Chapter One

Introduction

The model of a hybrid system used in this book is informally presented in this section. The focus is on the data structure and on modeling. Several examples are given, including models of hybrid control systems. The model of a hybrid system is then related to other modeling frameworks, such as hybrid automata, impulsive differential equations, and switching systems. A formal presentation of the model, together with a rigorous definition of the solution, is postponed until Chapter 2.

1.1 THE MODELING FRAMEWORK

The model of a hybrid system used in this book can be represented in the following form:

$$\begin{cases} x \in C & \dot{x} \in F(x) \\ x \in D & x^+ \in G(x). \end{cases} \tag{1.1}$$

A reader less familiar with set-valued mappings and differential or difference inclusions may choose to keep in mind a less general representation involving equations:

$$\begin{cases} x \in C & \dot{x} = f(x) \\ x \in D & x^+ = g(x). \end{cases} \tag{1.2}$$

This representation suggests that the state of the hybrid system, represented by x, can change according to a differential inclusion $\dot{x} \in F(x)$ or differential equation $\dot{x} = f(x)$ while in the set C, and it can change according to a difference inclusion $x^+ \in G(x)$ or difference equation $x^+ = g(x)$ while in the set D. The notation $\dot{x}$ represents the velocity of the state x, while x^+ represents the value of the state after an instantaneous change.

A rigorous statement of what constitutes a model of a hybrid system and what is a solution to the model is postponed until Chapter 2. This chapter focuses on modeling of various hybrid systems in the form (1.1) or (1.2).

To shorten the terminology, the behavior of a dynamical system that can be described by a differential equation or inclusion is referred to as *flow*. The behavior of a dynamical system that can be described by a difference equation or inclusion is referred to as *jumps*. This leads to the following names for the four objects involved in (1.1) or (1.2):

- C is the *flow set*.
- F (or f) is the *flow map*.
- D is the *jump set*.
- G (or g) is the *jump map*.

This book discusses hybrid systems in finite-dimensional spaces, that is, the flow set C and the jump set D are subsets of an n-dimensional Euclidean space $\mathbb{R}^n$. For consistency in the model, it will be required that the function f, respectively g, be defined on at least the set C, respectively D. In the case of set-valued flow and jump maps, it will be required that F, respectively G, have nonempty values on C, respectively D.

As the model in (1.2) or (1.1) suggests, the flow set, the flow map, the jump set, and the jump map can be specialized to capture the dynamics of purely continuous-time or discrete-time systems on $\mathbb{R}^n$. The former corresponds to a flow set equal to $\mathbb{R}^n$ and an empty jump set, while the latter can be captured with an empty flow set and a jump set defined as $\mathbb{R}^n$.

1.2 EXAMPLES IN SCIENCE AND ENGINEERING

Many mechanical systems experience impacts. Examples range from elaborate systems such as walking robots, through colliding billiard balls or the Newton's cradle, to a seemingly simple bouncing ball. Such systems flow in between impacts. A rough approximation of the impacts suggests considering them as instantaneous, and hence, as leading to jumps in the state of the system. Consequently, systems with impacts can be viewed as hybrid systems.

The first example is the mentioned bouncing ball. This example, and some of the later ones in this chapter, reappear throughout the book as illustrations of various properties and results.

Example 1.1. (Bouncing ball) Consider a point-mass bouncing vertically on a horizontal surface. In between impacts the point-mass flows, experiencing acceleration due to gravity. At impacts, when the point-mass hits the surface, the change in velocity is approximated as being instantaneously reversed and possibly diminished in magnitude due to dissipation of energy.

The state of the point-mass can be described with

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2,$$

where x_1 represents the height above the surface and x_2 represents the vertical velocity. It is natural to say that flow is possible when the point-mass is above the surface, or when it is at the surface and its velocity points up. Hence, the flow set is

$$C = \left\{ x \in \mathbb{R}^2 \,:\, x_1 > 0 \text{ or } x_1 = 0, x_2 \geq 0 \right\}.$$

The choice of a flow map is delicate at one point in C, that is, at $x = 0$. First, it is natural to say that

$$f(x) = \begin{pmatrix} x_2 \\ -\gamma \end{pmatrix} \qquad \text{when } x_1 > 0 \text{ or } x_1 = 0, x_2 > 0,$$

where $-\gamma$ is the acceleration due to gravity. Second, it is natural to say that $f(0) = 0$; it has to be accepted, though, that the resulting flow map f is not continuous at 0. Impacts happen when the point-mass is on the surface with negative velocity. Hence, the jump set is

$$D = \left\{ x \in \mathbb{R}^2 \,:\, x_1 = 0, x_2 < 0 \right\}.$$

The jump map is given, for some $\lambda \in (0, 1)$, by

$$g(x) = \begin{pmatrix} 0 \\ -\lambda x_2 \end{pmatrix}.$$

An alternative choice for g is the vector $-\lambda x$ since this function agrees with $g(x)$ on the set D. Figure 1.1 illustrates the data of the bouncing ball system.

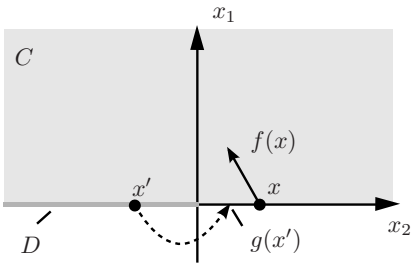


Figure 1.1: Flow and jump sets for the bouncing ball system in Example 1.1.

In the bouncing ball model above, every jump is followed by a period of flow. In other words, consecutive jumps do not happen. Consecutive jumps can happen in other systems with impacts, like in a model of Newton's cradle. Newton's cradle consists of at least three identical steel balls, each of which is suspended on a pendulum. At the stationary state, the balls are aligned along a horizontal line. Lifting a ball from one end of the alignment and releasing it leads to a collision of the lifted ball with the remaining balls. After the collision, the ball that was lifted and released becomes stationary and the ball on the other end of the alignment swings up. One way to model this interaction is to consider a sequence of collisions between pairs of adjacent balls.

A number of biological systems, such as groups of fireflies or crickets, are able to produce synchronized behavior, flashing or chirping, respectively, through a dynamical mechanism that can be viewed as hybrid.

Example 1.2. (Flashing fireflies) The timing of flashes of a firefly is determined by the firefly's internal clock. In between flashes, the internal clock gradually increases. When it reaches a threshold, a flash occurs and the clock is instantly reset to 0. In a group of fireflies, the flash of one firefly affects the internal clock of all other fireflies. That is, when a firefly witnesses a flash from another firefly, its internal clock instantly increases to a value closer to the threshold.

To model the internal clocks of n fireflies, normalize units so that each firefly's internal clock, denoted x_i, takes values in the interval $[0, 1]$, i.e., every threshold is 1. The flow set is then

$$C = [0, 1)^n := \{x \in \mathbb{R}^n \,:\, x_i \in [0, 1),\ i = 1, 2, \ldots, n\}.$$

In between the flashes, every clock state flows toward the threshold according to the differential equation $\dot{x}_i = f_i(x_i)$, where $f_i : [0, 1] \to \mathbb{R}_{>0}$, $i = 1, 2, \ldots, n$, is continuous. This defines the flow map f.

