Chapter 2

System Models

In this chapter we introduce some basic elements of (dynamical) system modeling. A system model is an *abstraction* of an actual real-world system: a necessarily simplified representation, attempting to capture those aspects essential to explain the system's behavior. Because *we mostly reason about the model*, we quickly adopt the convention that system means the mathematical system model and we state explicitly when we refer to the physical system it represents. It is very important to keep the abstraction process in mind however, and the fact that in most cases, our mathematical abstraction is only a rough approximation of a physical system. This is generally necessary to obtain tractable models useful for design purposes. In particular, this issue is as important when verifying software written in a programming language with precise semantics, or even digital circuits, where fabrication processes can be tightly controlled to correspond to models precisely. We will have more to say about this point in later chapters when we discuss verification and certification issues.

2.1 General and Input-Output Systems

A general dynamical system Σ is simply a set of signals [PW98]. Namely, Σ is defined as a triple $(\mathbb{T}, \mathbb{W}, \mathcal{B})$, where $\mathbb{T}$ is a subset of $\mathbb{R}$ called the *time axis*, $\mathbb{W}$ is a set, and $\mathcal{B}$ is a subset of the function space $\mathbb{W}^{\mathbb{T}}$ called the *behavior*. Note that $\mathbb{W}^{\mathbb{T}}$ denotes the set of maps from $\mathbb{T}$ to $\mathbb{W}$. It can be quite interesting to develop a general systems theory from such an abstract point of view [PW98], but we will adopt a somewhat more operational point of view and directly deal with some important classes of behavioral descriptions. We can restrict our attention to a strict subset $\mathbb{T}$ of $\mathbb{R}$, in order to simplify the analysis or because the system behavior at other instants is not interesting for our purposes. This leads for example to the notion of discrete-time (DT) systems for $\mathbb{T} = \mathbb{Z}$ or $\mathbb{T} = \mathbb{N}$, which are usually obtained by sampling continuous-time (CT) systems at a discrete set of points, typically regularly spaced, as discussed in chapter 3. When combining DT and CT models however, or several DT systems with

Figure 2.1: A system with input u and output y.

different sampling periods, we can always embed the time axes back into $\mathbb{R}$ for reasoning about the global system.

Input-Output Systems

We call any function space $\mathbb{S}^{\mathbb{T}}$, with $\mathbb{S}$ a set, a *signal space*, and its elements are called *signals*. Given two signal spaces $\mathcal{U} = \mathbb{U}^{\mathbb{T}}, \mathcal{Y} = \mathbb{Y}^{\mathbb{T}}$, an *input-output* system is a subset of $\mathcal{U} \times \mathcal{Y}$, i.e., a binary *relation* between the sets $\mathcal{U}$ and $\mathcal{Y}$. Then $\mathcal{U}$ is called the input signal space and $\mathcal{Y}$ the output signal space. This is a particular class of behavioral models, with $\mathbb{W} = \mathbb{U} \times \mathbb{Y}$ and $\mathcal{B} \subset \mathcal{U} \times \mathcal{Y}$, where we have explicitly stated which signals are considered as inputs and which signals are considered as outputs, although this choice is not necessarily natural. Hence an input-output system is interpreted as a set of possible input/output signal pairs $(u, y) \in \mathcal{U} \times \mathcal{Y}$. Note that we associate a *set* (possibly empty) of possible output signals $\{y(\cdot)\}$ to an input signal $u(\cdot)$. This allows us to take into account the presence of unknown or unmodeled *initial conditions*, as well as other sources of nondeterminism in the system such as noise.

Input-output methods view systems as black boxes that relate external signals, independently of any internal state-space representation of these systems, see Fig. 2.1. They have important applications in linear and nonlinear control, e.g. in robust control, and are particularly useful for the analysis of large-scale complex and interconnected systems. Pioneering work in this area was done by Zames, Sandberg, Popov, Desoer and others. Standard references on input-output methods include [Wil71, DV09, Saf80].

