x}_e(t) = \bar{x}, \mathbf{m}(x) = 0, \mathbf{Y}_t = Y_t) \\ &\quad \quad P(\mathbf{x}_e(t) = \bar{x} \mid \mathbf{Y}_t = Y_t) P(\mathbf{m}(x) = 0 \mid \mathbf{Y}_t = Y_t) \end{aligned}$$

Here, we used the fact that $\mathbf{x}_e(t) = x$ automatically implies that $\mathbf{m}(x) = 0$, and also the low obstacle density assumption to conclude that $\mathbf{m}(x)$ is approximately independent of the position $\mathbf{x}_e(t+1) = \bar{x}$ of the evader, when $x \neq \bar{x}$. From (1) and the low obstacle density assumption, we conclude that

$$\begin{aligned} p_e(x, Y_t) &\approx (1 - |\mathcal{A}(x)|\rho) P(\mathbf{x}_e(t) = x \mid \mathbf{Y}_t = Y_t) \\ &\quad + \rho P(\mathbf{m}(x) = 0 \mid \mathbf{Y}_t = Y_t) \sum_{\bar{x} \in \mathcal{A}(x)} P(\mathbf{x}_e(t) = \bar{x} \mid \mathbf{Y}_t = Y_t). \end{aligned} \quad (5)$$

$P(\mathbf{m}(x) = 0 \mid \mathbf{Y}_t = Y_t)$, $x \in \mathcal{X}$ can be computed recursively. similar to the computation of the $p_e(x, Y_t)$. Since we are assuming that the sensor used for obstacle detection is perfect and obstacles do not move, we simply have

$$P(\mathbf{m}(x) = 0 \mid \mathbf{Y}_t = Y_t) = \begin{cases} 1 & x \notin o(t) \text{ and is in or adjacent to an element in } v(t) \\ 0 & x \in o(t) \text{ and is in or adjacent to an element in } v(t) \\ P(\mathbf{m}(x) = 0 \mid \mathbf{Y}_{t-1} = Y_{t-1}) & \text{otherwise} \end{cases}$$

It is critical to keep in mind that these steps are done independently for each enemy left in the game and that these steps depend only on information of that particular enemy (i.e. it's unique probabilistic history and whether or not it was sighted during the last time instant).

4 Pursuit Policy

As mentioned earlier, a distinct map is maintained for each enemy in the game. Once an enemy is captured its map is not updated nor is it observed by the pursuers or eye-in-the-sky again. A couple of different pursuit algorithms were experimented with in order to reduce capture time of the enemies

4.1 Greedy Policy

The original strategy implemented by Hespanha and Kim [9] was a greedy policy in which each pursuer moved to the adjacent cell (or possibly did not move at all) which had the highest probability of containing the enemy (recall, the original work only had one enemy in the game). Our extension of this strategy was fairly simple, each pursuer moved to the adjacent cell (or remained in its current location) with the highest probability of containing an enemy over all the enemy maps.

$$\mathbf{x}(t+1) = \max_{\text{enemies}} \max_{\mathcal{A}(\mathbf{x}(t))} p_{e(i)}(\mathbf{x}, Y_t) \quad (6)$$

where again, $p_{e(i)}(\mathbf{x}, Y_t)$ represents the probability of enemy i being at position \mathbf{x} at time $t+1$.³

When using the greedy search strategy, we had the eye-in-the-sky set a high value in positions where it detected an enemy, otherwise it did nothing. Ideally, we would have liked to set the enemy map values at the locations where the enemy was not sighted to zero, but this would effectively create a buffer zone of low probability around a sighted enemy through which an agent following a greedy strategy would not pass.

³The appropriate exception handling was added to prevent an pursuer from moving into a cell with an obstacle or a cell already occupied by another pursuer

4.2 Global Maximum Policy

In order to have the pursuer more effectively respond to the sighting of an enemy, we implemented a global maximum search strategy, where each pursuer set a trajectory to the global location with the greatest (weighted) probability of there being an enemy at the next time instant.⁴

$$\mathbf{x}(t+1) = \text{trajectory}(\max_{\text{enemies}} \max_{x \in \mathcal{X}} q(\mathbf{x}(t), x) * p_{e(i)}(x, Y_t)) \quad (7)$$

trajectory is just a function to determine which discrete cell location to move to in order to follow a path to the desired location. To still encourage local search, a weighting, q , is also added to the algorithm. Three different weightings were tested:

$$q_0(\mathbf{x}(t), x) = 1 \quad (8)$$

$$q_1(\mathbf{x}(t), x) = \frac{1}{d((\mathbf{x}(t), x))} \quad (9)$$

$$q_2(\mathbf{x}(t), x) = \frac{1}{d((\mathbf{x}(t), x))^2} \quad (10)$$

$$(11)$$

where $d((\mathbf{x}(t), x))$ is the distance between the pursuers current position and the desired position calculated as:

$$d((y_1, y_2), (z_1, z_2)) = \max(|y_1 - z_1|, |y_2 - z_2|) \quad (12)$$

When using the global maximum search strategy, we have the eye-in-the-sky set to zero the value of all map positions in which it does not see an enemy and set to one those positions in which it does see an enemy. Additionally, if the eye-in-the-sky does see an enemy, it sets the rest of the map for that particular enemy to zero elsewhere.

4.3 Combined Policy

The major drawback to the global maximum strategy is that the system frequently evolves to the state where all the pursuers are going to a common position and then subsequently they begin to act as one joint pursuer as opposed to a number of independent ones. To remedy this problem a combined greedy-global maximum policy was instituted. Basically the pursuers followed a global maximum search strategy unless a pursuer was directed to go to a specific location to which another pursuer was already heading. In this case the later pursuer switched over to just using a greedy search strategy. Despite the movement strategy a particular pursuer was using, the updates of the eye-in-the-sky were based on the global maximum policy.

5 Simulation & Results

Figures(1-3) shows a run of the combined policy (with weighting q_1) for the case of three enemies and an eye-in-the-sky. Pursuers are represented as ★ (not filled), enemies as ♦, the eye-in-the-sky as a ◇, and obstacles as *. The shading of each map represents the probability of the enemy being in that location (darker = higher probability). Figure(1) is a snapshot before any movements have taken place. The map for enemy 1 is white because the eye-in-the-sky has detected it (enemy 1 is located in the same spot as the dark square). The map for enemy 2 is darker than that of enemy 3, because the pursuer in the upper left has falsely detected enemy 3, thereby (wrongly) reducing the probability of it appearing anywhere else. Also note that the eye-in-the-sky does not have full vision over its maximum possible range because of obstacles. Figure(2) shows the capture of enemy 1. Notice that not all the pursuers were not following it. Figure(3) shows the capture of the last enemy, enemy 3. Here, all the pursuers were converging on it. Note that the maps of the enemies do not update once they have been caught and that is why there are inconsistencies between the maps in Figure(3).

For each of the search strategies 100 simulation runs were performed for both the case with and without an eye-in-the-sky for each of one through five enemies being present. The average capture time of all enemies are shown in Tables (1-3). Without an eye-in-the-sky, the greedy strategy shows the best performance and the global maximum

⁴ Again, the proper exception handling was added to prevent a pursuer from moving into a cell with another pursuer or an obstacle

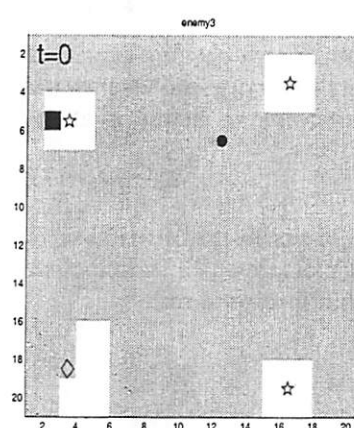
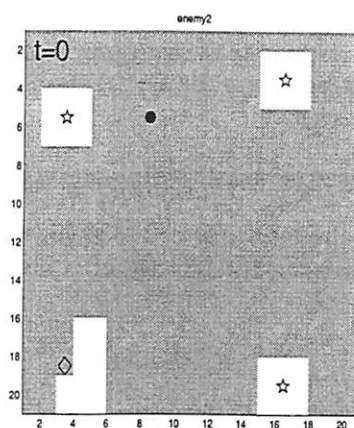
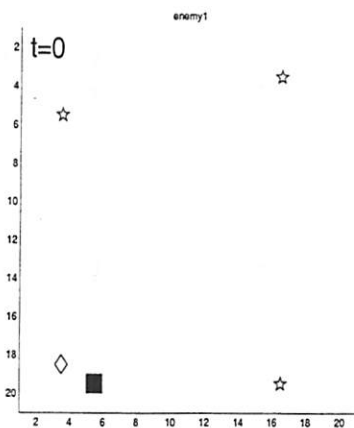


Figure 1: Before First Movements

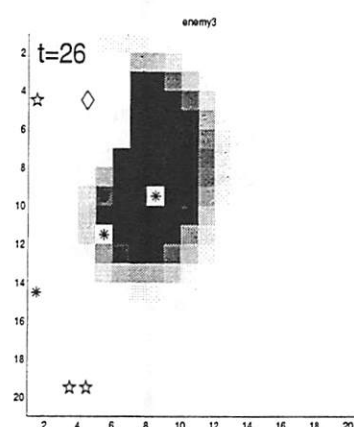
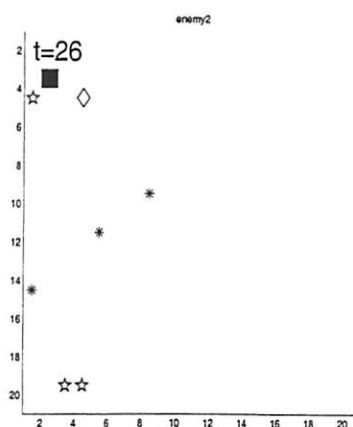
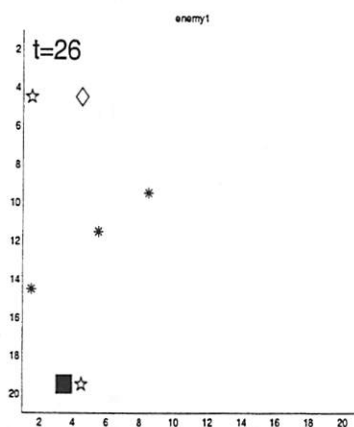


Figure 2: First Enemy Capture

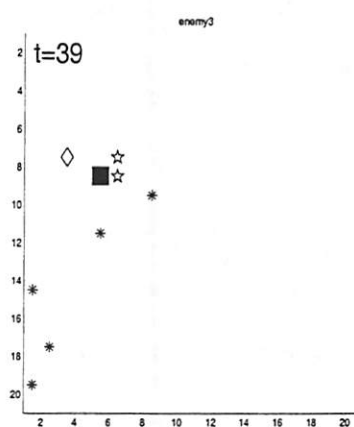
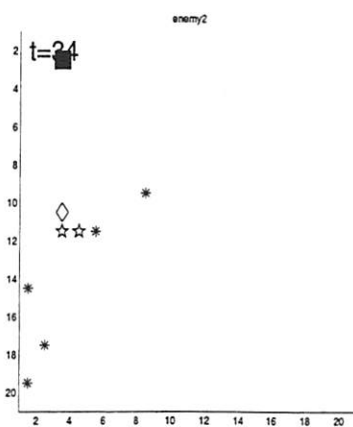
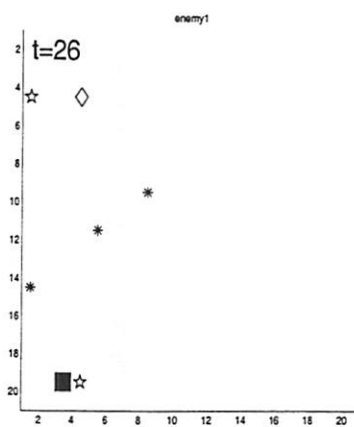