Jumps occur when one of the internal clocks reaches the threshold. Thus, the jump set is

$$D = \left\{x \in [0, 1]^n \,:\, \max_i x_i = 1\right\}.$$

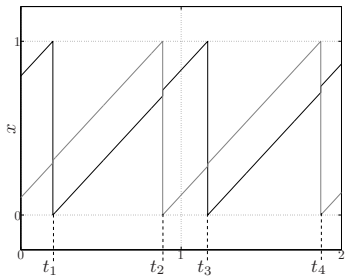
One method to model the (instantaneous) changes in internal clocks during a flash is through the jump map defined by

$$g(x) = \begin{pmatrix} g_1(x_1) \\ g_2(x_2) \\ \vdots \\ g_n(x_n) \end{pmatrix}, \qquad g_i(x_i) = \begin{cases} (1 + \varepsilon)x_i, & \text{when } (1 + \varepsilon)x_i < 1, \\ 0, & \text{otherwise,} \end{cases}$$

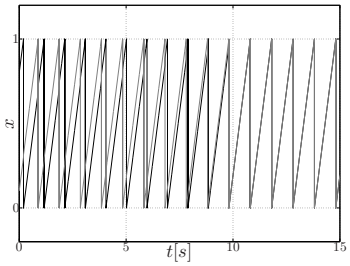
where $\varepsilon > 0$. This indicates that the internal clock x_i of a firefly witnessing a flash increases to $(1 + \varepsilon)x_i$, unless this would result in reaching or exceeding the threshold, in which case the internal clock is reset to 0 together with the internal clock of the flashing firefly. Figure 1.2 illustrates the evolution of the clock variable x for $n = 2$ and $n = 10$ when $f_i \equiv 1$ for each i.

Example 1.3. (Power control with a thyristor) Consider the electric circuit in Figure 1.3(a) for controlling the power delivered to a load. The load consists of a resistor R and an inductance L that is connected to a power source through a thyristor with a gate control port. A simple model describing the operation of the thyristor is as follows. When in conduction mode, which can be triggered through the gate port, the thyristor allows flow of current from anode to cathode, which are the terminals denoted as $a+$ and $c-$ in Figure 1.3(a), respectively. It will turn off once the current from anode to cathode becomes zero. The load current is denoted by i_L, its voltage by v_L, and the capacitor's voltage by $v_\circ$. The sinusoidal input voltage with angular frequency ω is denoted by v_s and is generated by the output $v_s = z_1$ of the system

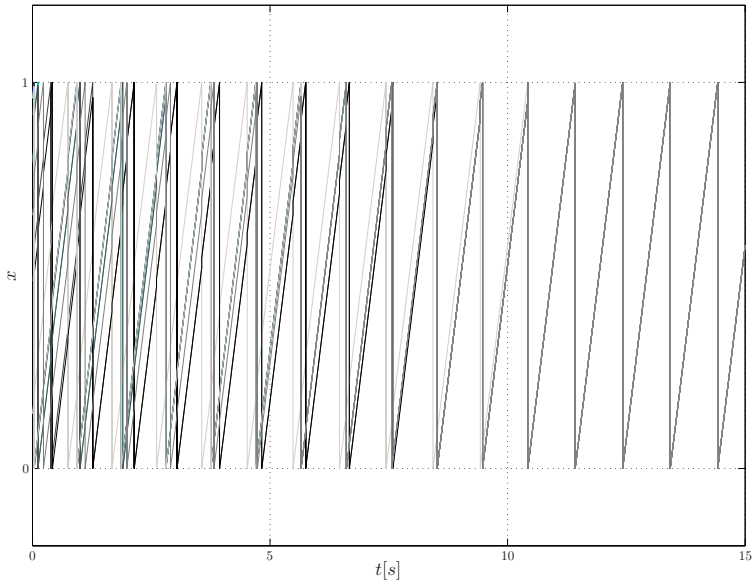
$$\dot{z}_1 = \omega z_2, \qquad \dot{z}_2 = -\omega z_1. \tag{1.3}$$



(a) $n = 2$



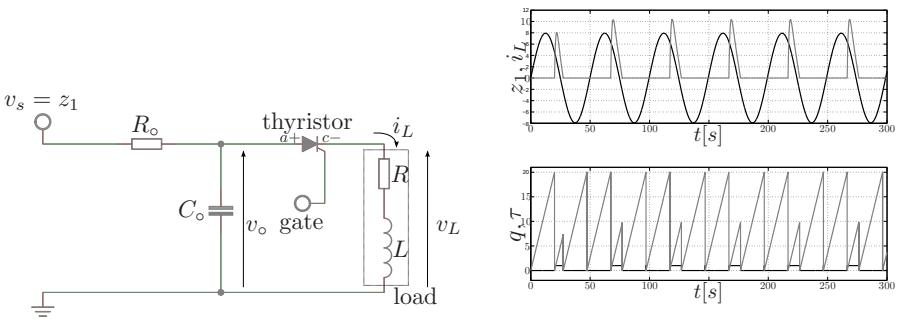
(b) $n = 2$



(c) $n = 10$

Figure 1.2: Evolution of coupled impulsive oscillators in fireflies with unitary threshold and $f_i \equiv 1$ for each i.

A discrete state $q \in \{0, 1\}$ is used to indicate whether the thyristor is *on* ($q = 1$) or *off* ($q = 0$), while a continuous state $\tau \in \mathbb{R}$ is used to model the firing events in the gate port, given as a function of the firing angle parameter $\alpha \in (0, \pi)$.



(a) Circuit diagram.

(b) System trajectories. Parameters: $R = 1, L = 0.5, C_\circ = 10, R_\circ = 0.25, \omega = 0.1/(2\pi), \alpha = 20\omega$, $x(0,0) = [0\ 1\ 0\ 0\ 0\ 0]^\top$.

Figure 1.3: Power control circuit with thyristor.

By defining the state of the system to be

$$x := (z_1, z_2, i_L, v_\circ, q, \tau) \in \mathbb{R}^6,$$

the continuous dynamics are defined by

$$F(x) = \begin{pmatrix} \omega z_2 \\ -\omega z_1 \\ q\left(\frac{v_\circ - R\, i_L}{L}\right) \\ -\frac{1}{C_\circ R_\circ} v_\circ + \frac{1}{C_\circ R_\circ} z_1 - \frac{1}{C_\circ} i_L \\ 0 \\ 1 \end{pmatrix}.$$

These equations can be derived applying electrical circuit theory for each mode of operation. Note that $\dot{q} = 0$ indicates that the discrete state remains constant during flows, and that $\dot{\tau} = 1$ enforces that τ counts the flow time in between switches. Assuming that when the thyristor is in off mode the load current is zero, two conditions trigger switches of the thyristor mode:

- When the thyristor is *off* ($q = 0, i_L = 0$), the firing angle has been reached ($\tau \geq \alpha/\omega$), and the capacitor voltage is positive ($v_0 > 0$), then switch to *on* ($q = 1$).