Signal Spaces

The function spaces $\mathcal{U}, \mathcal{Y}$ of most interest for control purposes are usually the L^p spaces[1], $1 \leq p \leq \infty$. To define these spaces, we equip a set $\mathbb{S}$ of signal values with a norm $\|\cdot\|$ (typically the Euclidean norm on $\mathbb{S} = \mathbb{R}^n$), and we fix a measure μ on $\mathbb{T}$. Then we define, for $p \in [1, \infty)$

$$L^p(\mathbb{T}, \mathbb{S}, \mu) := \left\{ x \in \mathbb{S}^{\mathbb{T}}, \mu - \text{measurable} \,\middle|\, \int_{\mathbb{T}} \|x(t)\|^p \mu(dt) < \infty \right\},$$

[1] also denoted $L_p, \mathcal{L}^p, \mathcal{L}_p, \mathcal{L}_p^n, \ldots$ in various other references. We follow here the more or less standard notation found in analysis textbooks.

and

$$L^\infty(\mathbb{T},\mathbb{S},\mu) := \left\{ x \in \mathbb{S}^{\mathbb{T}}, \mu - \text{measurable} \middle| \operatorname*{ess\,sup}_{\mathbb{T}} \|x(t)\| < \infty \right\}.$$

In the usual contexts where these spaces arise, we have $\mathbb{S} = \mathbb{R}^n$, for some $n \in \mathbb{N}$, and $\mathbb{T} = \mathbb{R}$ or $\mathbb{R}_+$ is fixed with μ the Lebesgue measure, in which case we denote $L^p(\mathbb{T},\mathbb{S},\mu) =: L^p_n(\mathbb{T})$, where $\mathbb{T}$ can be omitted if it is understood. Hence we have for example

$$L^p_n(\mathbb{R}_+) = \left\{ f : \mathbb{R}_+ \to \mathbb{R}^n \middle| \int_0^\infty \|f(t)\|^p dt < \infty \right\},$$

and the space L^∞ consists of functions that are essentially bounded, i.e., bounded on their domain of definition except on a set of Lebesgue measure zero.

If $\mathbb{T}$ is a discrete set such as $\mathbb{Z}$ or $\mathbb{N}$ with μ the counting measure, then we denote $L^p(\mathbb{T},\mathbb{R}^n,\mu) =: \ell^p_n$, so that, in the case $\mathbb{T} = \mathbb{Z}$ for example,

$$\ell^p_n := \left\{ \{x_k\}_{k\in\mathbb{Z}} \middle| \left(\sum_{k=-\infty}^{\infty} \|x_k\|^p \right)^{1/p} < \infty \right\}, \quad p \in [1,\infty)$$

$$\ell^\infty_n := \left\{ \{x_k\}_{k\in\mathbb{Z}} \middle| \|x_k\| < \infty, \forall k \in \mathbb{Z} \right\}.$$

The ℓ^p spaces arise in the context of discrete-time (DT) systems, the L^p spaces in the context of continuous-time (CT) systems.

The L^p and ℓ^p spaces, $p \in [1,\infty]$, are *normed* vector spaces (see the remark below theorem 2.1.1). The p-norm $\|\cdot\|_p$ or $\|\cdot\|_{L^p}$ of $u : [0,\infty) \to \mathbb{R}^n$ is defined as

$$\|u\|_p := \left(\int_0^\infty \|u(\tau)\| d\tau \right)^{1/p}, \quad 1 \le p < \infty$$

$$\|u\|_\infty := \operatorname*{ess\,sup}_{\tau\ge 0} \|u(\tau)\|,$$

and similarly for other domains of definition, e.g. $u : \mathbb{R} \to \mathbb{R}^n$, and discrete-time sequences. With these norms, the L^p spaces are in fact *complete*, i.e., Banach spaces. An important consequence is that contractive maps on these spaces have a (unique) fixed point. Moreover, L^2 is a Hilbert space, with inner product

$$\langle f, g \rangle = \int f(t)^T g(t) dt,$$

and similarly with

$$\langle x_1, x_2 \rangle = \sum x_{1,k}^T \, x_{2,k},$$

for ℓ^2.

The most important L^p spaces are those obtained for $p = 2$, capturing the notion of power and energy, $p = \infty$, capturing the worst case behavior of signals, and $p = 1$.

Relations Between L^p Spaces

Theorem 2.1.1. *[Hölder, Schwarz and Minkowski Inequalities] For $p, q \in [1, \infty]$, such that*

$$\frac{1}{p} + \frac{1}{q} = 1, i.e., q = \frac{p}{p-1},$$

if $f \in L^p$, $g \in L^q$, then $fg \in L^1$, and we have the Hölder inequality

$$\|fg\|_1 \leq \|f\|_p \|g\|_q. \tag{2.1}$$

When $p = q = 2$, this inequality is known as the Cauchy-Schwarz inequality. Moreover, if $f, g \in L^p$, then $f + g \in L^p$ and we have the Minkowski inequality

$$\|f + g\|_p \leq \|f\|_p + \|g\|_p,$$

for all $p \in [1, \infty]$.

