Outline

1 Recap
   - Existence and uniqueness of solutions
   - Continuity with respect to initial condition and simulation

2 Hybrid automata
   - Definition
   - State evolution
   - Example: water tank system

Sufficient condition for existence and uniqueness

To eliminate pathological cases we need to impose some assumptions on \( f \)

**Definition**
A function \( f : \mathbb{R}^n \to \mathbb{R}^n \) is called Lipshitz continuous if there exists \( \lambda > 0 \) such that for all \( x, \hat{x} \in \mathbb{R}^n \)

\[
\| f(x) - f(\hat{x}) \| \leq \lambda \| x - \hat{x} \|
\]

\( \lambda \) is known as the Lipschitz constant. A Lipshitz continuous function is continuous, but not necessarily differentiable. All differentiable functions with bounded derivatives are Lipshitz continuous.

**Exercise**
Show that for \( x \in \mathbb{R} \) the function \( f(x) = |x| \) that returns the absolute value of \( x \) is Lipshitz continuous. What is the Lipshitz constant? Is \( f \) continuous? Is it differentiable?
Sufficient condition for existence and uniqueness

**Theorem (existence & uniqueness of solutions)**

If $f$ is Lipschitz continuous, then the differential equation

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

has a unique solution $x(\cdot): [0, T] \to \mathbb{R}^n$ for all $T \geq 0$ and all $x_0 \in \mathbb{R}^n$.

This theorem allows us to check whether the differential equation models we develop make sense.

It also allows us to spot potential problems with proposed solutions. For example, uniqueness implies that solutions can not cross.

**Home Work**

Why does uniqueness imply that trajectories can not cross?

---

Continuity with respect to initial condition

**Theorem (continuity with initial state)**

Assume $f$ is Lipschitz continuous with Lipschitz constant $\lambda$. Let $x(\cdot): [0, T] \to \mathbb{R}^n$ and $\hat{x}(\cdot): [0, T] \to \mathbb{R}^n$ be solutions to $\dot{x} = f(x)$ with $x(0) = x_0$ and $\hat{x}(0) = \hat{x}_0$ respectively. Then for all $t \in [0, T]$,

$$||x(t) - \hat{x}(t)|| \leq ||x_0 - \hat{x}_0||e^{\lambda t}$$

In other words, solutions that start close to one another remain close to one another.

This is a theoretical justification for simulation algorithms.

---

Simulation

- Most nonlinear differential equations are impossible to solve by hand.
- One can however approximate the solution on a computer, using numerical algorithms for computing integrals (Euler, Runge-Kutta, etc.). This is a process known as simulation.
- Powerful computer packages, such as Matlab and Mathematica, make the simulation of most systems relatively straightforward.
- For example, the pendulum trajectories can be generated with:

  ```matlab
  function [xprime] = pendulum(t,x)
  l=1;
  m=1;
  d=1;
  g=9.8;
  xprime(1) = x(2);
  xprime(2) = -sin(x(1))*g/l-x(2)*d/m;
  ```

The continuity property ensures that the numerical approximation and the actual solution remain close.

---

Simulation

- The simulation code is then simply

  ```matlab
  >> x=[0.75 0];
  >> [T,X]=ode45('pendulum', [0 10], x');
  >> plot(T,X);
  >> grid;
  ```

---

Figure 2.2: Trajectory of the pendulum.

Figure 2.3: The pendulum vector field.

- The state of the system. The size of the state vector (in this case $n=2$) is called the dimension of the system. Notice that the dimension is the same as the order of the original ODE. The function $f(\cdot): \mathbb{R}^2 \to \mathbb{R}^2$ which describes the dynamics is called a vector field, because it assigns a “velocity” vector to each state vector. Figure 2.3 shows the vector field of the pendulum.

Exercise 2.3

Other choices are possible for the state vector. For example, for the pendulum one can use $x_1 = \theta + \dot{\theta}$ and $x_2 = \dot{\theta}$. What would the vector field be for this choice of state?

Solving the ODE for $\theta$ is equivalent to finding a function $x(\cdot): \mathbb{R} \to \mathbb{R}^2$.
Hybrid automata

One needs a modeling language that is

- **descriptive**, to allow one to capture different types of continuous and discrete dynamics
  - be capable of modeling different ways in which discrete evolution affects and is affected by continuous evolution
  - allow non-deterministic models (e.g., the thermostat) to capture uncertainty, etc.