- When the thyristor is *on*, the load current is zero and decreasing ($i_L = 0$, $\dot{i}_L < 0$), then switch to *off*.

These conditions can be captured with the flow and jump sets

$$\begin{aligned} C &:= \left\{x : q = 0, \tau < \frac{\alpha}{\omega}, i_L = 0\right\} \cup \{x : q = 1, i_L > 0\}, \\ D &:= \left\{x : q = 0, \tau \geq \frac{\alpha}{\omega}, i_L = 0, v_0 > 0\right\} \cup \{x : q = 1, i_L = 0, v_0 < 0\}, \end{aligned}$$

and the jump map

$$G(x) := (z_2 \;\; z_1 \;\; i_L \;\; v_\circ \;\; 1 - q \;\; 0)^\top.$$

At every jump, q is toggled and the timer is restarted to trigger the next jump to *on* mode at the programmed firing angle. The top plot in Figure 1.3(b) shows the input voltage with $\omega = 0.1/(2\pi)$ rad/sec and the resulting load's current with a firing angle of 20ω rad, while the bottom plot shows the associated logic and timer states.

1.3 CONTROL SYSTEM EXAMPLES

The control of a continuous-time system with state feedback faces both practical and theoretical obstacles: precise information about the state may not be available at all times, even if frequent measurements of the state are available; the behavior of the closed-loop system may be very sensitive to errors in the state measurements; or satisfactory performance of the closed-loop system may not be achievable by using just one state-feedback controller. These issues provide motivation for the use of hybrid control, several simple instances of which are described below.

Example 1.4. (Sample-and-hold control) Given a continuous-time control system and a state-feedback controller, associating with each state of the system the control to be applied there, a *sample-and-hold* implementation of the feedback is essentially as follows:

- *sample:* measure the state of the system, and use the feedback controller to obtain the control value based on the measurements;
- *hold:* apply the computed constant control value for certain amount of time;

and repeat the procedure infinitely many times. The processes of sampling and computing the control can be modeled as an instantaneous event. This leads to a continuous behavior of the closed-loop system in between the sampling times, according to the continuous-time dynamics of the control system and the constant value of the control, and an instantaneous change at every sampling time, when the control value is instantly updated.

A schematic example of a sample-and-hold control system is in Figure 1.4, where a digital device controls an analog plant. The basic operation of the system is as follows. The output of the plant is sampled by an *analog-to-digital*

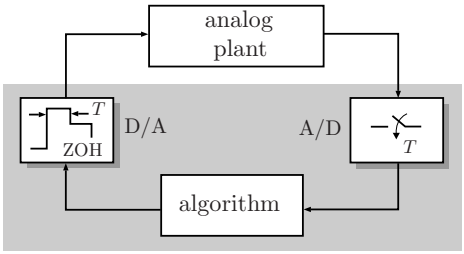


Figure 1.4: Digital control of a continuous-time nonlinear system with sample-and-hold devices.

converter, denoted A/D. The digitized output is processed by the algorithm, and the result is applied to the plant through a *digital-to-analog converter*, denoted D/A. For a periodic A/D sampler and a zero-order hold (ZOH) type of D/A, the output samples and control input updates occur at a fixed sampling period T.

To model such a system as a hybrid system, suppose that the control system is given by

$$\dot{z} = \widetilde{f}(z, u), \tag{1.4}$$

where $z \in \mathbb{R}^{n_p}$ is the state of the system, $u \in \mathbb{R}^{n_c}$ is the control variable, and $\widetilde{f} : \mathbb{R}^{n_p} \times \mathbb{R}^{n_c} \to \mathbb{R}^{n_p}$ is a function. Let the state-feedback controller be given by $u = \kappa(z)$. The standard closed-loop, without a sample-and-hold strategy, leads to a continuous-time closed-loop system

$$\dot{z} = \widetilde{f}(z, \kappa(z)).$$

A sample-and-hold implementation can be modeled as a hybrid system, with the state variable

$$x = \begin{pmatrix} z \\ u \\ \tau \end{pmatrix} \in \mathbb{R}^{n_p+n_c+1}.$$

Note that, for simplicity, the control input u itself is taken to be a state variable for the closed-loop system resulting from sample-and-hold control. Suppose that the sampling period is T. Flow occurs when the timer variable τ belongs to the interval $[0, T)$. During flow, the variable u remains constant, τ keeps track of elapsed time, and the state of the plant z evolves according to the dynamics in (1.4). Thus, the flow set and the flow map can be taken to be

$$C = \mathbb{R}^{n_p} \times \mathbb{R}^{n_c} \times [0, T), \qquad f(x) = \begin{pmatrix} \widetilde{f}(z, u) \\ 0 \\ 1 \end{pmatrix}. \tag{1.5}$$

Jumps occur when the timer variable reaches T. At jumps, the variable u is updated to $\kappa(x)$, the timer is reset to 0, and the state of plant does not change. Hence the jump set and the jump map can be taken to be

$$D = \mathbb{R}^{n_p} \times \mathbb{R}^{n_c} \times \{T\}, \qquad g(x) = \begin{pmatrix} z \\ \kappa(z) \\ 0 \end{pmatrix}. \tag{1.6}$$

Example 1.5. (A quantized control system) Some control systems that use quantized measurements include a mechanism for adjusting quantization parameters on-line. These adjustments are made to vary the accuracy of the measurements at different locations in the state space. For example, consider the control system

$$\dot{\zeta} = \zeta + u \tag{1.7}$$

with measurements

$$y = \mu q(\zeta/\mu),$$

where $q : \mathbb{R} \to \mathbb{R}$ is a function that represents measurement quantization and μ is a positive parameter that can be adjusted discretely as part of a control algorithm. The main requirement on the function q is that there exist positive real numbers Δ and M with $\Delta \ll M$ such that

$$|z| \leq M \text{ implies } |q(z) - z| \leq \Delta$$

$$|q(z)| \leq M - \Delta \text{ implies } |z| \leq M.$$

In this way, the value $q(z)$ gives some rough information about the value of z. An adaptive, quantized hybrid feedback law could consist of

- a feedback rule $u = -ky$, where $k > 1$, designed to steer the state ζ of (1.7) to zero;
- a discrete-time update rule for the parameter μ;
- a specification of sets where flows are allowed because μ does not need to be adjusted;
- a specification of sets where jumps are allowed because the parameter μ should be increased or decreased to put the argument of q into an acceptable range.