Remark. Note that the Minkowski inequality is crucial in establishing that L^p is in fact a vector space, by showing that it is stable under addition.

Now suppose that the domain of definition of f, denoted I, has *finite Lebesgue measure* $|I| < \infty$. First, if $f \in L^\infty$, then

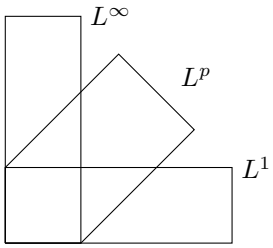
$$\left(\int_I \|f(t)\|^p \right)^{1/p} \leq \|f\|_\infty |I|^{1/p},$$

so that $f \in L^p$, for all $p \in [1, \infty]$. Next if $1 \leq r < s < \infty$ and $f \in L^s$, then by the Hölder inequality (2.1) with $p := s/r$ and $q = p/(p-1)$, we have

$$\int_I \|f(t)\|^r dt = \int_I \|f(t)\|^r \cdot 1 \, dt \leq \left(\int_I \|f(t)\|^{r(s/r)} dt \right)^{r/s} |I|^{1/q},$$

so that $\|f\|_r \leq \|f\|_s |I|^{1/qr}$. In other words, if I has finite measure, then integrability of $\|f\|^p$ for a function f is more restrictive for larger values of p, and we have $L^s(I) \subset L^r(I)$ for $1 \leq r < s \leq \infty$.

Typically for us however, the domain of definition of the signals (say $\mathbb{R}_+$ or $\mathbb{R}$) does not have finite measure, and these inclusions do not hold in general any more. For example, $t \mapsto 1$ in in $L^\infty(\mathbb{R}_+)$ but not in $L^p(\mathbb{R}_+)$ for any $p < \infty$. In fact, for any $1 < p < \infty$, we have the following Venn diagram, where all drawn subsets are non empty



Note in particular the following fact.

Proposition 2.1.2. *If $f : \mathbb{R}_+ \to \mathbb{R}$ and $f \in L^1 \cap L^\infty$, then $f \in L^p$ for $p \in [1, \infty]$.*

Proof. Denote $I := \{t | \|f(t)\| \geq 1\}$. Then since $f \in L^1$, the Lebesgue measure of I is finite and since $f \in L^\infty$ we must then have $\int_I \|f(t)\|^p dt < \infty$. Now with I^c the complement of I, we have

$$\infty > \int_{I^c} \|f(t)\| dt \geq \int_{I^c} \|f(t)\|^p dt, \quad \text{for all } p \in [1, \infty).$$

Together, these two observations give the conclusion. □

Example 2.1.1. The function $t \mapsto \frac{1}{1+t}$, from $\mathbb{R}_+$ to $\mathbb{R}$, is in L^∞ and L^2 but not in L^1. Finish populating the Venn diagram above with examples of functions, for $p = 2$.

Extended L^p Spaces

Integrability of signals on $\mathbb{R}_+$ or $\mathbb{R}$ is often too strong a requirement for our purposes, e.g. when dealing with unstable systems. For this reason, we are led to consider locally integrable functions, i.e., functions that are integrable on every bounded interval. For $T \geq 0$, define the linear operator P_T by

$$(P_T f)(t) := f_T(t) := \begin{cases} f(t), & 0 \leq t \leq T, \\ 0, & t > T. \end{cases}$$

P_T is a projection since $P_T^2 = P_T$. We say that f_T is obtained by *truncating* f at T. Define the extended space L_e^p by

$$L_{n,e}^p(\mathbb{R}_+) = \{f : \mathbb{R}_+ \to \mathbb{R}^n | \|f_T\|_p < \infty, \forall T \geq 0\}.$$

That is, $f \in L_e^p$ if $f_T \in L^p$ for all $T \geq 0$.

Example 2.1.2. The function $t \mapsto \frac{1}{1+t}$ is not in L^1 but it is in L_e^1.

Note the following facts

(i) $\forall f \in L_e^p$, the map $T \mapsto \|f_T\|_p$ is monotonically increasing.

(ii) $\forall f \in L^p$, $\|f_T\|_p \to \|f\|_p$ as $T \to \infty$.