Figure 3: Last Enemy Capture

# of enemies	capture time w/o eye-in-the-sky	capture time with eye-in-the-sky
1	20.75	34.50
2	42.10	35.50
3	46.70	48.60
4	58.45	48.80
5	57.90	53.00

Table 1: Greedy Policy

# of enemies	capture time w/o eye-in-the-sky	capture time with eye-in-the-sky
1	55.87	24.28
2	84.48	39.90
3	88.80	60.24
4	117.01	67.02
5	150.03	75.04

Table 2: Global Maximum Policy (using weighting function q_1)

# of enemies	capture time w/o eye-in-the-sky	capture time with eye-in-the-sky
1	30.54	20.98
2	45.21	35.15
3	50.04	40.11
4	57.55	43.81
5	69.65	49.38

Table 3: Combined Policy (using weighting function q_1)

# of enemies	capture time w/ q_0 weighting fn	capture time w/ q_1 weighting fn	capture time w/ q_2 weighting fn
1	27.50	24.28	27.50
2	50.14	39.90	42.72
3	61.54	60.24	51.11
4	66.17	67.02	66.31
5	83.36	75.04	77.47

Table 4: Various Weighting Functions for Global Maximum Policy

the worst. All the strategies shown an improvement with the addition of an eye-in-the-sky, with the performance of the global maximum strategy significantly better (but this is still worse than either of the other strategies without one). The combined global maximum-greedy policy shows the fastest capture time with an eye-in-the-sky, but the improvement over the greedy algorithm alone is still somewhat minimal.

In an attempt to improve performance of the global maximum search strategy, different weightings, q , were experimented with. The average capture time of 100 runs of the algorithm under each weighting (and always with an eye-in-the-sky) are shown in Table (4). The different weightings do not seem to improve (effect) the performance of the algorithm.

6 Conclusions & Future Research

A multiple agent pursuit-evasion game was extended to include multiple evaders and a 2.5 D environment. It was seen that the addition of an eye-in-the-sky can be utilized to reduce the time required to capture a set of enemies. A number of heuristic strategies were implemented in an effort to reduce capture time.

The next big challenge on the theoretic level is the derivation of an optimal search policy for evaders. On a more experimental level, the addition of artificial intelligence to the pursuers or creating intelligent evaders would also be of interest.

References

- [1] R. Isaacs. *Differential Games*. Wiley, New York, NY, 1965.
- [2] T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, London, 1995.
- [3] T. D. Parsons. Pursuit-Evasion in a graph. In Y. Alani and D. R. Lick, editors, *Theory and Application of Graphs*, pp 426-441. Springer-Verlag, Berlin, 1976.
- [4] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The Complexity of Searching a Graph, In *Journal of the ACM*, Vol. 35(1):18-44, January 1988.
- [5] A. S. Lapaugh. Recontamination does not help to search a graph, In *Journal of ACM*, Vol. 41(2):224-245, April 1993.
- [6] S. M. Lavalle, D. Lin, L. J. Guibas, J.-C. Latombe and R. Motwani. Finding an unpredictable Target in a Workspace with Obstacles. In *IEEE Int. Conf. Robot. & Autom.*, 1997.
- [7] S. M. Lavalle and J. Hinrichsen. Visibility-Based Pursuit-Evasion: The Case of Curved Environments. Submitted to *IEEE Int. Conf. Robot. & Autom.*, 1999.
- [8] X. Deng, T. Kameda and C. Papadimitriou. How to Learn an Unknown Environment I: The Rectilinear Case. In *Journal of the ACM*, Vol. 45(2):215-245, March 1998.
- [9] J. Hespanha, H. J. Kim and S. Sastry. Multiple-Agent Probabilistic Pursuit-Evasion Games Submitted to *IEEE Conf. on Dec. and Contr.*, 1999.
- [10] L. Alfaro, T. Henzinger and O. Kupferman. Concurrent Reachability Games. In *Proc. of FOCS*, 1998

Maximal controlled invariant set and least restrictive controller for discrete-time linear systems

Shawn Schaffert

René Vidal

May 17, 1999

Abstract

In this project, an algorithm for computing the maximal controlled invariant set and least restrictive controller for discrete-time linear systems with a scalar input and disturbance is proposed, and the decidability of the problem is analyzed. It is proved that if the system is in canonically controllable form and the initial set is a rectangle, the problem is decidable. It is also shown that for a general discrete-time linear system and an initial set specified by linear inequalities, the problem is semi-decidable. The proposed algorithm is also illustrated with two examples.

1 Introduction

Consider the discrete time system

$$x(t+1) = f(x(t), u(t), d(t)) \quad (1)$$

where $x \in \mathbb{R}^n$ is the state, $u \in U = \mathbb{R}$ is the control action, $d \in D \subset \mathbb{R}$ is a bounded disturbance and $f : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^n$ is continuous map.

Definition 1 (Controlled invariant set) A set $W \in \mathbb{R}^n$ is called a controlled invariant set of (1) if $\exists u \ni \forall d (x, \square W)$, that is, $x(0) \in W \Rightarrow x(t) \in W \forall t$. Additionally W is called maximal if it is not a proper subset of another controlled invariant set, that is, for all controlled invariant sets W' , we have $W' \subset W$.

Definition 2 (Memoryless controller) A controller $C : \mathcal{X}^* \rightarrow 2^U$, where \mathcal{X}^* is the set of all finite executions of (1) projected onto the state space, is called memoryless if for all finite executions $\mathcal{X}, \mathcal{X}'$ ending at the same state, $C(\mathcal{X}) = C(\mathcal{X}')$. A memoryless controller can be characterized by a map $U : \mathbf{X} \rightarrow 2^U$.

Definition 3 (Least restrictive controller) A memoryless controller $U : \mathbf{X} \rightarrow 2^U$ is called a least restrictive controller if it is maximal among the controllers that solve the problem $(x, \square W)$, that is, for all U' that solve $(x, \square W)$, $U'(x) \subseteq U(x) \forall x \in W$. A controller is said to solve the problem $(x, \square W)$ if it makes the trajectories of the system satisfy $(x, \square W)$.

Using these definitions, the problem of finding the maximal controlled invariant set W and the least restrictive controller $U(x)$ for (1), namely controlled invariant problem (CIP), can be stated as follows:

Definition 4 (CIP) Given a set $F \neq \emptyset$, find, if there exists, $W \subseteq F$ and $U(x)$ such that W and $U(x)$ are the maximal controlled invariant set and least restrictive controller of (1), respectively.

The following algorithm gives an intuitive way of computing such sets.

Initialization

$$W^0 = F, W^{-1} = \emptyset, U^0(x) = U \quad \forall x \in F, l = 0$$

while $W^l \neq W^{l-1}$ **do**
begin

$$\begin{aligned} W^{l+1} &= \{x \in W^l \ni \exists u \forall d f(x, u, d) \in W^l\} \\ U^{l+1}(x) &= \begin{cases} \{u \in U \ni \forall d f(x, u, d) \in W^l\} & x \in W^{l+1} \\ U & x \notin W^{l+1} \end{cases} \\ l &= l + 1 \end{aligned}$$

end

set $W = W^l$ **and** $U(x) = U^l(x)$

In general, there is no straightforward way of implementing the formula $\exists u \forall d f(x, u, d) \in W^l$. For some special systems, however, this implementation can be carried out. Here we consider discrete time linear systems, that is, systems of the form

$$x(t+1) = Ax(t) + bu(t) + cd(t) \quad (2)$$

In section 2, we will prove that the CIP problem is decidable when the pair (A, b) is in canonically controllable form and F is a rectangle. In section 3, we will extend these results to the case in which F is defined by a set of linear inequalities.

2 Rectangular case

In this section, we consider the case where both D and F are bounded rectangles, that is $d \in [d_1, d_2] = D$ and $F = [\alpha, \beta] = [\alpha_1, \beta_1] \times \dots \times [\alpha_n, \beta_n]$ with $\alpha_i \leq \beta_i$ and $[\alpha_i, \beta_i] \subset \mathbb{R}, i = 1 \dots n$. We also assume that (A, b) is in canonically controllable form.

2.1 One dimensional case

Here $F = [\alpha, \beta]$ and

$$x(t+1) = ax(t) + bu(t) + cd(t) \quad b \neq 0 \quad (3)$$

Therefore,

$$W^1 = \{x \ni \exists u \forall d (\alpha \leq x \leq \beta) \wedge (\alpha \leq ax + bu + cd \leq \beta)\} \quad (4)$$

$$= \{x \ni \exists u \psi^1(x, u, d)\} \quad (5)$$

$$U^1(x) = \{u \ni \forall d (\alpha \leq x \leq \beta) \wedge (\alpha \leq ax + bu + cd \leq \beta)\} \quad (6)$$

$$= \{u \ni \psi^1(x, u, d)\} \quad (7)$$

In order to do quantifier elimination on d , we define the function $\delta : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

$$\delta(c, d_1, d_2) = \begin{cases} d_2 & \text{if } c > 0, \\ d_1 & \text{otherwise} \end{cases} \quad (8)$$

Thus $\psi^1(x, u, d)$ is equivalent to

$$(\alpha \leq x \leq \beta) \wedge (\alpha - ax - c\delta(-c) \leq bu \leq \beta - ax - c\delta(c)) \quad (9)$$

where we have written $\delta(c)$ instead of $\delta(c, d_1, d_2)$ as a shorthand notation.

From the last expression, it is clear that given $x \in [\alpha, \beta]$, u exists if and only if

$$\alpha - c\delta(-c) \leq \beta - c\delta(c) \iff |c|(d_2 - d_1) \leq \beta - \alpha \quad (10)$$

Note that this condition is independent of x , therefore, if the condition is satisfied, after one iteration the set W^i does not change. Thus the CIP problem can be solved in one iteration if (10) is true. If false, no solution exists. This result can be summarized as follows.