For example, letting the positive real numbers ℓ_{in}, ℓ_{out}, λ_{in}, and λ_{out} satisfy $\ell_{\text{in}} < \ell_{\text{out}}$ and $\lambda_{\text{in}} < 1 < \lambda_{\text{out}}$, consider taking the flow set to be

$$C = \{(\zeta, \mu) \in \mathbb{R} \times (0, \infty) : |q(\zeta/\mu)| \in [\ell_{\text{in}}, \ell_{\text{out}}]\},$$

the jump set to be

$$\begin{aligned} D_{\text{in}} &= \{(\zeta, \mu) \in \mathbb{R} \times (0, \infty) : |q(\zeta/\mu)| < \ell_{\text{in}}\} \\ D_{\text{out}} &= \{(\zeta, \mu) \in \mathbb{R} \times (0, \infty) : |q(\zeta/\mu)| > \ell_{\text{out}}\} \\ D &= D_{\text{in}} \cup D_{\text{out}}, \end{aligned}$$

and the jump map to be

$$g(\zeta,\mu)=\begin{cases}\lambda_{\text{in}}\mu & \forall(\zeta,\mu)\in D_{\text{in}}\\ \lambda_{\text{out}}\mu & \forall(\zeta,\mu)\in D_{\text{out}}.\end{cases}$$

The hybrid control algorithm increases or decreases the size of μ in an attempt to drive the state to the flow set. Depending on the initial value of (ζ,μ), multiple consecutive jumps may be required to reach the flow set. Ideally, ℓ_{in} and ℓ_{out} are chosen based on M and Δ so that, after some point in time, the system no longer reaches D_{out}, it repeatedly reaches D_{in}, and $|q(\zeta/\mu)|\leq M-\Delta$ so that $|\zeta/\mu|\leq M$. In this case, μ repeatedly shrinks by the factor λ_{in} and the convergence of μ to zero implies that ζ also converges to zero.

Example 1.6. (Reset linear control systems) In classical control theory, the output of a controller of a continuous-time plant evolves continuously in time. *Reset control systems* differ from those traditional controllers as their output experiences jumps caused by resets of the controller state. These resets may depend on the value of the controller inputs. In some scenarios, in comparison to (non-reset) classical controllers, reset controllers lead to improved system performance.

The first reset controller that appeared in the literature is the so-called *Clegg integrator*, a single-input/single-output linear controller that resets its output to zero when its input and output do not have the same sign. Figure 1.5 shows the response of the Clegg integrator to a sinusoidal input. During flows, the Clegg integrator's output is the integral of its input. Since the Clegg integrator does not permit the signs of its input and output to differ from one another, it forces a jump in its state when the input changes sign. After such a jump, the system flows again.

A schematic example of a plant controlled by a reset control system is in Figure 1.6. If the controller's input, which in Figure 1.6 is the error between the plant output and the reference input, and the controller's output satisfies a *reset condition* then the controller state is reset to a pre-specified value. The closed-loop system is a hybrid system with flows interrupted by state-dependent jumps, which are triggered when the reset condition is satisfied.

Consider a reset linear control system where the plant state is x_p and the controller state is x_c. The closed-loop system state is

$$x=\begin{pmatrix}x_p\\ x_c\end{pmatrix}\in\mathbb{R}^{n_p+n_c}.$$

Since the closed-loop system without resets is linear, the flow map is a linear function

$$f(x)=A_f x.$$

The resetting mechanism is also linear, so that the jump map has the form

$$g(x)=A_g x.$$

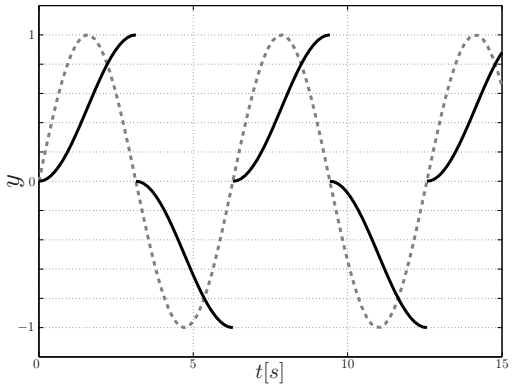


Figure 1.5: Output response y (solid) of Clegg integrator to a sinusoidal input (dashed).

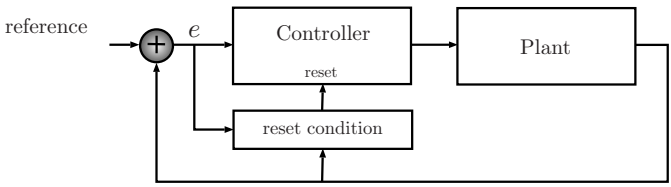


Figure 1.6: Closed-loop system with reset controller.

Resets typically occur when the state x satisfies some quadratic inequality, perhaps coming from insisting that two variables are always related by having the same sign. Thus, the jump set may have the form

$$D = \left\{ x \in \mathbb{R}^{n_p+n_c} : x^T M x \leq 0 \right\}$$

where $M = M^T$, that is, M is a symmetric matrix. One then may consider taking $C = \mathbb{R}^{n_p+n_c} \setminus D$. A particular construction of these sets is depicted in Figure 1.7.

Notice that the origin does not belong to the flow set but belongs to the jump set and that g maps the origin back to the origin. Thus, from the origin it is only possible to jump without ever flowing. To address this situation, one may consider forcing a small amount $\delta > 0$ of flow time between jumps. This can be done with a technique called "temporal regularization." In this case, one

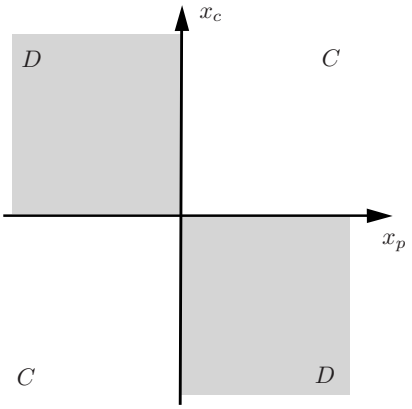


Figure 1.7: Examples of flow and jump sets for reset control with plant output $y = x_p$ and input $u = x_c$ ($n_p = n_c = 1$). The matrix $M \in \mathbb{R}^{2\times 2}$ is given by $(0, 1; 1, 0)$, which enforces that flows occur when the components of $x = (x_p, x_c)$ have the same sign.

augments the state with a timer variable τ and takes the jump set to be

$$D := \left\{ x \in \mathbb{R}^{n_p+n_c} \ , \ \tau \in \mathbb{R} : x^T M x \leq 0 \ , \ \tau \geq \delta \right\}.$$

The flow set is taken to be

$$C := \left\{ x \in \mathbb{R}^{n_p+n_c} \ , \ \tau \in \mathbb{R} : x^T M x > 0 \text{ or } \tau \in [0, \delta] \right\}.$$

The jump map is augmented with the equation $\tau^+ = 0$ and the flow map is augmented with the equation $\dot{\tau} = 1$ for $\tau \in [0, 2\delta)$, $\dot{\tau} = 0$ for $\tau = 2\delta$, which, in particular, keeps τ bounded.

Example 1.7. (Combining local and global controllers) In several control applications, the design of a continuous-time feedback controller that performs a particular control task is not possible. For example, in the problem of globally stabilizing a multi-link pendulum to the upright position with actuation on the first link only, topological constraints rule out the existence of a continuous-time feedback controller that accomplishes this task globally and robustly. However, it is often possible to overcome such topological obstructions using hybrid feedback control to combine continuous-time feedback controllers that achieve certain subtasks.

To illustrate this idea, consider the task of combining a high-performance controller that works only near a reference point with a controller that is able to steer every trajectory toward the reference point, but does not have very good performance near that point. We refer to these controllers as *local* and *global* controllers, respectively.