- **composable**, to allow one to build large models by composing models of simple components
  - e.g., for the AHS application

- **abstractable**, refinement of design problems for composite models down to design of individual components
  - compose results about the individual components to study the overall system

Modeling languages that possess at least some subset of these properties have been developed in the hybrid systems literature

- Different languages place emphasis on different aspects, depending on types and problems they are designed to address

- In this class we will concentrate on one such language, called **hybrid automata**

- We begin with studying autonomous systems, i.e., have no inputs and outputs
Two Perspectives ➞ Different Focus

- The study of Hybrid Systems comes from two perspectives:
  - **Computer Science**: Software engineering for systems that interact with the physical environment
  - **Control Theory**: Apply the power of the new technology that most controllers are implemented with

- Accordingly, focus is put on different aspects:
  - **Computer Science**: Simple differential equations (constant), focus on rules that govern mode switches (automata)
  - **Control Theory**: Focus on more realistic differential equations (state space, non-linear), the rules that govern mode switches are not modeled (switched systems, $\Sigma^*$)

Formal Definition

**Definition (hybrid automaton)**

A hybrid automaton $H$ is a tuple $H = \langle Q, X, Init, f, Inv, E, G, R \rangle$, where

- $Q$ is a countable (usually finite) set of discrete states
- $X$ is the space of continuous states (usually $\mathbb{R}^n$, for some $n \in \mathbb{N}$)
- $Init \subseteq Q \times X$ is a set of initial states
- $f : Q \times X \to X$ assigns to each $q \in Q$ a vector field
- $Inv : Q \to 2^X$ assigns to each $q \in Q$ an invariant set
- $E \subseteq Q \times Q$ is a collection of discrete transitions
- $G : E \to 2^X$ assigns to each $e = \langle q, q' \rangle \in E$ a guard
- $R : E \times X \to 2^X$ assigns to each $e = \langle q, q' \rangle \in E$ and $x \in X$ a reset relation

State evolution

- Starting from an initial value $\langle q_0, x_0 \rangle \in Init$, the continuous state $x$ flows according to the differential equation
  \[
  \dot{x} = f(q_0, x), \quad x(0) = x_0
  \]
  while the discrete state $q$ remains constant $q(t) = q_0$
- continuous evolution can go on as long as $x$ remains in $Dom(q_0)$
- if at some point the continuous state $x$ reaches the guard $G(q_0, q_1) \subseteq \mathbb{R}^n$ of some edge $\langle q_0, q_1 \rangle \in E$, the discrete state may change value to $q_1$
- at the same time, the continuous state gets reset to some value in $R(q_0, q_1, x) \subseteq \mathbb{R}^n$
- after this discrete transition, continuous evolution resumes and the whole process is repeated
Simplifying Assumptions

To simplify the discussion, we assume from now on:

- the number of discrete states is finite
  - enough for most application
- for all $q \in Q$ the vector field $f(q, \cdot)$ is Lipschitz continuous
  - ensures that the solutions of $\dot{x} = f(q, x)$ are well defined
- for all $e \in E$, $G(e) \neq \emptyset$, and for all $x \in G(e)$, $R(e, x) \neq \emptyset$.
  - eliminates pathological cases
  - can be imposed without loss of generality

Graphical representation

- It is often convenient to visualize hybrid automata as directed graphs $(Q, E)$ with vertices $Q$ and edges $E$
- With each vertex $q \in Q$, we associate:
  - a set of initial states $\{x \in X : \langle q, x \rangle \in \text{Init} \}$
  - a vector field $f(q, \cdot) : X \to X$
  - a domain $\text{Dom}(q) \subseteq X$
- With each edge $\langle q, q' \rangle \in E$, we associate:
  - a guard $G(q, q') \subseteq X$
  - a reset function $R(q, q', \cdot) : X \to 2^X$

Example: water tank system

- Two tanks containing water
- Both tanks are leaking at a constant rate
- Water is added to the system at a constant rate through a hose that, at any point in time, is dedicated to either one tank or the other
- The hose can switch between the tanks instantaneously

Example: water tank system

- $x_i$ volume of water in Tank $i$
- $v_i$ the constant flow of water out of Tank $i$
- $w$ the constant flow of water into the system

The objective is to keep the water volumes above $r_1$ and $r_2$
Hybrid automaton

- \( Q = \{q_1, q_2\} \): two discrete states, inflow going left and inflow going right
- \( X = \mathbb{R}^2 \): two continuous variables, \( x_1 \) and \( x_2 \), represent levels of water in the tanks
- \( f(q_1, x) = \left( \frac{w - v_1}{w - v_2} \right) \) and \( f(q_2, x) = \left( \frac{-v_1}{w - v_2} \right) \)
- \( \text{Init} = \{q_1, q_2\} \times \{x \in \mathbb{R}^2: x_1 \geq r_1 \land x_2 \geq r_2\} \)
- \( \text{Dom}(q_1) = \{x \in \mathbb{R}^2: x_2 \geq r_2\} \) and \( \text{Dom}(q_2) = \{x \in \mathbb{R}^2: x_1 \geq r_1\} \)
- \( E \): put water in the current tank, inflow can switch from left to right and vice versa
- \( G(q_1, q_2) = \{x \in \mathbb{R}^2: x_2 \leq r_2\} \) and \( G(q_2, q_1) = \{x \in \mathbb{R}^2: x_1 \leq r_1\} \)
- \( K \): can switch the inflow to the other tank while the continuous state does not change as a result of switching the inflow

Example: water tank system

- The graphs contain exactly the same information as the definition
- They can therefore be treated as definitions of hybrid automata
- It is common to remove the assignment \( x := x \) from an edge of the graph when the continuous state does not change as a result of the discrete transition corresponding to that edge