Systems

Causality

Most models used in systems theory are nonanticipative.

Definition 2.1.1. A mapping $H : L_e^p \to L_e^p$ is *causal* (or nonanticipative) if $P_T H P_T = P_T H$ for all $T \geq 0$, i.e.,

$$(Hu_T)_T = (Hu)_T, \ \forall T \geq 0. \tag{2.2}$$

Example 2.1.3. Let $h \in L^1$ be the impulse response of a linear time-invariant system $H : L^2 \to L^2$:

$$Hu = h * u, \ \ \forall u \in L^2,$$

i.e.,

$$(Hu)(t) = \int_{-\infty}^{\infty} h(t-\tau)u(\tau)d\tau, \ \ \forall t \in \mathbb{R}.$$

Then H is causal if and only if $h(t) = 0$ almost everywhere on $(-\infty, 0)$.

In words, if H is causal then the responses to the inputs u and u_T are the same up to time T, i.e., the output does not depend on the future values (for $t > T$) of the inputs. Note that an equivalent definition of a causal system is

$$\forall u_1, u_2 \in L^p_e, P_T u_1 = P_T u_2 \Rightarrow P_T H u_1 = P_T H u_2. \tag{2.3}$$

that is, two input signals u_1 and u_2 which are identical on $[0, T]$ give identical output responses Hu_1 and Hu_2 on $[0, T]$. Indeed, if H is causal according to definition 2.1.1, then $P_T H u_1 = P_T H P_T u_1 = P_T H P_T u_2 = P_T H u_2$. Conversely, if (2.3) is satisfied, then because $P_T(P_T u) = P_T u$, we have $P_T H P_T u = P_T H u$, i.e. (2.2).

Remark. Note that systems given in a standard state space form, e.g.

$$\begin{aligned} \dot{x} &= f(x, u, t) \\ y &= h(x, u, t) \end{aligned}$$

are always causal, when viewed as input-output systems $u(\cdot) \mapsto y(\cdot)$.

Induced Norms of Linear Maps

A system $H : \mathcal{U} \to \mathcal{Y}$ is linear if and only if

$$H(\alpha_1 u_1 + \alpha_2 u_2) = \alpha_1 H u_1 + \alpha_2 H u_2, \ \ \forall u_1, u_2 \in \mathcal{U}.$$

It is well-know that if $H : L^p \to L^q$ is linear, then there exists a function $h(\cdot, \cdot)$ such that

$$(Hu)(t) = \int_{-\infty}^{+\infty} h(t, \tau)u(\tau)d\tau.$$

Moreover if H is time-invariant, then in fact we have

$$(Hu)(t) = \int_{-\infty}^{+\infty} h(t-\tau)u(\tau)d\tau = (h * u)(t).$$

For the set $\mathcal{L}(\mathcal{U}, \mathcal{Y})$ of linear operators between the normed vector spaces $\mathcal{U}$ and $\mathcal{Y}$ with norms $\|\cdot\|_{\mathcal{U}}$ and $\|\cdot\|_{\mathcal{Y}}$, we define the induced norm on $\mathcal{L}(\mathcal{U}, \mathcal{Y})$ by

$$||H|| = \sup_{u \neq 0} \frac{\|Hu\|_{\mathcal{Y}}}{\|u\|_{\mathcal{U}}} = \sup_{\|u\|_{\mathcal{U}}=1} \frac{\|Hu\|_{\mathcal{Y}}}{\|u\|_{\mathcal{U}}}.$$

This provides a norm for linear systems. Indeed, consider H a SISO causal linear time-invariant system with impulse response h. Assume that $\|h\|_1 < \infty$. Then H maps L^∞ to L^∞ and its induced norm as a map of $\mathcal{L}(L^\infty, L^\infty)$ is $\|H\|_{\infty\to\infty} = \|h\|_1$. One part of this statement is easy, namely since

$$|(Hu)(t)| \le \int_0^t |h(t-\tau)u(\tau)|d\tau \le \|u\|_\infty \int_0^t |h(t-\tau)|d\tau \le \|h\|_1\|u\|_\infty,$$

we have $\|H\|_{\infty\to\infty} \le \|h_1\|$. The bound is approached arbitrarily closely by an appropriate choice of input signals. We can also consider H as a map in $\mathcal{L}(L^2, L^2)$. Note that since $\|h\|_1 < \infty$, its Fourier transform $\hat{h}$ is well-defined and one can show that the induced norm of H in this case is $\|H\|_{2\to2} = \sup_{\omega\in\mathbb{R}} |\hat{h}(j\omega)|$. The reader should refer to standard references on linear systems for more details and the corresponding MIMO results.