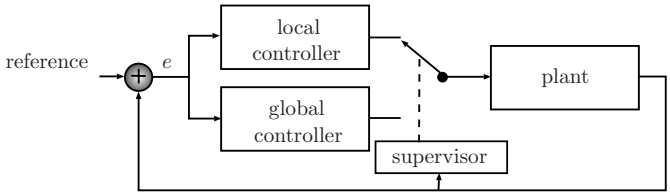


Figure 1.8: Closed-loop system combining local and global controllers.

Figure 1.8 depicts a block diagram of the control mechanism being described. Each controller measures the error signal given by the plant output and reference input. The controller selection is performed by a *supervisor* and is based on the plant's output and on the controller currently applied. Switching from one controller to the other results in a jump in the logic variable. In between the jumps, continuous evolution of the state of the system occurs.

More precisely, suppose that each of the two state feedback control laws, κ_1 and κ_2, asymptotically stabilizes the origin for the control system (1.4). Furthermore, suppose that κ_1 produces efficient transient responses, but works only near the origin, while κ_2 produces less efficient transients but works globally. The goal is to build a hybrid feedback law that globally asymptotically stabilizes the origin while using κ_1 near the origin and κ_2 far from the origin.

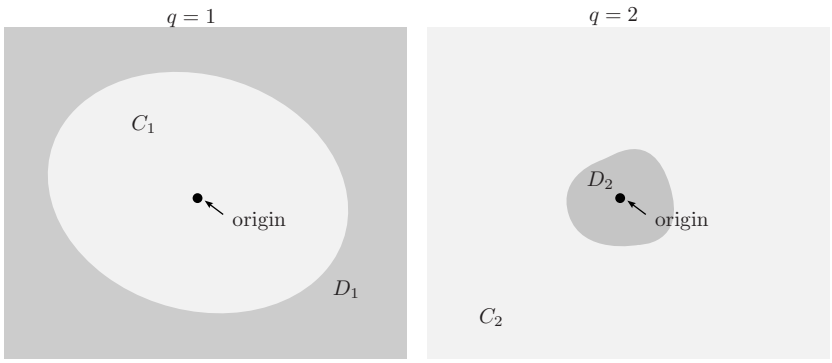


Figure 1.9: Sets for the hybrid controller combining control laws κ_1 and κ_2.

To eliminate the possibility of several instantaneous switches between controllers, a hysteresis mechanism is used. With the sets C_i, D_i, $i = 1, 2$ as in Figure 1.9, the switching idea is as follows: if κ_1 is being used and $z \in C_1$, do not switch, but if $z \in D_1$, switch to κ_2; while if κ_2 is being used and $z \in C_2$, do not switch, but if $z \in D_2$, switch to κ_1. Thus, when κ_1 is used, continuous evolution takes place when $z \in C_1$ and is described by $\dot{z} = \widetilde{f}(z, \kappa_1(z))$, while a jump

takes place when $z \in D_1$ and results in q toggled to 2. When κ_2 is used, continuous evolution takes place when $z \in C_2$ and is described by $\dot{z} = \widetilde{f}(z, \kappa_2(z))$, while a jump takes place when $z \in D_2$ and results in q toggled to 1. A general approach to modeling systems of this kind, where a logical variable (here equal to either 1 or 2) determines the hybrid dynamics, is given in Section 1.4.1. Here, it is illustrated that this can be modeled by including a logical variable q, taking values in $\{1, 2\}$, in the state

$$x = \begin{pmatrix} q \\ z \end{pmatrix} \in \mathbb{R}^{n_p+1},$$

and with the following sets and functions:

$$\begin{aligned} C &= (\{1\} \times C_1) \ \cup \ (\{2\} \times C_2), \qquad f(x) = \begin{pmatrix} 0 \\ \widetilde{f}(z, \kappa_q(z)) \end{pmatrix}, \\ D &= (\{1\} \times D_1) \ \cup \ (\{2\} \times D_2), \qquad g(x) = \begin{pmatrix} 3-q \\ z \end{pmatrix}. \end{aligned} \tag{1.8}$$

In fact, since $3 - q = 2$ when $q = 1$ and $3 - q = 1$ when $q = 2$, the state q is toggled when z enters the set D_q.

Finally, in order for the hybrid feedback law to work as intended, there should be a relationship between D_2 and C_1. In particular, if solutions to $\dot{z} = \widetilde{f}(z, \kappa_1(z))$ start in D_2, they should remain in a closed set that is a strict subset of C_1; moreover, any trajectory of this system that starts in C_1 and remains in C_1 should converge to the origin. Since the local controller is locally asymptotically stabilizing, both of these properties can be induced by first picking C_1 to be a sufficiently small neighborhood of the origin and then picking D_2 to be another sufficiently small neighborhood of the origin strictly contained in C_1.

To illustrate this hybrid feedback construction, consider a positioning control system used for data read/write in hard disk drives. The objective of the control algorithm is to provide precise positioning of the magnetic heads to read and write information from the disk's tracks. A technique utilized in commercial devices for this purpose is called *mode-switching control*. It combines two controllers for stabilizing the position of the magnetic head to a desired position p^* on the disk with zero velocity: a controller (global) capable of steering it to a neighborhood of p^* and a controller (local) capable of stabilizing it to p^* with high precision. The plant can be modeled as a double integrator $\dot{p} = v$, $\dot{v} = u$ with state $z = (p, v)$, where $p \in \mathbb{R}$ is the position, $v \in \mathbb{R}$ the velocity of the magnetic head of the hard disk drive, and $z^* = (p^*, 0)$ is the point to stabilize. Suppose that the global controller is given by κ_2 and the local controller by κ_1. The hybrid control scheme leading to the closed-loop system in (1.8) can be employed to accomplish the control objective. The set C_1 can be taken to be a compact neighborhood of z^* that is contained in the basin of attraction for z^* when using κ_1, and D_2 can be taken to be a compact neighborhood of z^* such

that solutions using κ_1 that start in D_2 do not reach the boundary of C_1. Then, $C_2 = \overline{\mathbb{R}^2 \setminus D_2}$ and $D_1 = \overline{\mathbb{R}^2 \setminus C_1}$.

1.4 CONNECTIONS TO OTHER MODELING FRAMEWORKS

The models (1.1) and (1.2) can describe several classes of hybrid systems that are frequently modeled in different frameworks. These different frameworks include hybrid automata, impulsive differential equations or inclusions, and switching systems. This section illustrates how models from these frameworks can be translated to (1.1) or (1.2). The benefit of passing to (1.1) or (1.2) is that the asymptotic stability theory developed in this book can then be applied to a broader class of systems. For example, as described in Section 8.5, invariance principles for hybrid systems can be applied to switching systems.