Lemma 1 *Given system (3) with $F = [\alpha, \beta] \subset \mathbb{R}$ and $\mathbf{D} = [d_1, d_2] \subset \mathbb{R}$, the solution to the CIP problem, obtained after one iteration, is given by:*

$$W = \begin{cases} \{x \ni \alpha \leq x \leq \beta\} & \text{if } |c|(d_2 - d_1) \leq \beta - \alpha \\ \emptyset & \text{otherwise} \end{cases} \quad (11)$$

$$U(x) = \begin{cases} \{u \ni \alpha - ax - c\delta(-c) \leq bu \leq \beta - ax - c\delta(c)\} & \text{if } x \in W \\ \mathbf{U} & \text{otherwise} \end{cases} \quad (12)$$

2.2 Two dimensional case

Here $F = [\alpha, \beta] = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2]$ and (2) is assumed to be in canonically controllable form, i.e.

$$x(t+1) = \begin{pmatrix} 0 & 1 \\ a_{21} & a_{22} \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) + \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} d(t) \quad (13)$$

Extending the definition of $\psi^1(\cdot)$ to $\mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}$ we get that $\psi^1(x, u, d)$ is equivalent to

$$(\alpha_1 \leq x_1 \leq \beta_1) \wedge (\alpha_2 \leq x_2 \leq \beta_2) \wedge (\alpha_1 - c_1\delta(-c_1) \leq x_2 \leq \beta_1 - c_1\delta(c_1)) \wedge \\ (\alpha_2 - a_{21}x_1 - a_{22}x_2 - c_2\delta(-c_2) \leq u \leq \beta_2 - a_{21}x_1 - a_{22}x_2 - c_2\delta(c_2)) \quad (14)$$

From the last expression, it is clear that given $x_1 \in [\alpha_1, \beta_1]$, x_2 exists if and only if

$$((\alpha_1 - c_1\delta(-c_1) \leq \beta_2) \wedge (\alpha_2 \leq \beta_1 - c_1\delta(c_1))) \quad \wedge \quad (\alpha_1 - c_1\delta(-c_1) \leq \beta_1 - c_1\delta(c_1)) \\ \Updownarrow \\ (\alpha_2^1 = \max(\alpha_2, \alpha_1 - c_1\delta(-c_1)) \leq \min(\beta_2, \beta_1 - c_1\delta(c_1)) = \beta_2^1) \quad (15)$$

and u exists if and only if

$$\alpha_2 - c_2\delta(-c_2) \leq \beta_2 - c_2\delta(c_2) \quad (16)$$

Therefore, after the first iteration we have:

$$W^1 = \begin{cases} \{(x_1, x_2) \ni \alpha_1 \leq x_1 \leq \beta_1 \wedge \alpha_2^1 \leq x_2 \leq \beta_2^1\} & \text{if } \alpha_2^1 \leq \beta_2^1 \wedge |c_2|(d_2 - d_1) \leq \beta_2 - \alpha_2 \\ \emptyset & \text{otherwise} \end{cases} \quad (17)$$

$$U^1(x) = \begin{cases} \{u \ni \alpha_2 - \sum_{j=1}^2 a_{2j}x_j - c_2\delta(-c_2) \leq u \leq \beta_2 - \sum_{j=1}^2 a_{2j}x_j - c_2\delta(c_2)\} & \text{if } x \in W^1 \\ \mathbf{U} & \text{otherwise} \end{cases} \quad (18)$$

Thus after one iteration variable x_2 gets restricted, which means that we have to run a second iteration.

In order to run this second iteration, we consider the valuations of α_2^1 and β_2^1 and check conditions (15) and (16) for the new values of α_2 and β_2 . It is straightforward to check that:

1. If $\alpha_2^1 = \alpha_2$ and $\beta_2^1 = \beta_2$, then $W = W^1$ and $U(x) = U^1(x)$. Thus the algorithm converges in one iteration.
2. If $\alpha_2^1 = \alpha_1 - c_1\delta(-c_1)$ and $\beta_2^1 = \beta_2$, then condition (15) remains the same and condition (16) transforms to $c_1\delta(-c_1) + |c_2|(d_2 - d_1) \leq \beta_2 - \alpha_1$. Thus the algorithm converges in two iterations to W and $U(x)$ as in (18), but with α_2 replaced by $\alpha_1 - c_1\delta(-c_1)$.
3. If $\alpha_2^1 = \alpha_2$ and $\beta_2^1 = \beta_1 - c_1\delta(c_1)$, then condition (15) remains the same and condition (16) transforms to $c_1\delta(c_1) + |c_2|(d_2 - d_1) \leq \beta_1 - \alpha_2$. Thus the algorithm converges in two iterations to W and $U(x)$ as in (18), but with β_2 replaced by $\beta_1 - c_1\delta(c_1)$.
4. If $\alpha_2^1 = \alpha_1 - c_1\delta(-c_1)$ and $\beta_2^1 = \beta_1 - c_1\delta(c_1)$, then condition (15) remains the same and condition (16) transforms to $(|c_1| + |c_2|)(d_2 - d_1) \leq \beta_1 - \alpha_1$. Thus the algorithm converges in two iterations to W and $U(x)$ as in (18), but with α_2 replaced by $\alpha_1 - c_1\delta(-c_1)$ and β_2 replaced by $\beta_1 - c_1\delta(c_1)$.

This result can be summarized as follows:

Lemma 2 *Given system (13) with $F = [\alpha, \beta] = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2]$ and $D = [d_1, d_2] \subset \mathbb{R}$, the solution to the CIP problem, obtained after two iterations at most, is given by:*

$$W = \begin{cases} \{(x_1, x_2) : \alpha_1 \leq x_1 \leq \beta_1 \wedge \alpha_2^1 \leq x_2 \leq \beta_2^1\} & \text{if } \alpha_2^1 \leq \beta_2^1 \wedge |c_2|(d_2 - d_1) \leq \beta_2^1 - \alpha_2^1 \\ \emptyset & \text{otherwise} \end{cases} \quad (19)$$

$$U(x) = \begin{cases} \left\{ u : \alpha_2^1 - \sum_{j=1}^2 a_{2j}x_j - c_2\delta(-c_2) \leq u \leq \beta_2^1 - \sum_{j=1}^2 a_{2j}x_j - c_2\delta(c_2) \right\} & \text{if } x \in W \\ U & \text{otherwise} \end{cases} \quad (20)$$

2.3 n dimensional case

Here $F = [\alpha, \beta] = [\alpha_1, \beta_1] \times \dots \times [\alpha_n, \beta_n]$ and is (2) assumed to be

$$x(t+1) = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & & & & & 1 \\ a_{n1} & a_{n2} & \dots & & & a_{nn} \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} u(t) + \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} d(t) \quad (21)$$

Extending the definition of $\psi^1(\cdot)$ to $\mathbb{R}^n \times \mathbb{R} \times \mathbb{R}$ we get that $\psi^1(x, u, d)$ is equivalent to

$$\bigwedge_{j=1}^n (\alpha_j \leq x_j \leq \beta_j) \wedge \bigwedge_{j=2}^n (\alpha_{j-1} - c_{j-1}\delta(-c_{j-1}) \leq x_j \leq \beta_{j-1} - c_{j-1}\delta(c_{j-1})) \wedge \quad (22)$$

$$\left(\alpha_n - \sum_{j=1}^n a_{nj}x_j - c_n\delta(-c_n) \leq u \leq \beta_n - \sum_{j=1}^n a_{nj}x_j - c_n\delta(c_n) \right) \quad (23)$$

From the last expression, it is clear that given $x_1 \in [\alpha_1, \beta_1]$, $x_j, j = 2 \dots n$ exists if and only if

$$\alpha_j^1 = \max(\alpha_j, \alpha_{j-1} - c_{j-1}\delta(-c_{j-1})) \leq \min(\beta_j, \beta_{j-1} - c_{j-1}\delta(c_{j-1})) = \beta_j^1, \quad j = 2 \dots n \quad (24)$$

and u exists if and only if

$$\alpha_n - c_n\delta(-c_n) \leq \beta_n - c_n\delta(c_n) \quad (25)$$

As in the two dimensional case, after one iteration variables $x_j, j = 2 \dots n$ get restricted, which means that we have to run a second iteration. It is straightforward to see that in the l -th iteration ($l \leq n$) W is defined by:

$$W = [\alpha_1, \beta_1] \times [\alpha_2^1, \beta_2^1] \times \dots \times [\alpha_l^{l-1}, \beta_l^{l-1}] \times [\alpha_{l+1}^l, \beta_{l+1}^l] \times [\alpha_{l+2}^l, \beta_{l+2}^l] \times \dots \times [\alpha_n^l, \beta_n^l] \quad (26)$$

where

$$\alpha_j^l = \max(\alpha_j^{l-1}, \alpha_{j-1}^{l-1} - c_{j-1}\delta(c_{j-1})) \quad j \geq l+1 \quad (27)$$

$$\beta_j^l = \min(\beta_j^{l-1}, \beta_{j-1}^{l-1} - c_{j-1}\delta(c_{j-1})) \quad j \geq l+1 \quad (28)$$

This means that after n iterations, the maximal controlled invariant set remains unchanged, and the least restrictive controller is given by equation (23), but with α_j, β_j replaced by $\alpha_j^{j-1}, \beta_j^{j-1}, j = 2 \dots n$. This result can be summarized as follows:

Theorem 1 *Given system (21) with $F = [\alpha, \beta] = [\alpha_1, \beta_1] \times \dots \times [\alpha_n, \beta_n]$ and $D = [d_1, d_2] \subset \mathbb{R}$, the solution to the CIP problem, obtained after at most n iterations of the algorithm, is given by:*

$$W = \begin{cases} \left\{ (x \ni \bigwedge_{j=1}^n \alpha_j^{j-1} \leq x_j \leq \beta_j^{j-1}) \right\} & \text{if } \alpha_j^{j-1} \leq \beta_j^{j-1}, j = 1 \dots n \wedge |c_n|(d_2 - d_1) \leq \beta_n^{n-1} - \alpha_n^{n-1} \\ \emptyset & \text{otherwise} \end{cases} \quad (29)$$

$$U(x) = \begin{cases} \left\{ u \ni \alpha_n^{n-1} - \sum_{j=1}^n a_{nj}x_j - c_n\delta(-c_n) \leq u \leq \beta_n^{n-1} - \sum_{j=1}^n a_{nj}x_j - c_n\delta(c_n) \right\} & \text{if } x \in W \\ U & \text{otherwise} \end{cases} \quad (30)$$

Note that if at the first iteration, we compute

$$\alpha_1^1 = \alpha_1 \quad (31)$$

$$\beta_1^1 = \beta_1 \quad (32)$$

$$\alpha_j^1 = \max(\alpha_j, \alpha_{j-1}^1 - c_{j-1}\delta(c_{j-1})) \quad j \geq 2 \quad (33)$$

$$\beta_j^1 = \min(\beta_j, \beta_{j-1}^1 - c_{j-1}\delta(c_{j-1})) \quad j \geq 2 \quad (34)$$

then the problem can be solved in one iteration.

3 Linear inequality case

In this section, we consider the case where F is a convex polygon defined by $F = \{x \in \mathbb{R}^n \ni Mx \leq \beta\}$ where $M \in \mathbb{R}^{m \times n}$ and $\beta \in \mathbb{R}^m$ with m being the number of constrains.

Therefore

$$\psi^1(x, u, d) = MAx + Mbu + Mcd \leq \beta \quad (35)$$

$$= \hat{A}x + \hat{b}u + \hat{c}d \leq \beta \quad (36)$$

$$= \hat{b}u \leq \beta - \hat{A}x - \hat{c}d \quad (37)$$

$$= \hat{b}_i u \leq \beta_i - \hat{c}_i d - \hat{a}_i^T x \quad i = 1 \dots m \quad (38)$$

where \hat{a}_i^T is the i -th row of $\hat{A} = MA$.