L^p Stability

Definition 2.1.2. A mapping $H : L_e^p \to L_e^p$ is L^p stable if there exists a class $\mathcal{K}$ function α, defined on $[0, \infty)$, and a nonnegative constant β, such that

$$\|(Hu)_T\|_p \le \alpha(\|u_T\|_p) + \beta, \ \ \forall u \in L_e^p \text{ and } T \ge 0.$$

We say that H is *finite gain L^p stable* if there exist nonnegative constants γ and β such that

$$\|(Hu)_T\|_p \le \gamma\|u_T\|_p + \beta, \ \ \forall u \in L_e^p \text{ and } T \ge 0. \tag{2.4}$$

The smallest γ such that (2.4) holds is called the gain of H, denoted $\gamma(H)$

$$\gamma(H) = \inf\{\gamma \in \mathbb{R}_+ | \exists \beta \text{ s.t. (2.4) holds}\}.$$

Remark. Clearly a system that is finite-gain L^p stable is L^p stable.

Note that if H is causal, then condition (2.4) can be replaced by

$$\|Hu\|_p \le \gamma\|u\|_p + \beta, \ \ \forall u \in L^p \ \ (\text{not } L_e^p!), \tag{2.5}$$

where the T subscripts are dropped. Indeed (2.4) always implies (2.5). Conversely, if (2.5) holds, then it holds for functions of the form $P_T u_1$, and then use $P_T H P_T = P_T H$.

Example 2.1.4. Let H be a causal linear time-invariant system with integrable impulse response h, i.e., $\|h\|_1 < \infty$, as in the previous paragraph. Then H is finite-gain L^1 and L^∞ stable with gain $\|h\|_1$ in both cases, hence this assumption guarantees that H is finite gain L^p stable for all $p \in [1, \infty]$ by Proposition 2.1.2. In particular, the assumption also ensures that the Fourier transform $\hat{h}$ of h is well defined, and it is well-known then that H is finite gain L^2 stable with gain $\sup_{\omega\in\mathbb{R}} |\hat{h}(j\omega)|$.

Remark. The definition of the gain of a system given above is perhaps not very satisfying, as system responses depend on the initial condition x_0, which does not appear in the definition. Hence one a priori can obtain a different gain for different initial conditions. In many cases, it turns out that the bias term β allows us to obtain a gain value independent of the initial condition (e.g. for linear system). An alternative definition of the gain of a (possibly multivalued) system can be given as follows. The L^p gain of an input/output system $\mathcal{S} \subset L_e^p(\mathbb{R}^m) \times L_e^p(\mathbb{R}^q)$ is

$$\inf \left\{ \gamma \geq 0 \Big| \inf_{T \geq 0} \int_0^T \gamma^p \|u(t)\|^p - \|y(t)\|^p dt > -\infty, \forall (u,y) \in \mathcal{S} \right\}.$$

Incremental L^p stability

Definition 2.1.3. A mapping $H : L_e^p \to L_e^p$ is said to be *incrementally finite gain L^p stable* if

1. $H(0) \in L^p$, where 0 is the identically zero input.
2. For all $T > 0$ and $u_1, u_2 \in L_e^p$, there exists $\gamma > 0$ such that

$$\|(Hu_1 - Hu_2)_T\|_p \leq \gamma \|u_{1,T} - u_{2,T}\|_p.$$

Note that an incrementally finite gain stable operator is also finite gain stable since

$$\|(Hu)_T\|_p \leq \gamma \|u_T\|_p + \|H(0)\|_p.$$

The two notions coincide for linear systems.

Small-Signal Finite Gain L^p Stability

Definition 2.1.4. A mapping $H : L_e^p \to L_e^p$ is small signal finite gain L^p stable if there exists $r > 0$ and $\gamma, \beta \geq 0$ such that

$$\|u_T\|_\infty < r \Rightarrow \|(Hu)_T\|_p \leq \gamma \|u_T\|_p + \beta, \ \ \forall T \geq 0.$$

2.2 State Space Model Realizations of Input-Output Systems

As mentioned earlier, input-output models are black-box models. They relate input and output signals independently of any description of the internal mechanisms producing these signals. As a result, they are of fundamental importance to reason about connected and large-scale systems, since the size and complexity of an overall model grows slowly with the number of components in this framework. To carry actual system-related computations however, it is often necessary to describe a component's internal mechanisms by means of state space models. First, recall the following more-or-less rigorous definition of state.