1.4.1 Systems with explicit "discrete states" or "logical modes"

The state in several hybrid systems can be decomposed into a "continuous state" and a "discrete state." The discrete state takes values in a discrete, often finite, set. It may represent a *mode* in which the system, or part of the system, is operating. For example, the discrete state can take values representing modes such as "on" or "off"; "first gear," "second gear," "third gear"; "controller 1" or "controller 2" as in Example 1.7; and so on. The discrete state, by its nature, can change only via a jump. The continuous state can change via flow and, sometimes, via a jump too. It may represent position, velocity, and other continuous-valued variables. For example, in a temperature control system, a discrete state can indicate whether a thermostat is "on" or "off" while a continuous state can indicate the temperature. In such a case, the continuous state may not change via a jump. If the discrete state represents whether a connection in an electrical circuit is "open" or "closed," as it does in Example 1.3, and the continuous variable represents the current in some part of the circuit, it may be natural to allow for instantaneous changes in the continuous variable that are simultaneous with changes in the discrete variable.

A system with continuous and discrete states usually can be represented by a set $Q = \{1, 2, \ldots, q_{\max}\}$, and for each $q \in Q$, a flow set $C_q \subset \mathbb{R}^n$, a flow map $F_q : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$, a jump set $D_q \subset \mathbb{R}^n$, and a jump map $G_q : \mathbb{R}^n \rightrightarrows Q \times \mathbb{R}^n$. The suggestive form to represent such a system, parallel to (1.1), is

$$\begin{cases} z \in C_q & \dot{z} \in F_q(z) \\ z \in D_q & (q, z)^+ \in G_q(z). \end{cases} \tag{1.9}$$

When the discrete variable q has the value $q^* \in Q$ and the continuous variable z is in the flow set C_{q^*}, flow is possible according to the inclusion $\dot{z} \in F_{q^*}(z)$. During flow, the discrete variable remains constant. The condition $\dot{q} = 0$ is not explicitly mentioned in (1.9). When the discrete variable has the value $q^* \in Q$

and the continuous variable z is in the jump set D_{q^*}, a jump is possible, with both q and z changing values according to G_{q^*}. For systems where the continuous variable does not change via jumps, the inclusion $(q, z)^+ \in G_q(z)$ can be replaced by the simpler $q^+ \in G_q(z)$, in which case the equation $z^+ = z$ is usually not mentioned explicitly.

The system (1.9) can be formulated in the form (1.1). To this end one takes

$$x = \begin{pmatrix} q \\ z \end{pmatrix} \in \mathbb{R}^{n+1}$$

and

$$\begin{aligned} C &= \bigcup_{q \in Q} (\{q\} \times C_q) \qquad & F(x) &= (0, F_q(z)), \\ D &= \bigcup_{q \in Q} (\{q\} \times D_q) \qquad & G(x) &= G_q(z). \end{aligned} \tag{1.10}$$

This construction covers the one used in Example 1.7.

Example 1.8. (Combining local and global controllers - revisited) The flow and jump maps of Example 1.7 can be written as in (1.10) by defining

$$F_q(z) := \widetilde{f}(z, \kappa_q(z)), \quad G_q(z) := \begin{pmatrix} 3 - q \\ z \end{pmatrix}.$$

The flow and jump sets were already defined in Example 1.7 as in (1.10).

The following example illustrates the use of a discrete state to explicitly model an on/off mechanism.

Example 1.9. (Thermostat) On/off control of a heater for temperature control of a room can be modeled with an explicit discrete state. The evolution of the room's temperature z can be approximated by the differential equation

$$\dot{z} = -z + z_0 + z_\Delta q \ , \tag{1.11}$$

where z_0 represents the natural temperature of the room, z_Δ the capacity of the heater to raise the temperature in the room by always being on, and q the state of the heater, which can be either 1 ("on") or 0 ("off"). Typically, it is desired to keep the temperature between two specified values $z_{\min}$ and $z_{\max}$, given in Fahrenheit units, satisfying the following relationship

$$z_0 < z_{\min} < z_{\max} < z_0 + z_\Delta \ .$$

For purposes of illustration, consider the case when $z_{\min} = 70$ and $z_{\max} = 80$. A control algorithm that attempts to keep the temperature between such thresholds is the following:

```
if q=1 and z >= 80 then
  q = 0
elseif q = 0 and z <= 70 then
  q = 1
end
```

This algorithm implements the following logic: if the heater is "on" and the temperature is larger than 80, then turn off the heater, while if the heater is "off" and the temperature is smaller than 70, then turn on the heater. With this algorithm, the temperature of the system evolves as shown in Figure 1.10, where parameters $z_0 = 60$ and $z_\Delta = 30$ were used.

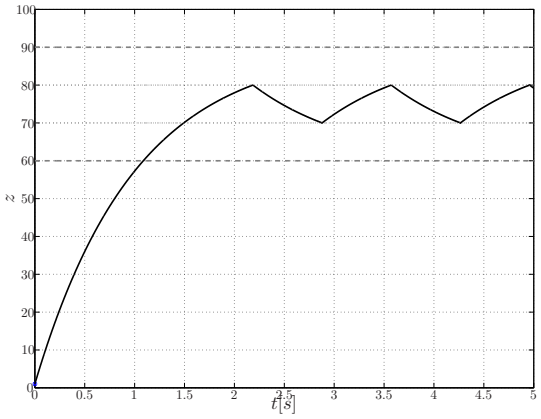


Figure 1.10: Temperature control. Evolution of temperature with control algorithm.

The control logic above results in a hybrid system of the form (1.10) with

$$\begin{array}{llll} F_q(z) & := & -z + z_0 + z_\Delta q, & G_q(z) := \begin{pmatrix} 1-q \\ z \end{pmatrix}, \\ C_0 & := & \{z : z > 70\}, & C_1 := \{z : z < 80\}, \\ D_0 & := & \{z : z \leq 70\}, & D_1 := \{z : z \geq 80\}. \end{array}$$

1.4.2 Hybrid automata

Systems with explicit "discrete states" or "logical modes" where, in each logical mode, different jump maps are specified on different subsets of a jump set, or where the jumps are modeled by an automaton, can also be molded into the framework of (1.1). Such systems are usually given by

- a *set of modes* Q, which is identified here with $\{1, 2, \ldots, q_{\max}\}$;

- a *domain* mapping Domain, giving for each $q \in Q$ a set $\mathrm{Domain}(q)$ in which the continuous state z may evolve;
- a *flow map* $f : Q \times \mathbb{R}^n \to \mathbb{R}^n$, which describes the continuous evolution of the continuous state variable z; in fact, it is enough that $f(q, \cdot)$ be defined on $\mathrm{Domain}(q)$, for each $q \in Q$;
- a *set of edges* $\mathrm{Edges} \subset Q \times Q$, identifying pairs (q, q') such that a transition from q to q' is possible;
- *guard conditions* which identify, for each edge $(q, q') \in \mathrm{Edges}$, the set $\mathrm{Guard}(q, q')$ to which the continuous state z has to belong for transitions from q to q' to be enabled;
- *reset map* $\mathrm{Reset} : \mathrm{Edges} \times \mathbb{R}^n \to \mathbb{R}^n$, which describes, for each edge $(q, q') \in \mathrm{Edges}$ and continuous state $z \in \mathbb{R}^n$, the jump of the continuous state during a transition from q to q'; in fact, it is enough for $\mathrm{Reset}(q, q', \cdot)$ to be defined on $\mathrm{Guard}(q, q')$. When the continuous variable z remains constant at jumps from q to q', the reset map $\mathrm{Reset}(q, q', \cdot)$ can be taken to be the identity.