Define

$$I^{1+} = \{p_i^1 \in \{1, \dots, m\} \ni \hat{b}_{p_i^1} > 0\},$$

$$I^{1-} = \{q_j^1 \in \{1, \dots, m\} \ni \hat{b}_{q_j^1} < 0\},$$

$$I^{10} = \{r_k^1 \in \{1, \dots, m\} \ni \hat{b}_{r_k^1} = 0\},$$

and let $p^1 = |I^{1+}|$, $q^1 = |I^{1-}|$ and $r^1 = |I^{10}|$. Then $m = p^1 + q^1 + r^1$. With these definitions $\psi^1(x, u, d)$ is equivalent to

$$\frac{1}{\hat{b}_{q_j^1}} (\beta_{q_j^1} - \hat{c}_{q_j^1} \delta(\hat{c}_{q_j^1}) - \hat{a}_{q_j^1}^T x) \leq u \leq \frac{1}{\hat{b}_{p_i^1}} (\beta_{p_i^1} - \hat{c}_{p_i^1} \delta(\hat{c}_{p_i^1}) - \hat{a}_{p_i^1}^T x) \quad i = 1 \dots p^1, j = 1 \dots q^1 \quad (39)$$

$$\wedge \quad \hat{a}_{r_k^1}^T x \leq \beta_{r_k^1} - \hat{c}_{r_k^1} \delta(\hat{c}_{r_k^1}) \quad k = 1 \dots r^1 \quad (40)$$

These equations give the following $p^1 q^1 + r^1$ constraints on x :

$$\begin{aligned} (\hat{b}_{p_i^1} - \hat{b}_{q_j^1}) \begin{pmatrix} \hat{a}_{q_j^1}^T \\ -\hat{a}_{p_i^1}^T \end{pmatrix} x &\leq (\hat{b}_{p_i^1} - \hat{b}_{q_j^1}) \begin{pmatrix} \beta_{q_j^1} \\ \beta_{p_i^1} \end{pmatrix} - (\hat{b}_{p_i^1} - \hat{b}_{q_j^1}) \begin{pmatrix} \hat{c}_{q_j^1} \delta(\hat{c}_{q_j^1}) \\ \hat{c}_{p_i^1} \delta(\hat{c}_{p_i^1}) \end{pmatrix} \quad \begin{matrix} i = 1 \dots p^1 \\ j = 1 \dots q^1 \end{matrix} \\ \hat{a}_{r_k^1}^T x &\leq \beta_{r_k^1} - \hat{c}_{r_k^1} \delta(\hat{c}_{r_k^1}) \quad k = 1 \dots r^1 \end{aligned} \quad (41)$$

Therefore, after the first iteration W^0 gets reduced by up to $p^1 q^1 + r^1$ new linear constraints.

Now we have to check the redundancy of each new constraint. In order to do that, let's define $M^l x \leq \beta^l$ as the set of linear constraints defining W^l after the l -th iteration. Clearly $M^0 = M$ and $\beta^0 = \beta$. Also define $\hat{A}^l = M^l A$, $\hat{b}^l = M^l b$, $\hat{c}^l = M^l c$, and I^{l+} , I^{l-} , I^{l0} , p_i^l , q_j^l , r_k^l , p^l , q^l and r^l , as usual, by replacing the 1's by l 's. Now let $g^{s'lT} x \leq \gamma^{s'}$ be the s^l -th new constraints (if one exists), $s^l = 1 \dots p^l q^l + r^l$. If this constraint is redundant, then $W^l \subset \{x \ni g^{s'lT} x \leq \gamma^{s'}\}$, or equivalently, $W^l \cap \{x \ni g^{s'lT} x > \gamma^{s'}\} = \emptyset$. In order to deal with intersection of sets, we solve the following linear programming problem:

$$\begin{aligned} \min \quad & f^{lT} x \\ \text{s.t.} \quad & M^l x \leq \beta^l \\ & g^{s'lT} x > \gamma^{s'} \end{aligned} \quad (42)$$

for any vector $f^l \neq 0$. If $\exists x^* = \text{argmin}(f^{lT} x)$, then the constraint is not redundant, otherwise it is.

However, linear programming algorithms usually don't manage open constraints such as $g^{s'lT} x > \gamma^{s'}$, and because of this, we are forced to consider the relaxed problem with $g^{s'lT} x \geq \gamma^{s'}$. This relaxation technique might, however, give a solution for x even though the new constraint is redundant (see Figure 1(a)). Nevertheless, is it straightforward to check that, in this case, the solution of the optimization problem satisfies $g^{s'lT} x^* = \gamma$. However it is not necessarily the case that if $g^{s'lT} x^* = \gamma$, then the new constraint is redundant as can be seen in Figure 1(b). This last problem can be solved by cleverly choosing vector $f^l = -g^{s'}$.

As a conclusion, the algorithm for checking redundancy is the following: at every iteration l and for each new constraint $g^{s'lT} x \leq \gamma^{s'}$ we solve the optimization problem:

$$\begin{aligned} \min \quad & -g^{s'lT} x \\ \text{s.t.} \quad & M^l x \leq \beta^l \\ & -g^{s'lT} x \leq -\gamma^{s'} \end{aligned} \quad (43)$$

If $\exists x^* = \text{argmin}(-g^{s'lT} x)$ satisfying $g^{s'lT} x^* > \gamma^{s'}$, then the constraint is not redundant, otherwise it is.

Then we define our new M and β by adding all non-redundant new constraints.

Finally, it is clear that if at some iteration l^* no new constraints are added, then we have found the controlled invariant set and the least restrictive controller as:

$$W = \{x \ni M^{l^*} x \leq \beta^{l^*}\} \quad (44)$$

$$U(x) = \left\{ u \ni \frac{1}{\hat{b}_{q_j}^{l^*}} \left(\beta_{q_j}^{l^*} - \hat{c}_{q_j}^{l^*} \delta(\hat{c}_{q_j}^{l^*}) - \hat{a}_{q_j}^{l^* T} x \right) \leq u \leq \frac{1}{\hat{b}_{p_i}^{l^*}} \left(\beta_{p_i}^{l^*} - \hat{c}_{p_i}^{l^*} \delta(\hat{c}_{p_i}^{l^*}) - \hat{a}_{p_i}^{l^* T} x \right) \right\} \quad (45)$$

for $i = 1 \dots p^{l^*}, j = 1 \dots q^{l^*}$.

Thus we have proposed an implementation of the algorithm to solve the CIP problem. However, we have not analyzed whether this implementation converges or not in a finite number of steps. In the following section, we will provide an example which actually converges after an infinite number of iterations. Thus we conclude the following.

Theorem 2 *Given system (2), with $F = \{x \ni Mx \leq \beta\}$ and $\mathbf{D} = [d_1, d_2] \subset \mathbb{R}$, the CIP problem is semi-decidable.*

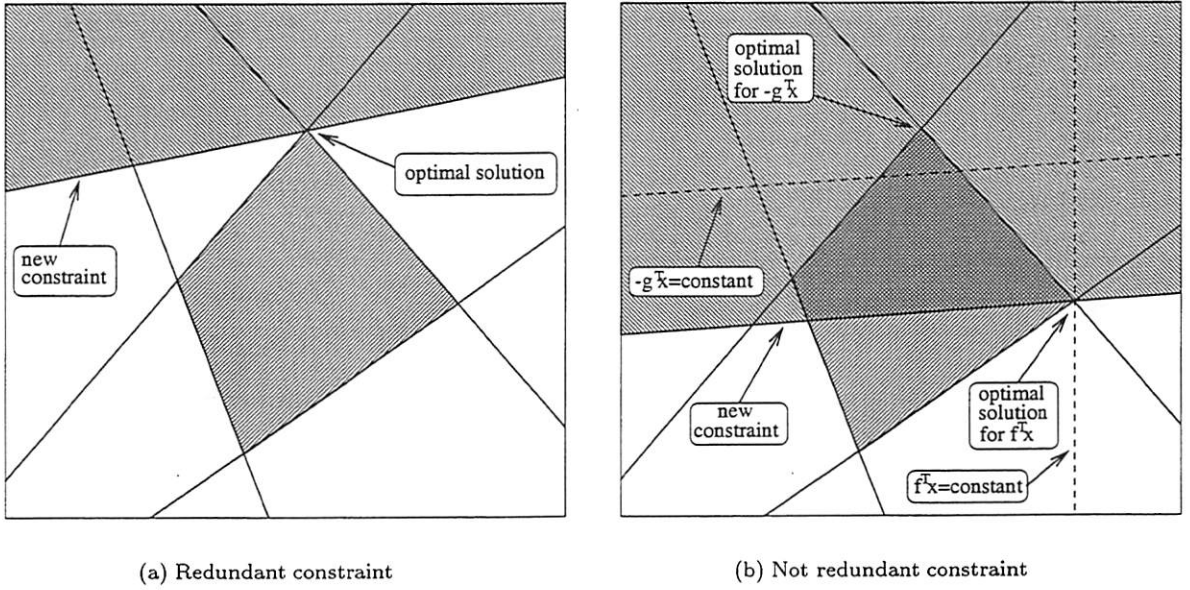


Figure 1: Checking the redundancy of new constraints

4 Experimental results

In this section, we present two examples to test the the MATLAB implementation of the algorithm proposed in the previous section. Both examples are worked out analytically and the results are compared to that of the MATLAB program. Both examples use the same discrete-time linear system, but with a different F set. The linear system we consider is

$$x(t+1) = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} d(t) \quad (46)$$

with $d(t) \in \mathbf{D} = [-1, 1]$.

4.1 Example 1

In this example, F is defined as $F = \{x \mid Mx \leq \beta\}$ with

$$M = \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix} \quad \beta = \begin{pmatrix} 80 \\ 40 \\ 80 \\ 70 \end{pmatrix}$$

1. Initialization $M^0 = M, \beta^0 = \beta$.

2. Iteration 1

(a) Computing $\hat{A}, \hat{b}, \hat{c}$

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -1 & -2 \\ -1 & 0 \\ 1 & 0 \end{pmatrix} \quad \hat{b} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \quad \hat{c} = \begin{pmatrix} 2 \\ -2 \\ 0 \\ 0 \end{pmatrix}$$

(b) Computing possible new constraints. Here $I^{1+} = \{1, 4\}$, $I^{1-} = \{2, 3\}$ and $I^{1^0} = \emptyset$

$$\begin{aligned} \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & -2 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 40 \\ 80 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \\ &\Rightarrow 0 \leq 116 \Rightarrow \text{Redundant} \\ \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 80 \\ 80 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ &\Rightarrow 2x_2 \leq 158 \Rightarrow \text{Redundant} \\ \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & -2 \\ 1 & 0 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 40 \\ 70 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \end{pmatrix} \\ &\Rightarrow -2x_2 \leq 108 \Rightarrow \text{Not Redundant} \\ \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & -2 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 40 \\ 80 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \\ &\Rightarrow 0 \leq 150 \Rightarrow \text{Redundant} \end{aligned}$$

(c) Computing new M and β

$$M^1 = \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 0 & -2 \end{pmatrix} \quad \beta^1 = \begin{pmatrix} 80 \\ 40 \\ 80 \\ 70 \\ 108 \end{pmatrix}$$

3. Iteration 2

(a) Computing $\hat{A}, \hat{b}, \hat{c}$

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -1 & -2 \\ -1 & 0 \\ 1 & 0 \\ -2 & -2 \end{pmatrix} \quad \hat{b} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -2 \end{pmatrix} \quad \hat{c} = \begin{pmatrix} 2 \\ -2 \\ 0 \\ 0 \\ -2 \end{pmatrix}$$

(b) Computing possible new constraints. Here $I^{2+} = \{1, 4\}$, $I^{2-} = \{5\}$ and $I^{2^0} = \emptyset$

$$\begin{aligned} \begin{pmatrix} 1 & 2 \end{pmatrix} \begin{pmatrix} -2 & -2 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 2 \end{pmatrix} \begin{pmatrix} 108 \\ 80 \end{pmatrix} - \begin{pmatrix} 1 & 2 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \\ &\Rightarrow 2x_2 \leq 266 \Rightarrow \text{Redundant} \\ \begin{pmatrix} 1 & 2 \end{pmatrix} \begin{pmatrix} -2 & 2 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 108 \\ 70 \end{pmatrix} - \begin{pmatrix} 1 & 2 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \end{pmatrix} \\ &\Rightarrow -2x_2 \leq 246 \Rightarrow \text{Redundant} \end{aligned}$$

(c) Therefore W and $U(x)$ converge to

$$\begin{aligned} W &= \left\{ x \ni \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 0 & -2 \end{pmatrix} x \leq \begin{pmatrix} 80 \\ 40 \\ 80 \\ 70 \\ 108 \end{pmatrix} \right\} \\ U(x) &= \begin{cases} \{u \in U \ni u \geq \max(-38 - x_1 - 2x_2, -80 - x_1, -52 - x_1 - x_2) \\ u \leq \min(78 - x_1 - 2x_2, 70 - x_1)\} \\ U \end{cases} \quad \begin{array}{l} \text{if } x \in W \\ \text{otherwise} \end{array} \end{aligned}$$

Figure 2 shows the plot obtained by MATLAB.