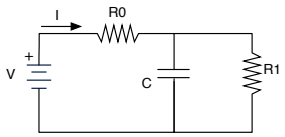


Figure 2.2: An electrical circuit.

Definition 2.2.1. The *state* of a system at time t is the information required at time t so that the output for all $t' \geq t$ is determined from this information and the knowledge of the inputs for $t' \geq t$.

State space models can be obtained directly by reasoning from first principles, or by fitting parameters via black-box system identification procedures, which analyze output signals obtained from test input signals. In the later case, the approximate nature of the models is clear, since often no attempt is made to justify the chosen structure of a state space model beyond the fact that it exhibits reasonable predictive power. Most state space models considered in these notes however, in particular those used for computing components and software, rely on first principle reasoning. Models of software and circuits can be quite precise, even though in this domain abstractions exist as well (e.g. when assuming that digital signals switch instantaneously). But for most physical systems controlled by these software and hardware components, first principle reasoning typically leaves certain behaviors unmodeled (e.g. high-order dynamics) in order to obtain reasonably tractable models. It is important to make sure that these unmodeled components do not invalidate the system design, either by employing robust design methods or by using a posteriori testing procedures.

Consider for example the electrical circuit shown on Fig. 2.2, where we are interested in the relationship between I as an input signal and V as an output signal [2]. If we did not know the composition of the circuit, we could try various test signals I and observe the behavior of the signal V, in order to fit the parameters of a model. On the other hand, if we have access to the composition of this circuit and know the parameters of its various components, we can model the relationships between signals as

$$\frac{dV_C}{dt} = -\frac{1}{R_1 C} V_C + \frac{1}{C} I \tag{2.6}$$

$$V = R_0 I + V_C, \tag{2.7}$$

where V_C is the voltage across the capacitor. It is likely that the physical components do not behave exactly as predicted by these equations and that the knowledge of the values of the capacitances and resistances is not perfect (recall

[2] This choice of input and output signals is arbitrary. See [PW98] for an intrinsic way of partitioning signals between free inputs and bound outputs.

for example that resistance values are usually associated with a tolerance), but for many purposes such a model is precise enough. Here the variable V_C is an internal state variable that is used to explain the input-output relation between I and V.

Linear Time-Invariant State Space Models

Continuous-Time Systems

Generalizing (2.6), (2.7), a continuous-time *linear time invariant* (LTI) state space model defines a system with m-dimensional input u and p-dimensional output y related by

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{2.8}$$
$$y(t) = Cx(t) + Du(t), \tag{2.9}$$

for $t \in \mathbb{T} = \mathbb{R}_+$ or $\mathbb{R}$ and $x(0) = x_0 \in \mathbb{R}^n$, where A, B, C, D are real matrices of appropriate dimensions. In general, solutions of differential equations $\dot{x} = f(x,t)$ are understood in the sense of Caratheodory, i.e., as absolutely continuous functions satisfying the integral equation

$$x(t) = x(0) + \int_0^t f(x(\tau), \tau) d\tau,$$

where $x(0) \in \mathbb{R}^n$ is an "initial condition" parameter. Hence such a solution satisfies the differential equation almost everywhere. A common notation for the system $\mathcal{G}$ defined by (2.8), (2.9) is

$$\mathcal{G} = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right].$$

The *transfer matrix* of this linear state space model is the maximal analytical extension of

$$G(s) = D + C(sI - A)^{-1}B,$$

defined for all complex s, except possibly at some eigenvalues of A. In addition,

$$G(\infty) := \lim_{|s| \to \infty} H(s) = D.$$

Discrete-Time Systems

DT LTI state space models are defined analogously for $\mathbb{T} \subset \mathbb{Z}$, replacing differential equations by difference equations, to get