To capture the dynamics resulting from such a set of data in the format (1.9), for each $q \in Q$, consider

$$\begin{aligned} C_q &= \mathrm{Domain}(q), \\ F_q(z) &= f(q, z) && \forall\, z \in C_q, \\ D_q &= \bigcup_{(q,q') \in \mathrm{Edges}} \mathrm{Guard}(q, q'), \\ G_q(z) &= \bigcup_{\{q' : z \in \mathrm{Guard}(q,q')\}} \begin{pmatrix} q' \\ \mathrm{Reset}(q, q', z) \end{pmatrix} && \forall\, z \in D_q. \end{aligned} \tag{1.12}$$

The values of F_q and G_q outside of C_q and D_q, respectively, can be taken to be empty. Such a definition of G_q naturally introduces set-valuedness. Indeed, $G_q(z)$ is a set whenever z is an element of two different guard sets $\mathrm{Guard}(q, q')$ and $\mathrm{Guard}(q, q'')$. In fact, $G_q(z)$ is a set in such a case even when all reset maps are identities, in other words, when z does not change during jumps.

Example 1.10. (Modeling a hybrid automaton) Consider the hybrid automaton shown in Figure 1.11, with the set of modes $Q = \{1, 2\}$; the domain map given by

$$\mathrm{Domain}(1) = \mathbb{R}_{\geq 0} \times \mathbb{R}, \quad \mathrm{Domain}(2) = \mathbb{R} \times \mathbb{R}_{\geq 0};$$

the flow map, for all $z \in \mathbb{R}^2$, given by

$$f(1, z) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad f(2, z) = \begin{pmatrix} z_2 \\ -z_1 \end{pmatrix};$$

the set of edges given by Edges $= \{(1,1),(1,2),(2,1)\}$; the guard map given by

$$\text{Guard}(1,1) = \{0\}\times\mathbb{R}_{\geq 0}, \quad \text{Guard}(1,2) = \{0\}\times\mathbb{R}_{\leq 0}, \quad \text{Guard}(2,1) = [1,3]\times\{0\};$$

and the reset map, for all $z \in \mathbb{R}^2$, given by

$$\text{Reset}(1,1,z) = (-1,0), \quad \text{Reset}(1,2,z) = z, \quad \text{Reset}(2,1,z) = -z.$$

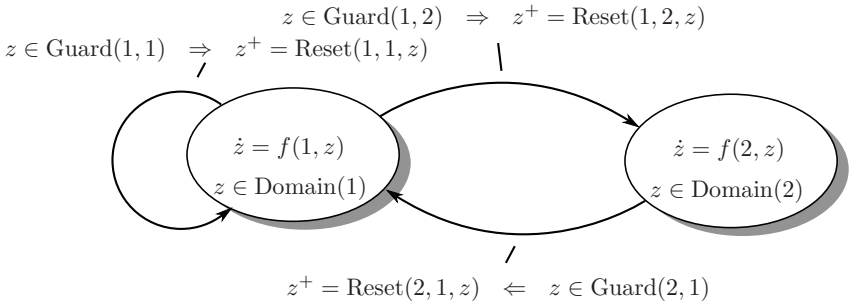


Figure 1.11: Two modes of the hybrid automaton.

The sets Guard$(1,1)$ and Guard$(1,2)$ overlap, indicating that in mode 1, a reset of the state z to $(-1,0)$ or a switch of the mode to 2 is possible from $z = 0$. Formulating this hybrid automaton as a hybrid system with explicitly shown modes (1.9) leads to $C_1 = \text{Domain}(1) = \mathbb{R}_{\geq 0} \times \mathbb{R}$, $C_2 = \text{Domain}(2) = \mathbb{R} \times \mathbb{R}_{\geq 0}$, $F_1(z) = f(1,z) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $F_2(z) = f(2,z) = \begin{pmatrix} z_2 \\ -z_1 \end{pmatrix}$, $D_1 = \text{Guard}(1,1) \cup \text{Guard}(1,2) = \{0\} \times \mathbb{R}$, a set-valued jump map G_1 given by

$$G_1(z) = \begin{cases} (1,-1,0), & \text{if} \quad z_1 = 0, z_2 > 0, \\ (1,-1,0) \cup (2,z), & \text{if} \quad z = 0, \\ (2,z), & \text{if} \quad z_1 = 0, z_2 < 0, \end{cases}$$

and $G_2(z) = (1,-z)$. Formulating the system with explicitly shown modes just described as (1.1) leads to a hybrid system in $\mathbb{R}^3$, where x_1 corresponds to q, x_2

corresponds to z_1, x_3 corresponds to z_2, and the data is given by

$$
\begin{aligned}
C &= (\{1\} \times \mathbb{R}_{\geq 0} \times \mathbb{R}) \cup (\{2\} \times \mathbb{R} \times \mathbb{R}_{\geq 0}), \\
F(x) &= \begin{cases} (0, -1, 0), & \text{if } x_1 = 1, \\ (0, x_3, -x_2), & \text{if } x_1 = 2, \end{cases}, \\
D &= (\{1\} \times \{0\} \times \mathbb{R}) \cup (\{2\} \times [1, 3] \times \{0\}), \\
G(x) &= \begin{cases} \begin{cases} (1, -1, 0), & \text{if } x_2 = 0, x_3 > 0, \\ (1, -1, 0) \cup (2, x_2, x_3), & \text{if } x_2 = 0, x_3 = 0, \\ (2, x_2, x_3), & \text{if } x_2 = 0, x_3 < 0, \end{cases} & \text{if } x_1 = 1, \\ (1, -x_2, -x_3), & \text{if } x_1 = 2. \end{cases}
\end{aligned}
$$

Figure 1.12 gives a pictorial representation of the data of the hybrid automaton as a hybrid system.

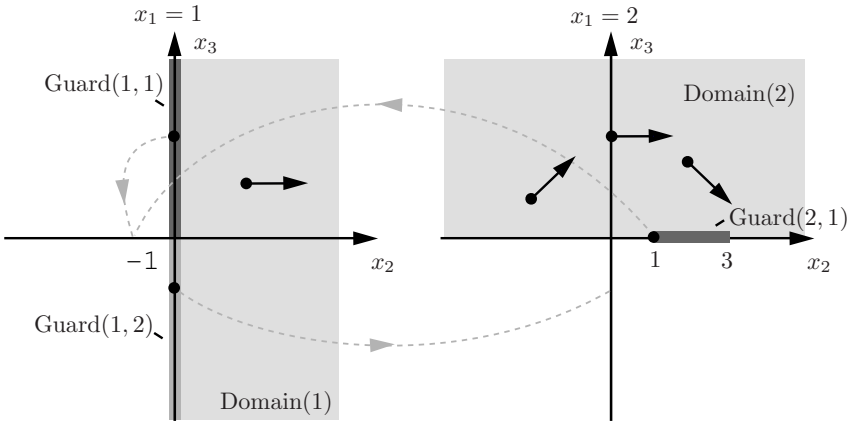


Figure 1.12: Data for the hybrid system in Example 1.10.