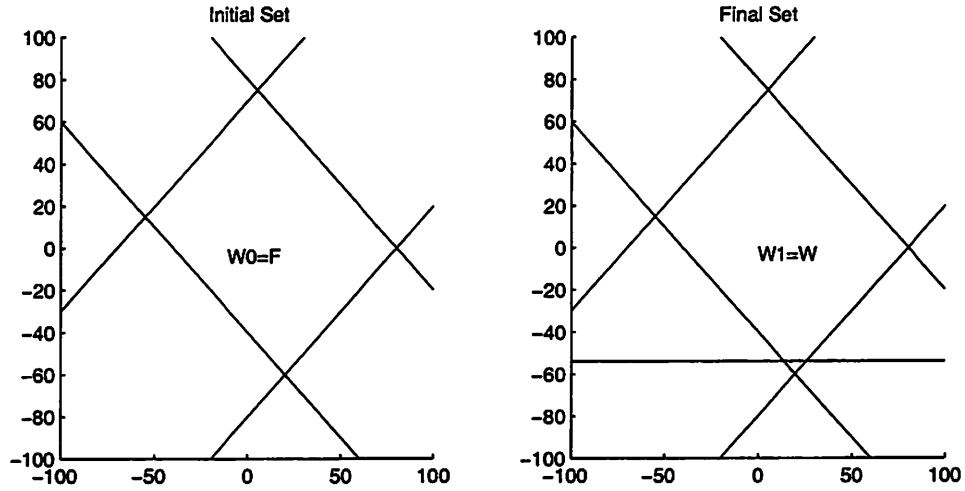


Figure 2: Iterations of the algorithm for Example 1

4.2 Example 2

In this example, F is defined as $F = \{x \ni Mx \leq \beta\}$ with

$$M = \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & 1 \end{pmatrix} \quad \beta = \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \end{pmatrix}$$

1. Initialization $M^0 = M$, $\beta^0 = \beta$.

2. Iteration 1

(a) Computing \hat{A} , \hat{b} , \hat{c}

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -3 & -4 \\ -1 & 0 \\ 1 & -2 \end{pmatrix} \quad \hat{b} = \begin{pmatrix} 1 \\ -3 \\ -1 \\ 1 \end{pmatrix} \quad \hat{c} = \begin{pmatrix} 2 \\ -4 \\ 0 \\ -2 \end{pmatrix}$$

(b) Computing possible new constraints. Here $I^{1+} = \{1, 4\}$, $I^{1-} = \{2, 3\}$ and $I^{10} = \emptyset$

$$\begin{aligned} \begin{pmatrix} 1 & 3 \end{pmatrix} \begin{pmatrix} -3 & -4 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 3 \end{pmatrix} \begin{pmatrix} -50 \\ 100 \end{pmatrix} - \begin{pmatrix} 1 & 3 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \end{pmatrix} \\ &\Rightarrow 2x_2 \leq 240 \Rightarrow \text{Redundant} \\ \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 100 \\ 100 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ &\Rightarrow 2x_2 \leq 198 \Rightarrow \text{Redundant} \\ \begin{pmatrix} 1 & 3 \end{pmatrix} \begin{pmatrix} -3 & -4 \\ 1 & -2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 3 \end{pmatrix} \begin{pmatrix} -50 \\ -50 \end{pmatrix} - \begin{pmatrix} 1 & 3 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \end{pmatrix} \\ &\Rightarrow -10x_2 \leq -210 \Rightarrow \text{Not Redundant} \\ \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 1 & -2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 100 \\ -50 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ &\Rightarrow -2x_2 \leq 48 \Rightarrow \text{Redundant} \end{aligned}$$

(c) Computing new M and β

$$M^1 = \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & -1 \\ 0 & -10 \end{pmatrix} \quad \beta^1 = \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \\ -210 \end{pmatrix}$$

3. Iteration 2

(a) Computing \hat{A} , \hat{b} , \hat{c}

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -3 & -4 \\ -1 & 0 \\ 1 & -2 \\ -10 & -10 \end{pmatrix} \quad \hat{b} = \begin{pmatrix} 1 \\ -3 \\ -1 \\ 1 \\ -10 \end{pmatrix} \quad \hat{c} = \begin{pmatrix} 2 \\ -4 \\ 0 \\ -2 \\ -10 \end{pmatrix}$$

(b) Computing possible new constraints. Here $I^{2+} = \{1, 4\}$, $I^{2-} = \{5\}$ and $I^{20} = \emptyset$

$$\begin{aligned} \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} -10 & -10 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} -210 \\ 100 \end{pmatrix} - \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} 10 \\ 2 \end{pmatrix} \\ &\Rightarrow 10x_2 \leq 760 \Rightarrow \text{Redundant} \\ \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} -10 & -10 \\ 1 & -2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} -210 \\ -50 \end{pmatrix} - \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} 10 \\ 2 \end{pmatrix} \\ &\Rightarrow -30x_2 \leq -740 \Rightarrow \text{Not redundant} \end{aligned}$$

(c) Computing new M and β

$$M^2 = \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & -1 \\ 0 & -10 \\ 0 & -30 \end{pmatrix} \quad \beta^2 = \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \\ -210 \\ -740 \end{pmatrix}$$

4. Iteration l : note that when adding the new non-redundant constraint, the one added in the previous iteration becomes redundant, but the algorithm is not checking for that.

(a) Updated M and β from iteration $l-1$.

$$M^{l-1} = \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & -1 \\ 0 & -10 \\ 0 & -30 \\ \vdots & \vdots \\ 0 & m_l \end{pmatrix} \quad \beta^{l-1} = \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \\ -210 \\ -740 \\ \vdots \\ \beta_l \end{pmatrix}$$

(b) Computing \hat{A} , \hat{b} , \hat{c}

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -3 & -4 \\ -1 & 0 \\ 1 & -2 \\ -10 & -10 \\ -30 & -30 \\ \vdots & \vdots \\ m_l & m_l \end{pmatrix} \quad \hat{b} = \begin{pmatrix} 1 \\ -3 \\ -1 \\ 1 \\ -10 \\ -30 \\ \vdots \\ m_l \end{pmatrix} \quad \hat{c} = \begin{pmatrix} 2 \\ -4 \\ 0 \\ -2 \\ -210 \\ -740 \\ \vdots \\ m_l \end{pmatrix}$$

(c) Computing possible new constraints. Here $I^{l+} = \{1, 4\}$, $I^{l-} = \{l+3\}$ and $I^{l^0} = \emptyset$

$$\begin{aligned} \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} m_l & m_l \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} \beta_l \\ 100 \end{pmatrix} - \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} -m_l \\ 2 \end{pmatrix} \\ &\Rightarrow -m_l x_2 \leq \beta_l - 97m_l \\ \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} m_l & m_l \\ 1 & -2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} \beta_l \\ -50 \end{pmatrix} - \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} -m_l \\ 2 \end{pmatrix} \\ &\Rightarrow 3m_l x_2 \leq \beta_l + 53m_l \end{aligned}$$

In order to check for both redundancy and convergence of the constraint on x_2 , we consider the second constraint first. We have $m_{l+1} = 3m_l$, $m_1 = -10$, and $\beta_{l+1} = \beta_l + 53m_l$, $\beta_1 = -210$. Thus $m_l = m_1 3^{l-1}$, and $\beta_l = \beta_1 + 53m_1(3^{l-1} - 1)/2$. Thus the second constraint becomes $x_2 \geq 26.5 - 5.5/3^l$. This means that if the first constraint is redundant $\forall l$, then the second constraint is always not redundant and converges after an infinite number of steps to $x_2 \geq 26.5$. Thus the only thing we need to check is the redundancy of the first constraint, which is $x_2 \leq 70.5 + 5.53^{l-1}$. Since the top vertex of W has $x_2 = 62.5$, the first constraint is redundant $\forall l$, and converges to $x_2 \leq 70.5$.

(d) Therefore after an infinite number of iterations, W and $U(x)$ converge to

$$W = \left\{ x \in \mathbb{R}^2 : \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & -1 \\ 0 & -2 \end{pmatrix} x \leq \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \\ -53 \end{pmatrix} \right\}$$

$$U(x) = \begin{cases} \{u \in \mathbb{U} : u \geq \max(18 - x_1 - 4x_2/3, -100 - x_1, -55/2 - x_1 - x_2) \\ \quad u \leq \min(98 - x_1 - 1 - 2x_2, -52 - x_1 + 2x_2)\} & \text{if } x \in W \\ \mathbb{U} & \text{otherwise} \end{cases}$$

Figure 3 shows the plot obtained by MATLAB for the first three iterations.

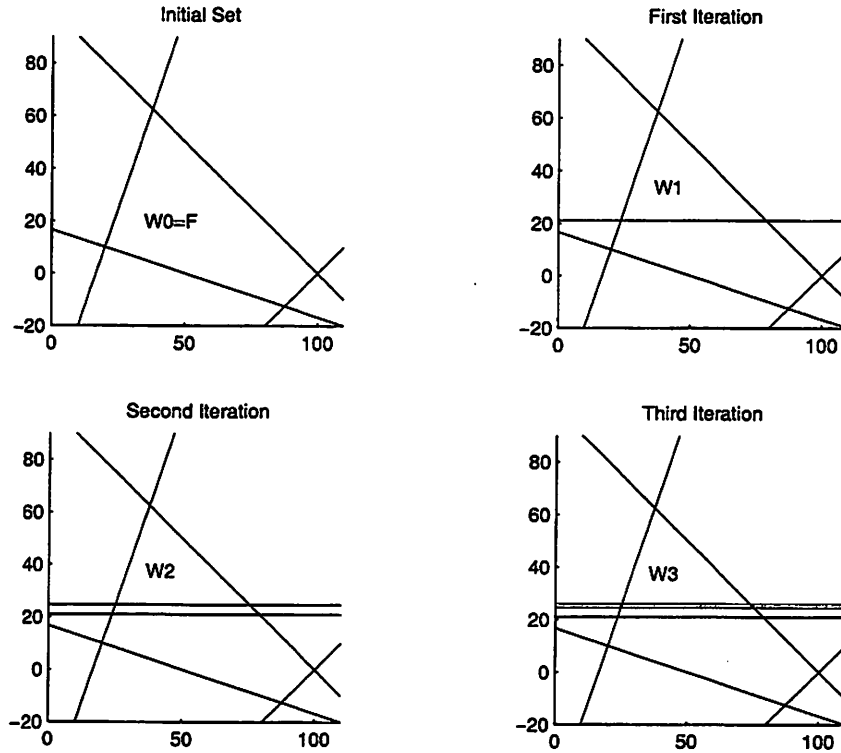


Figure 3: Iterations of the algorithm for Example 2

5 Conclusions and Future Work

In this project, an algorithm for computing the maximal controlled invariant set and least restrictive controller for discrete-time linear systems with a scalar input and disturbance was developed, and the decidability of the problem was analyzed. It was proved that if the system is in canonically controllable form and the initial set is a rectangle, the problem is decidable. It was also shown, that for a general discrete-time linear system and an initial set specified by a set of linear inequalities, the problem is semi-decidable. In the future, it would be interesting to extend these results to systems with multiple inputs and disturbances. It would also be interesting to determine under which conditions on the system the problem is decidable. So far, it seems that the decidability property is not only dependent on the system itself, but also on the initial set, as shown by Example 2.