$$x[k+1] = Ax[k] + Bu[k], \tag{2.10}$$
$$y[k] = Cx[k] + Du[k]. \tag{2.11}$$

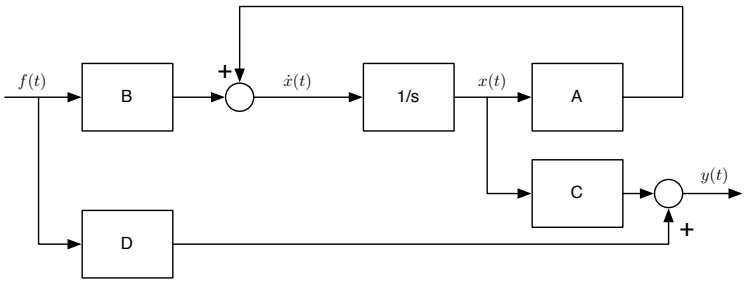


Figure 2.3: An LTI state space model.

Nonlinear Continuous State Space Models

Physical systems exhibiting certain types of behaviors (e.g. chaos, bifurcations, etc.) cannot be modeled by linear systems. In this case, we might want to use a nonlinear state space model, perhaps time-varying, e.g. of the continuous-time form

$$\dot{x} = f(x, u, t) \tag{2.12}$$
$$y = h(x, u, t), \tag{2.13}$$

with $x(0)$ given. Additional assumptions are made on f to obtain reasonable notions of solutions. Similarly, in discrete-time we can model a nonlinear system using the difference equation

$$x_{k+1} = f(x_k, u_k, k), \quad x_0 \text{ given},$$
$$y_k = g(x_k, u_k, k).$$

In general, nonlinear systems can exhibit extremely complex behaviors. In fact, very simple one dimensional discrete recurrences can exhibit chaotic behavior, e.g. the logistic map

$$x_{k+1} = r x_k (1 - x_k),$$

for certain values of r. As a result, one must generally start by identifying special classes of interesting nonlinear systems relevant for a problem of interest and amenable to analysis, and develop analysis and synthesis tools dedicated to these special cases. Moreover, an often adequate solution from the computational and robustness point of view is to try to approximate a nonlinear system by a linear time-invariant one plus some perturbation term that is only approximately characterized. This is the basic principle of the field of robust control.

Nondeterminism

The previous linear and nonlinear models produce essentially a unique output once an initial condition and an input are given. More realistic models attempt

to directly include some notion of perturbation, either in a stochastic or deterministic framework. In such models, dynamics can be perturbed, and output measurements can be noisy. One way of capturing uncertainty is by using differential inclusions rather than differential equations as in (2.12), (2.13). More precise information can be given in the form of probabilistic transition systems and output maps. See also definition 2.2.2. For example, we can have a DT model

$$x_{k+1} = f(x_k, u_k, w_k, k), \quad x_0 \text{ given}, \tag{2.14}$$

$$y_k = g(x_k, u_k, v_k, k), \tag{2.15}$$

where w_k and v_k are uncertain vectors perturbing the dynamics and observations respectively.

Transition Systems and Automata

In computer science, system models (for software, hardware and physical systems) are often expressed in the form of *transition systems*. Many variations of such models exist, depending on the problem of interest, and the terminology also tends to differ slightly depending on the application area. In particular, transition systems are sometimes also called *automata*, but many authors reserve the latter term for certain specific types of transition systems (e.g. finite number of states and transitions, resulting in a finite-state machine, presence of initial and final or marked states). From the computational point of view, the situation is roughly as follows: in engineering (in particular control engineering here), LTI models are preferred because many things about them can be computed using linear algebra (this also includes convex optimization methods such as semi-definite programming). Computer science tends to focus on finite finite automata for which many algorithms are also available, e.g. based on the graph representation of the automata or more advanced data structures such as Binary Decision Diagrams (BDDs).

The theoretical investigations of the properties of automata, e.g. related to the language generated by an automaton, to reachability, safety and liveness properties, or various notions of compositions, have focused on somewhat different issues than classical control theory, and these ideas are useful in the study of NECS. Moreover, differential and difference equations have traditionally been used in control engineering to model *time-driven processes*. In the analysis of complex systems involving say human operators and computers, it is important to also be able to efficiently model asynchronous occurrences of discrete events, resulting in Discrete-Event Systems [CL08] and *event-driven processes*, or hybrid systems combining time-driven and event-driven dynamics. Finally, automata-based models are convenient to study general finite state models, whereas the models presented so far are mostly used for continuous state spaces. Note however that difference equation models are useful to study certain discrete models with special structure as well, e.g. queueing systems and even general types of finite state Hidden Markov Models [EAM94].