1.4.3 Impulsive differential equations

Consider the differential equation

$$\dot{z} = f(z),$$

for some $f : \mathbb{R}^n \to \mathbb{R}^n$, with impulses leading to instantaneous change at predetermined times $t_1, t_2, t_3, \ldots$, according to

$$\Delta z(t_i) = g(z, t_i),$$

for some $g : \mathbb{R}^n \times \mathcal{T} \to \mathbb{R}^n$ and $\mathcal{T} = \{t_1, t_2, \dots\}$. For simplicity, suppose that $\{t_i\}_{i=1}^{\infty}$ is an increasing and divergent to ∞ sequence of positive numbers. Such impulsive differential equations can be modeled in the format (1.1). The straightforward approach is to consider $x = (z, \tau) \in \mathbb{R}^{n+1}$ and the hybrid system

$$C = \mathbb{R}^n \times (\mathbb{R}_{\geq 0} \setminus \mathcal{T}) \qquad F(x) = \begin{pmatrix} f(z) \\ 1 \end{pmatrix}$$
$$D = \mathbb{R}^n \times \mathcal{T} \qquad G(x) = \begin{pmatrix} z + g(z, \tau) \\ \tau \end{pmatrix}$$

and consider initial conditions with $\tau = 0$, so that the τ-variable represents time. The discussion in Chapter 4 will show that such a formulation is not robust to perturbations. A preferred approach may be to consider $x = (z, \tau_1, \tau_2) \in \mathbb{R}^{n+2}$ and the hybrid system

$$C = \mathbb{R}^n \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \qquad F(x) = \begin{pmatrix} f(z) \\ 1 \\ -1 \end{pmatrix}$$
$$D = \mathbb{R}^n \times \mathcal{T} \times \{0\} \qquad G(x) = \begin{pmatrix} z + g(z, \tau_1) \\ \tau_1 \\ t_{i+1} - t_i \end{pmatrix} \text{ when } \tau_1 = t_i.$$

Here the variable τ_1 represents time and variable τ_2 is a timer that ensures, robustly, that flow does not occur when $\tau_1 \in \mathcal{T}$.

1.4.4 Switching systems

Broadly speaking, switching systems are continuous-time systems given by a family of differential equations, where the particular differential equation that governs the evolution of the state at any given time instant is determined by a switching rule/signal. That is, consider a set Q and, for each $q \in Q$, a function $f_q : \mathbb{R}^n \to \mathbb{R}^n$. When the switching signal σ, which takes on values in Q, remains constant, the variable z evolves continuously according to the differential equation

$$\dot{z} = f_\sigma(z).$$

When a switch in the switching signal occurs, it can be described by $\sigma^+ \in Q$. This can be modeled by the following hybrid system:

$$\begin{cases} z \in \mathbb{R}^n,\, q \in Q & \dot{z} = f_q(z) \\ z \in \mathbb{R}^n,\, q \in Q & q^+ \in Q \end{cases} \tag{1.13}$$

with the state

$$x = \begin{pmatrix} z \\ q \end{pmatrix} \in \mathbb{R}^{n+1}.$$

In (1.13), q remains constant during flow and z remains constant during jumps. More rigorously, (1.13) fits the framework of hybrid systems with

$$\begin{pmatrix} \dot{z} \\ \dot{q} \end{pmatrix} = F(x) := \begin{pmatrix} f_q(z) \\ 0 \end{pmatrix}, \quad \begin{pmatrix} z^+ \\ q^+ \end{pmatrix} \in G(x) := \begin{pmatrix} z \\ Q \end{pmatrix},$$

while $C = D = \mathbb{R}^n \times Q$.

The model (1.13) is not very helpful in practice. For example, when a switching signal is given, a switching system is just a time-varying differential equation. Similarly, questions about behavior of a switching system under all possible switching signals are better handled in a framework of differential inclusions and not through the analysis of (1.13). This is further justified by Corollary 4.24 and the discussion surrounding it.

On the other hand, a hybrid systems approach to the analysis of switching systems is useful when only switching signals from certain classes are allowed. For example, when the frequency of switching is limited, a clock state can be introduced to limit how many switches occur in a given time interval. Modeling such cases is presented in Section 2.4, where the relationships between solutions to switching systems and to hybrid systems that model them are also discussed.

1.5 NOTES

The model (1.1) or (1.2), identifying the data of a hybrid system as consisting of a flow set, flow map, jump set, and jump map, was proposed in Goebel et al. [37] and more formally stated in Goebel and Teel [40]. Models closely related to (1.1), also involving set-valued dynamics, appeared previously in Aubin and Haddad [8] and Aubin et al. [9], and concurrently in Collins [29]. Early consideration of set-valued dynamics in hybrid systems is found in Puri and Varaiya [98] and Aubin [5].

Notable early references with models of hybrid systems that distinguish between "continuous states" and "discrete states" or use the language of hybrid automata include Witsenhausen [128], Tavernini [116], Alur et al. [1], Henzinger [51], doctoral dissertations by Branicky [18] and Lygeros [78], and the book by van der Schaft and Schumacher [123]. A thorough discussion of numerous early models of hybrid systems is included in [18]. References for impulsive differential equations, as summarized here, include several books: Bainov and Simeonov [13], Lakshmikantham et al. [66], Yang [129], and Haddad et al. [45]. The standard reference for switching systems is Liberzon [73].

A different approach to modeling the behavior of mechanical systems with friction, unilateral constraints, and impacts is visible in Moreau [93], Monteiro Marques [86], and Brogliato [20], with an extensive review of mathematical literature in Stewart [114]. The approach often leads to dynamical complementarity

systems, which mix differential equations and complementarity systems common in optimization. Relation of such dynamical systems to hybrid systems is discussed by van der Schaft and Schumacher [122] and, in the linear case, by Heemels et al. [50]. See also Heemels and Brogliato [49]. Numerous references in the area are listed by Brogliato [21]. Closely related is the framework of measure-driven differential equations and inclusions; see Dal Maso and Rampazzo [30] and Silva and Vinter [111].

A detailed discussion of the Newton's cradle can be found in [123]. The flashing fireflies model in Example 1.2 draws inspiration from the hybrid model used by Strogatz and Mirollo [92]. The adjustment mechanism in the quantized control system and the typical conditions on the quantizer used in Example 1.5 are taken from Liberzon [72]. Further analysis of reset systems can be found in Beker et al. [15] and Nešić et al. [95]. The idea behind the hybrid control strategy in Example 1.7 applies to arbitrary nonlinear control systems and state-feedback laws by Prieur [97], and also motivated the hybrid control strategy in Sanfelice and Teel [106] combining state-feedback and open-loop laws. The illustration of the hybrid control strategy in this example on mode-switching control algorithms for hard disk drives follows the algorithms reported in Goh et al. [43], Venkataramanana et al. [124], and Taghirad and Jamei [115].