EE291 Final Project

Properties of Zeno Executions

Jun Zhang

May 11, 1999

1 Hybrid Automaton

In this section we present the basic background material.

For a finite collection V of variables, let V denote the set of valuations of these variables. We use lower case letters to denote both a variable and its valuation. We refer to variables whose set of valuations is finite as *discrete* and to variables whose set of valuations is a subset of a Euclidean space as *continuous*. We assume that Euclidean spaces, \mathbb{R}^n for $n \geq 0$, are given the Euclidean metric topology, whereas countable and finite sets are given the discrete topology (every subset is an open set). Subsets of a topological space are given the induced topology and products of topological spaces are given the product topology. The Euclidean norm is denoted $\|\cdot\|$ and the convex hull $\text{co}\{\cdot\}$. For a subset U of a topological space we use \bar{U} to denote its closure, U° its interior, ∂U its boundary, U^c its complement, $|U|$ its cardinality, and 2^U the set of all subsets of U .

Definition 1 (Hybrid Automaton)

A hybrid automaton H is a collection $H = (Q, X, \text{Init}, f, I, E, G, R)$, where

- Q is a finite collection of discrete variables with $|Q| < \infty$;
- X is a finite collection of continuous variables with $X = \mathbb{R}^n$;
- $\text{Init} \subseteq Q \times X$ is a set of initial states;
- $f : Q \times X \rightarrow TX$ is a vector field, Lipschitz continuous in its second argument;
- $I : Q \rightarrow 2^X$ assigns to each $q \in Q$ an invariant set;
- $E \subseteq Q \times Q$ is a collection of edges;
- $G : E \rightarrow 2^X$ assigns to each edge $e = (q, q') \in E$ a guard; and
- $R : E \times X \rightarrow 2^X$ assigns to each edge $e = (q, q') \in E$ and $x \in X$ a reset relation.

We refer to $(q, x) \in Q \times X$ as the state of H . Pictorially, a hybrid automaton can be represented by a digraph (Q, E) , with vertices Q and edges E . With each vertex $q \in Q$, we associate a set of continuous initial states $\text{Init}_q = \{x \in X : (q, x) \in \text{Init}\}$, a vector field $f(q, x)$, and an invariant set $I(q)$. With each edge $e \in E$, we associate a guard $G(e)$ and a reset relation $R(e, x)$. In the following part, we assume that the vector field is Lipschitz continuous unless otherwise stated.

Definition 2 (Hybrid Time Trajectory)

A hybrid time trajectory $\tau = \{I_i\}_{i=0}^N$ is a finite or infinite sequence of intervals of the real line, such that

- $I_i = [\tau_i, \tau'_i]$ for $i < N$;
- $\tau_i \leq \tau'_i$ for $i \geq 0$ and $\tau_i = \tau'_{i-1}$ for $i > 0$; and
- $I_N = [\tau_N, \tau'_N]$ or $I_N = [\tau_N, \tau'_N)$ if $N < \infty$.

Note that hybrid time trajectories can extend to infinity if τ is an infinite sequence or if it is a finite sequence ending with an interval of the form $[\tau_N, \infty)$. We denote by \mathcal{T} the set of all hybrid time trajectories. Basically, we will study the nontrivial hybrid automaton whose time trajectory τ has positive measure, otherwise, it can be investigated as a Finite State machine.

Definition 3 (Execution)

An execution χ of a hybrid automaton H is a collection $\chi = (\tau, q, x)$ with $\tau \in \mathcal{T}$, $q : \tau \rightarrow \mathbf{Q}$, and $x : \tau \rightarrow \mathbf{X}$, satisfying

- $(q(\tau_0), x(\tau_0)) \in \text{Init}$ (initial condition);
- for all i with $\tau_i < \tau'_i$, $x(t)$ is absolutely differentiable and $q(t)$ is constant for $t \in [\tau_i, \tau'_i]$, and $x(t) \in I(q(t))$ and $dx(t)/dt = f(q(t), x(t))$ for all $t \in [\tau_i, \tau'_i]$ (continuous evolution); and
- for all i , $e = (q(\tau'_i), q(\tau_{i+1})) \in E$, $x(\tau'_i) \in G(e)$, and $x(\tau_{i+1}) \in R(e, x(\tau'_i))$ (discrete evolution).

We say a hybrid automaton *admits* an execution χ . For an execution $\chi = (\tau, q, x)$, we use $(q_0, x_0) = (q(\tau_0), x(\tau_0))$ to denote the initial state of χ . An execution is *finite* if τ is finite sequence ending with a closed interval; it is called *infinite* if it is either an infinite sequence or if $\sum_i (\tau'_i - \tau_i) = \infty$.

Definition 4 (Non-Blocking and Deterministic automaton)

A hybrid automaton H is called *non-blocking* if it accepts at least one infinite executions for all $(q_0, x_0) \in \text{Init}$. It is called *deterministic* if it accepts at most one infinite execution for all $(q_0, x_0) \in \text{Init}$.

Definition 5 (Reachable State)

A state $(\hat{q}, \hat{x}) \in \mathbf{Q} \times \mathbf{X}$ is called *reachable* by H , if there exists a finite execution $\chi = (\tau, q, x)$ with $\tau = \{[\tau_i, \tau'_i]\}_{i=0}^N$ and $(q(\tau'_N), x(\tau'_N)) = (\hat{q}, \hat{x})$.

The set of states reachable by H is denoted $\text{Reach}(H) \subseteq \mathbf{Q} \times \mathbf{X}$.

A hybrid automaton is said to be *invariant preserving* if $x(t) \in \overline{I(q(t))}$ for all executions $\chi = (\tau, q, x)$ and for all $t > 0$.

Proposition 1

A hybrid automaton is invariant preserving if

- $x_0 \in \overline{I(q_0)}$ for all $(q_0, x_0) \in \text{Init}$; or after finite discrete transitions, there exists some N such that $x_{\tau_N} \in \overline{I(q_{\tau_N})}$
- $R(q, q', x) \subseteq \overline{I(q')}$ for all $(q, q') \in E$ and $x \in G(q, q') \cap \overline{I(q)}$.

For proof, see Karl's paper.

Consider a hybrid automaton with $f : \mathbf{Q} \times \mathbf{X} \rightarrow T\mathbf{X}$ Lipschitz continuous in its second argument, then for every $q_i \in \mathbf{Q}$, $x \in \mathbb{R}^n$, there exists some $c_i > 0$ such that

$$\|f(q_i, x)\| \leq c_i(\|x\| + 1) \leq c(\|x\| + 1),$$

where $c = \max c_i$. We say the reset relation R is *non-expanding* if there exists some $\delta \in [0, 1]$ such that for all $e = (q, q') \in E$, all $x \in G(e)$, all $x' \in R(e, x)$,

$$\|x'\| \leq \delta \|x\|.$$

In particular, non-expanding implies $R(e, 0) = 0$. When $0 \leq \delta < 1$, we say that R is *contracting*.

In the continuous systems, the Lipschitz continuity assumption on the vector field excludes the possibility of finite escape time. For the hybrid systems, we have the similar results as follows.

Lemma 1 (Bellman-Gronwall, Shankar's book)

Let $z(\cdot)$, $a(\cdot)$, $u(\cdot) : \mathbb{R}_+ \rightarrow \mathbb{R}$ be given positive functions defined on $[0, \infty)$ and let $T > t_0 \geq 0$, then if for all $t \in [t_0, T]$,

$$z(t) \leq u(t) + \int_{t_0}^t a(\tau) z(\tau) d\tau,$$

we have that for $t \in [t_0, T]$

$$z(t) \leq u(t) + \int_{t_0}^t a(\tau) u(\tau) e^{\int_{\tau}^t a(\sigma) d\sigma} d\tau.$$

Proposition 2

Given a hybrid automaton with non-expanding reset, for every execution $\chi = (\tau, q, x)$, there exists some $c > 0$ such that for all $t \in \tau$,

$$\|x(t)\| \leq (\|x(\tau_0)\| + 1)e^{c(t-\tau_0)} - 1.$$

Proof: For $\tau_i \leq t \leq \tau'_i$,

$$x(t) = x(\tau_i) + \int_{\tau_i}^t f(q(\tau_i), x(\tau)) d\tau,$$

then, using the triangular inequality,

$$\|x(t)\| \leq \|x(\tau_i)\| + \int_{\tau_i}^t \|f(q(\tau_i), x(\tau))\| d\tau.$$

There exists some $c > 0$ such that

$$\|f(q(\tau_i), x(\tau))\| \leq c(\|x(\tau)\| + 1).$$

Thus, it follows that

$$\|x(t)\| \leq \|x(\tau_i)\| + \int_{\tau_i}^t c(\|x(\tau)\| + 1) d\tau.$$

Let $z(t) = \|x(t)\| + 1$, $u(t) = \|x(\tau_i)\| + 1$, $a(t) = c$. Applying Bellman-Gronwall lemma, we have

$$\|x(t)\| + 1 \leq (\|x(\tau_i)\| + 1)e^{c(t-\tau_i)}, \quad t \in [\tau_i, \tau'_i].$$

Since by non-expanding assumption, $\|x(\tau_i)\| \leq \|x(\tau'_{i-1})\|$, we deduce that

$$\begin{aligned} \|x(t)\| + 1 &\leq (\|x(\tau'_{i-1})\| + 1)e^{c(t-\tau_i)} \\ &\leq (\|x(\tau_{i-1})\| + 1)e^{c(\tau'_{i-1}-\tau_{i-1})}e^{c(t-\tau_i)}. \end{aligned}$$

Proceeding further,

$$\|x(t)\| + 1 \leq (\|x(\tau_0)\| + 1)e^{c(t-\tau_0)},$$

that is,

$$\|x(t)\| \leq (\|x(\tau_0)\| + 1)e^{c(t-\tau_0)} - 1.$$

■

Strictly speaking, the continuous trajectory of a hybrid automaton execution is not a function of time, since it takes more than one value at switch time $t = \tau'_i$. However, on every closed interval $[\tau_i, \tau'_i]$, it is a continuous differentiable function. Therefore we can treat $x(\cdot)$ as a function.

2 Zeno Hybrid Automaton

In this section we discuss the Zeno hybrid automaton.

Definition 6 (Zeno Hybrid Automaton)

An infinite execution is called Zeno if $\sum_{i=0}^{\infty}(\tau'_i - \tau_i)$ is bounded. The time $\tau_{\infty} = \sum_{i=0}^{\infty}(\tau'_i - \tau_i)$ is the Zeno time. If all executions are infinite and Zeno for some initial state, then the automaton is called a Zeno hybrid automaton.

Now we will present some particular features of Zeno Executions. First we introduce the notion of Zeno state.

Definition 7 (Zeno State)

A state $(\hat{q}, \hat{x}) \in \mathbf{Q} \times \mathbf{X}$ is called a Zeno state of a Zeno execution $\chi = (\tau, q, x)$, if there exists a sequence $\{\theta_i\}_{i=0}^{\infty}$, $\theta_i \in [\tau_i, \tau'_i]$, for all $N > 0$ and $\epsilon > 0$, $q(\theta_i) = \hat{q}$ and $\|x(\theta_i) - \hat{x}\| < \epsilon$ for some $i > N$.

In other words, the set of Zeno states consists of all cluster points of sequence $\{(q(\theta_i), x(\theta_i))\}_{i=0}^{\infty}$. The discrete part of the Zeno state will be visited infinitely often. We use $Z_{\infty} \subset \mathbf{Q} \times \mathbf{X}$ to denote the set of Zeno states.

Example 1

- $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}^2$;
- $\text{Init} = \{q_1\} \times \{x \in \mathbb{R}^2 : 0 \leq x_1 < 3\}$;
- $f(q, x) = (x_2, -10)^T$, for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$;
- $I(q_1) = \{x \in \mathbf{X} : x_1 \geq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : x_1 \geq 3\}$;
- $E = \{(q_1, q_2), (q_2, q_2), (q_2, q_1)\}$;

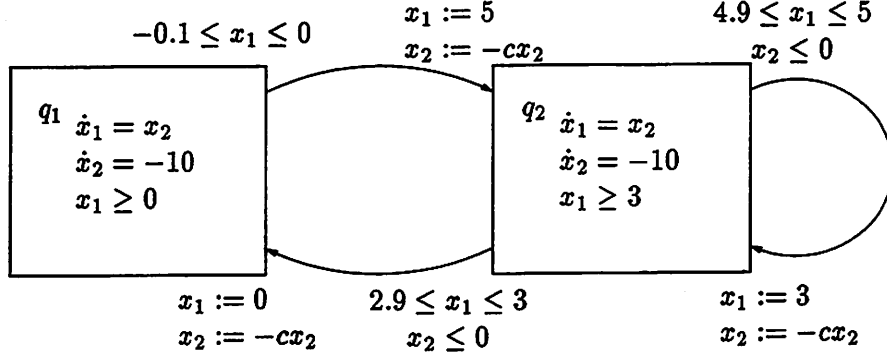


Figure 1: Example 1

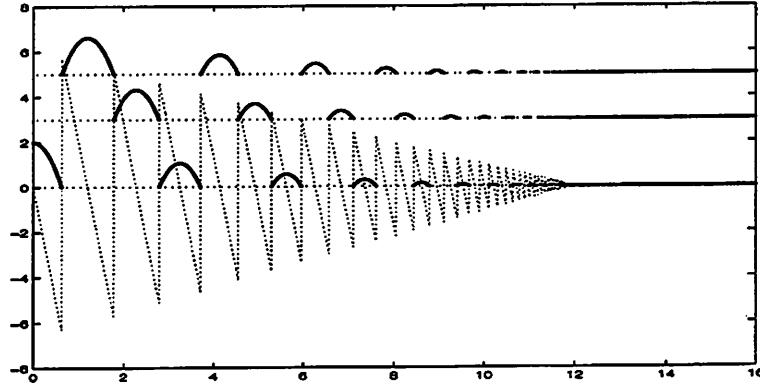


Figure 2: Simulation for Example 1: x_1 solid, x_2 dotted

- $G(q_1, q_2) = \{x \in \mathbf{X} : -0.1 \leq x_1 \leq 0\}$, $G(q_2, q_2) = \{x \in \mathbf{X} : 4.9 \leq x_1 \leq 5 \wedge x_2 \leq 0\}$,
 $G(q_2, q_1) = \{x \in \mathbf{X} : 2.9 \leq x_1 \leq 3 \wedge x_2 \leq 0\}$;
 - $R(q_1, q_2, x) = (5, -cx_2)^T$, $R(q_2, q_2, x) = (3, -cx_2)^T$, $R(q_2, q_1, x) = (0, -cx_2)^T$, for some $c \in (0, 1)$.
- The Zeno states are $\{q_1, (0, 0)^T\}$, $\{q_2, (3, 0)^T\}$ and $\{q_2, (5, 0)^T\}$.

Example 2

Consider a Zeno execution with $Z_\infty = \{(\hat{q}, \hat{x})\}$, modify this hybrid automaton by extending two extra continuous states (x_e, x_f) with

$$\begin{aligned}\dot{x}_e &= 0, \\ \dot{x}_f &= 0,\end{aligned}$$

and reset maps

$$\begin{pmatrix} x_e \\ x_f \end{pmatrix} := \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_e \\ x_f \end{pmatrix}$$

where $\theta/2\pi$ is irrational, and the initial condition $x_e(0) = 0$, $x_f(0) = 0$. Then the Zeno state set is

$$Z_\infty = \{\hat{q}\} \times \{(\hat{x}, x_e, x_f) : (x_e, x_f)^T \in S^1\}.$$

The above examples illustrate that the Zeno state set of could be either a finite set or an uncountable set. And also, one discrete Zeno state may correspond to multiple continuous Zeno states.

A Zeno hybrid automaton may have Zeno executions with no Zeno states. Here we will give an example with non-Lipschitz continuous vector field.

Example 3

Consider the hybrid automaton H , defined by

- $\mathbf{Q} = \{q_1, q_2\}$ and $\mathbf{X} = \mathbb{R}^2$;
- $\text{Init} = \mathbf{Q} \times \mathbf{X}$;
- $f(q, x) = (1, x_2^2)^T$, for all $(q, x) \in \mathbf{Q} \times \mathbf{X}$;
- $I(q_1) = \{x \in \mathbf{X} : x_1 \sin \frac{1}{|x_1|} \leq 0\}$ and $I(q_2) = \{x \in \mathbf{X} : x_1 \sin \frac{1}{|x_1|} \geq 0\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbf{X} : x_1 \sin \frac{1}{|x_1|} \geq 0\}$ and $G(q_2, q_1) = \{x \in \mathbf{X} : x_1 \sin \frac{1}{|x_1|} \leq 0\}$;
- $R(q_1, q_2, x) = R(q_2, q_1, x) = \{x\}$.

The execution of H with initial state $(q_1, (-1, 1)^T)$ exhibits an infinite number of discrete transitions by $\tau_\infty = 1$ and $x_2(t) = 1/(1-t)$, for all $t \in [0, \tau_\infty)$. However, for all $\{\theta_i\}_{i=0}^\infty$, $\theta_i \in [\tau_i, \tau_i']$, the sequence $\{x_2(\theta_i)\}_{i=0}^\infty$ is strictly monotonic increasing and unbounded. Therefore, H has no Zeno state.

Example 4

Similar to example 2, if we have a Zeno execution with $Z_\infty = \{(\hat{q}, \hat{x})\}$, augment an extra continuous state x_e with trivial continuous dynamics $\dot{x}_e = 0$, reset map $x_e := 2x_e$ and the initial condition $x_e(0) = 1$. The iteration of this map gives $x_e(\theta_i) = 2^i$, for all $\theta_i \in [\tau_i, \tau_i']$. Evidently, the modified hybrid automaton has no Zeno state.

In example 3, the Zeno execution has no Zeno state because f is not Lipschitz continuous, whereas in example 4, it is due to the exploding reset map. It is clear that the continuous dynamics and the reset map are crucial to the Zeno state.

When $x(\cdot)$ is bounded, Weierstrass theorem says that $\{q(\theta_i), x(\theta_i)\}_{i=0}^\infty$, $\theta_i \in [\tau_i, \tau_i']$, has at least one cluster point. Thus if the Zeno hybrid automaton satisfies the condition in Proposition 2, it has at least one Zeno state.

Proposition 3

Consider a Zeno hybrid automaton with $R(q, q', x) = \{x\}$ for all $(q, q') \in E$, for every Zeno execution $\chi = (\tau, q, x)$, it holds that $Z_\infty = Q_\infty \times \{\hat{x}\}$ for some $Q_\infty \subseteq \mathbf{Q}$ and $\hat{x} \in \mathbf{X}$.

Proof: Since the reset is non-expanding, from Proposition 2, for Zeno execution $\chi = (\tau, q, x)$, $\forall t \in [\tau_0, \tau_\infty)$, we have

$$\|x(t)\| \leq (\|x(\tau_0)\| + 1)e^{c(t-\tau_0)} - 1.$$

Then it is clear that $x(t)$ is bounded on $[\tau_0, \tau_\infty]$, so for all $q_i \in \mathbf{Q}$, all $t \in [\tau_0, \tau_\infty]$, there exists some $K_i > 0$ such that $\|f(q_i, x(t))\| \leq K_i$. Let $K = \max K_i$, for all $\{\theta_i\}_{i=0}^\infty$, $\theta_i \in [\tau_i, \tau'_i]$, it follows that

$$\begin{aligned} x(\theta_i) &= x(\tau_i) + \int_{\tau_i}^{\theta_i} f(q(\tau_i), x(\tau)) d\tau \\ &= x(\tau_i) + (\theta_i - \tau_i) f(q(\tau_i), x(\xi_i)), \end{aligned}$$

for some $\xi_i \in [\tau_i, \tau'_i]$. Hence for all $k > l \geq 0$,

$$\begin{aligned} x(\theta_k) &= x(\theta_l) + (\tau'_l - \theta_l) f(q(\tau_l), x(\xi_l)) \\ &\quad + \sum_{i=l+1}^{k-1} (\tau'_i - \tau_i) f(q(\tau_i), x(\xi_i)) \\ &\quad + (\theta_k - \tau_k) f(q(\tau_k), x(\xi_k)), \end{aligned}$$

which gives that

$$\|x(\theta_k) - x(\theta_l)\| \leq K \sum_{i=l}^k (\tau'_i - \tau_i).$$

Since $\sum_{i=0}^\infty (\tau'_i - \tau_i) < \infty$, it follows that for all $\epsilon > 0$ there exists $M > 0$, such that $k, l > M$ implies that $\|x(\theta_k) - x(\theta_l)\| < \epsilon$. Hence, $\{x(\theta_i)\}_{i=0}^\infty$ is a Cauchy sequence. The space $\mathbf{X} = \mathbb{R}^n$ is complete, so the sequence has a limit $\hat{x} = \lim_{i \rightarrow \infty} x(\theta_i)$. ■

The merit of the above proposition is that as far as R is non-expanding, the continuous part of Zeno states set is a singleton. Furthermore, if R is contracting, we can prove that the singleton is just $\{0\}$.

Proposition 4

Consider a Zeno hybrid automaton with contracting reset, for every Zeno execution $\chi = (\tau, q, x)$, it holds that $Z_\infty = Q_\infty \times \{0\}$ for some $Q_\infty \subseteq \mathbf{Q}$.

Proof: Same as above, for all $q_i \in \mathbf{Q}$, $f(q_i, x(t))$ is bounded by some $K > 0$. For all $\{\theta_i\}_{i=0}^\infty$, $\theta_i \in [\tau_i, \tau'_i]$, we have

$$\begin{aligned} \|x(\theta_i)\| &\leq \|x(\tau_i)\| + \left\| \int_{\tau_i}^{\theta_i} f(q(\tau_i), x(\tau)) d\tau \right\| \\ &\leq \|x(\tau_i)\| + K(\tau'_i - \tau_i). \end{aligned}$$

Using the fact that $\|x(\tau_i)\| \leq \delta \|x(\tau'_{i-1})\|$, it follows that

$$\begin{aligned} \|x(\theta_i)\| &\leq \delta \|x(\tau'_{i-1})\| + K(\tau'_i - \tau_i) \\ &= \delta \|x(\tau_{i-1}) + \int_{\tau_{i-1}}^{\tau'_{i-1}} f(q(\tau_{i-1}), x(\tau)) d\tau\| + K(\tau'_i - \tau_i) \\ &\leq \delta \|x(\tau_{i-1})\| + K\delta(\tau'_{i-1} - \tau_{i-1}) + K(\tau'_i - \tau_i). \end{aligned}$$

By induction,

$$\|x(\theta_i)\| \leq \delta^i \|x(\tau_0)\| + K \sum_{m=0}^i \delta^{i-m} (\tau'_m - \tau_m).$$

Let $S_i = \sum_{m=0}^i \delta^{i-m}(\tau'_m - \tau_m)$, from Abel's Test, S_i converges as $i \rightarrow \infty$. And also, notice that

$$\begin{aligned} \sum_{i=0}^{\infty} S_i &= \sum_{i=0}^{\infty} \sum_{m=0}^i \delta^{i-m}(\tau'_m - \tau_m) \\ &= \left(\sum_{m=0}^{\infty} (\tau'_m - \tau_m) \right) \left(\sum_{i=0}^{\infty} \delta^i \right) \\ &= \frac{\tau_{\infty}}{1-\delta} < \infty \end{aligned}$$

Therefore, as $i \rightarrow \infty$, $S_i \rightarrow 0$, which yields that $\|x(\theta_i)\| \rightarrow 0$, hence $\hat{x} = 0$. ■

Proposition 5

Consider a Zeno execution with $Z_{\infty} = \{\hat{q}\} \times X_{\infty}$, for some $\hat{q} \in Q$, $X_{\infty} \subseteq X$, if $R(\hat{q}, \hat{q}, x)$ is a function and for some $\delta \in (0, 1)$,

$$\|R(\hat{q}, \hat{q}, x) - R(\hat{q}, \hat{q}, y)\| \leq \delta \|x - y\|, \quad \text{for all } x, y \in G(\hat{q}, \hat{q}),$$

then $Z_{\infty} = \{(\hat{q}, \hat{x})\}$ for some $\hat{x} \in X$.

Proof: By hypothesis, $R(\hat{q}, \hat{q}, x)$ is a contracting mapping, thus it has a unique fixed point $x^* \in X$. We claim that x^* is the continuous part of the Zeno state, i.e., $\hat{x} = x^*$. There are two steps in the proof.

1) We start by showing that $f(\hat{q}, x(t))$ is bounded for Zeno execution $\chi = (\tau, q, x)$

Since \hat{q} is the only discrete state defining Z_{∞} , there exists $N > 0$ such that for all $t \in [\tau_N, \tau_{\infty})$, $q(t) = \hat{q}$. For $\tau_0 \leq t \leq \tau_N$, $x(t)$ is continuous on $[\tau_i, \tau'_i]$, $\forall 0 \leq i \leq N-1$, hence $x(t)$ is bounded. For $\tau_i \leq t \leq \tau'_i$, $i \geq N$,

$$x(t) = x(\tau_i) + \int_{\tau_i}^t f(\hat{q}, x(\tau)) d\tau,$$

There exists some $c > 0$, such that

$$\begin{aligned} \|x(t) - x^*\| &\leq \|x(\tau_i) - x^*\| + \int_{\tau_i}^t c(\|x(\tau)\| + 1) d\tau \\ &= \|x(\tau_i) - x^*\| + c(\|x^*\| + 1)(t - \tau_i) + \int_{\tau_i}^t c\|x(\tau) - x^*\| d\tau. \end{aligned}$$

Let

$$\begin{aligned} z(t) &= \|x(t) - x^*\|, \\ u(t) &= \|x(\tau_i) - x^*\| + c(\|x^*\| + 1)(t - \tau_i), \\ a(t) &= c. \end{aligned}$$

Applying Bellman-Gronwall lemma, we have

$$\begin{aligned} \|x(t) - x^*\| &\leq \|x(\tau_i) - x^*\| + (\|x^*\| + 1)(e^{c(t-\tau_i)} - 1) \\ &\leq \|x(\tau_i) - x^*\| + (\|x^*\| + 1)e^{c(t-\tau_i)}. \end{aligned}$$

Since $x(\tau_i) = R(\hat{q}, \hat{q}, (x(\tau'_{i-1})))$ and $r(x^*) = x^*$,

$$\|x(\tau_i) - x^*\| \leq \delta \|x(\tau'_{i-1}) - x^*\|.$$

Now

$$\begin{aligned} \|x(t) - x^*\| &\leq \delta \|x(\tau'_{i-1}) - x^*\| + (\|x^*\| + 1)e^{c(t-\tau_i)} \\ &\leq \|x(\tau'_{i-1}) - x^*\| + (\|x^*\| + 1)e^{c(t-\tau_i)}. \end{aligned}$$

Proceeding further,

$$\|x(t) - x^*\| \leq \|x(\tau'_{N-1}) - x^*\| + (\|x^*\| + 1)e^{c(t-\tau_N)},$$

then

$$\|x(t)\| \leq \|x(\tau'_{N-1}) - x^*\| + (\|x^*\| + 1)e^{c(t-\tau_N)} + \|x^*\|.$$

We can now define $x(\tau_\infty)$ such that

$$\|x(\tau_\infty)\| \leq \|x(\tau'_{N-1}) - x^*\| + (\|x^*\| + 1)e^{c(\tau_\infty - \tau_N)} + \|x^*\|.$$

Then $x(t)$ is bounded on $[\tau_0, \tau_\infty]$, and there exists $K > 0$, $\|f(q_i, x(t))\| \leq K$.

2) Next, for all $\{\theta_i\}_{i=0}^\infty$, $\theta_i \in [\tau_i, \tau'_i]$, $i > N$,

$$\begin{aligned} \|x(\theta_i) - x^*\| &\leq \|x(\tau_i) - x^*\| + \left\| \int_{\tau_i}^{\theta_i} f(\hat{q}, x(\tau)) d\tau \right\| \\ &\leq \|x(\tau_i) - x^*\| + K(\theta_i - \tau_i), \end{aligned}$$

then

$$\begin{aligned} \|x(\theta_i) - x^*\| &\leq \delta \|x(\tau'_{i-1}) - x^*\| + K(\tau'_i - \tau_i) \\ &= \delta \|x(\tau_{i-1}) + \int_{\tau_{i-1}}^{\tau'_{i-1}} f(\hat{q}, x(\tau)) d\tau - x^*\| + K(\tau'_i - \tau_i) \\ &\leq \delta \|x(\tau_{i-1}) - x^*\| + K\delta(\tau'_{i-1} - \tau_{i-1}) + K(\tau'_i - \tau_i). \end{aligned}$$

So we can derive

$$\|x(\theta_i) - x^*\| \leq \delta^{i-N} \|x(\tau_N)\| + K \sum_{m=N}^i \delta^{i-m} (\tau'_m - \tau_m).$$

Similar to the proof in the Proposition 4,

$$\|x(\theta_i) - x^*\| \rightarrow 0, \quad \text{as } i \rightarrow \infty,$$

hence, $\hat{x} = x^*$. ■

Proposition 6

Consider an invariant preserving Zeno hybrid automaton with $Z_\infty = \{(\hat{q}_i, \hat{x}_i)\}_{i \in I}$, where I is some index set. If $G(\hat{q}, \hat{q}') \cap I(\hat{q})^\circ = \emptyset$ for all $\hat{q}, \hat{q}' \in Q_\infty = \{q_i\}_{i \in I}$ and $(\hat{q}, \hat{q}') \in E$, then for all $i \in I$, $\hat{x}_i \in \partial I(\hat{q}_i)$. Furthermore, if $\hat{x}_i = \hat{x}$, $\forall i \in I$, then $\hat{x} \in \bigcap_{i \in I} \partial I(\hat{q}_i)$.

Proof: Note that for all (\hat{q}_i, \hat{x}_i) , there exists a subsequence $\{\theta_{i_n}\}_{n=0}^\infty, \theta_{i_n} \in [\tau_{i_n}, \tau'_{i_n}]$ such that $q(\theta_{i_n}) = \hat{q}_i$ and $\|x(\theta_{i_n}) - \hat{x}_i\| \rightarrow 0$. Since

$$\|x(\theta_{i_n}) - x(\tau_{i_n})\| \leq K\|\theta_{i_n} - \tau_{i_n}\| \rightarrow 0,$$

we have $\|x(\tau_{i_n}) - \hat{x}_i\| \rightarrow 0$ for all θ_{i_n} . By assumption the automaton is invariant preserving, $x(\theta_{i_n}) \in \overline{I(\hat{q}_i)}$. When n is large enough, all the discrete transitions will only take place in Q_∞ . Otherwise, if for all n there exist some $e_n \notin Q_\infty$ will happen after $t = \tau'_{i_n}$, then we can find some $e \in \{e_n\}_{n=0}^\infty$ that appears infinitive often, which contradicts to $e \notin Q_\infty$. So there exists some $\hat{q}'_i \in Q_\infty$ such that $x(\tau_{i_n}) \in G(\hat{q}_i, \hat{q}'_i)$, hence $x(\tau_{i_n}) \in G(\hat{q}_i, \hat{q}'_i) \cap \overline{I(\hat{q}_i)}$. Since $G(\hat{q}_i, \hat{q}'_i) \cap I(\hat{q}_i)^\circ = \emptyset$, we have $x(\tau_{i_n}) \in \partial I(\hat{q}_i)$. Notice that $\partial I(\hat{q}_i)$ is a closed set, so that $\hat{x} = \lim_{n \rightarrow \infty} x(\tau_{i_n}) \in \partial I(\hat{q}_i)$. The second part is obvious. ■

Proposition 7

An invariant preserving hybrid automaton with non-expanding reset has no Zeno execution if

- $G(q, q') \cap \overline{I(q)} \subseteq \partial \overline{I(q)}$ for all $(q, q') \in E$,
- $\bigcap_{i=1}^N \partial I(q_i) = \emptyset$.

Proof: Assume the hybrid automaton has Zeno execution $\chi = (\tau, q, x)$, from Proposition 3, we know that $Z_\infty = Q_\infty \times \{\hat{x}\}$ for some $Q_\infty \subseteq Q$ and $\hat{x} \in X$. By virtue of Proposition 6, we have $\hat{x} \in \bigcap_{i=1}^N \partial I(q_i) = \emptyset$, which gives the contradiction. ■