We start with the following extremely general notion of transition system, a constrain it later as computational aspects become important.

Definition 2.2.2. A (Moore) transition system TS is a tuple $(X, X_0, U, \longrightarrow, Y, L)$ consisting of

- a set of states X (not necessarily finite or even countable)
- a set of initial states $X_0 \subseteq X$
- a set of inputs U (not necessarily finite or even countable)
- a transition relation $\longrightarrow \subseteq X \times U \times X$
- a set of outputs Y
- a set-valued output map $L : X \to 2^Y$.

The transition system is called *finite* if X and Y are finite. The system starts in one of the initial states (non-determinism is introduced if X_0 consists of more than one state), and evolves according to the transition relation. Namely, a transition $(x, u, x') \in \longrightarrow$, denoted in the following $x \xrightarrow{u} x'$, means that the system initially in state x is subject to an input or event u and moves to the state x'. When the transition system is finite, we can represent it as a graph with nodes associated to states and edges associated to triplets of the transition relation and labeled by the input value. It is sometimes useful to have a special input label, denoted τ, with $U = U' \cup \{\tau\}$ and $\tau \notin U'$, to denote transitions occurring in the absence of an observable input, called silent transitions (these transitions are similar to ϵ-transitions in the standard terminology for nondeterministic automata). The set U' then denotes the observable inputs. Note that we allow non-determinism in the dynamics, since there can be several states $x' \in X$ associated to a given pair (x, u) by the transition relation. In the graph representation, this means that we allow several edges labeled with the same input value u starting from a node x. Nondeterministic choices in particular are useful to model the parallel execution of independent activities, unknown or unpredictable environments (e.g., a human user), as well as for abstraction purposes and to leave certain aspects of the system unspecified, e.g. in early design phases to leave several options for the possible final behaviors.

The set-valued map L means that in state x, there is a *set* of possible observable outputs $L(x)$, with possible non-determinism introduced to model sensor uncertainty for example. If L is a single-valued map (deterministic observations), with output y in state x, then we use the standard notation $L(x) = y$ instead of $L(x) = \{y\}$. The case of full state observation corresponds to $L(x) = x$, i.e., L is the identity map. Allowing non-determinism in the output map is done for convenience, but is not strictly necessary. We then obtain models similar to (2.14), (2.15) and to the way Hidden Markov Models (HMM) are formulated in the probabilistic setting for example. However, we could have pushed all the non-determinism in the transition relation, so that

starting from a transition $x \xrightarrow{u} x'$ with say $L(x') = \{y_1, y_2\}$, we split the transition into two transitions $x \xrightarrow{u} x'_1$ and $x \xrightarrow{u} x'_1$ with $L(x'_1) = y_1$ and $L(x'_2) = y_2$.

Remark. The name Moore transition system is not really standard, and is used here by analogy with the definition in the finite state case of Moore automaton, which is just a finite state automaton with outputs associated to states. With input-output transition systems in general, there is an alternative way of introducing outputs, namely by associating outputs to transitions rather than states. Both conventions are used (including, e.g., in the theory of controlled Markov chains). Then the transition relation becomes a relation on $X \times U \times Y \times X$, denoted $x \xrightarrow{u/y} x'$, and in the graphical representation the edges are now labeled by input-output pairs. The corresponding automata are usually called *Mealy automata*. It is not hard to see that the two types of representations are equivalent.

Exercise 1. Show that Moore and Mealy transition systems are equivalent ways of describing input-output behaviors, and in fact both can be represented as automata with trivial output set, i.e., with Y a singleton. Again, this can be exploited to develop general computational tools but is not necessarily convenient for modeling purposes. Hint: starting say with a Moore transition system, convert it to an equivalent Mealy transition system, and then reinterpret $U \times Y$ as the new input set.

Note again the generality of the definition 2.2.2. For example, one can represent a system described by a differential inclusion or a controlled differential equation using such a transition system. For simplicity, consider a time-invariant control system obeying a given controlled differential equation. Define the set of inputs as the set of pairs $(T, u|_{[0,T]})$, where $u|_{[0,T]})$ denotes the restriction of an admissible input signal to the interval $[0, T]$. Then two states x and x' are related by a transition if there exists $T \geq 0$ and an input signal $u|_{[0,T]}$ driving the state from x to x'. Note here that we have an uncountable number of states and generally an uncountable number of transitions out of every state. By fixing T in the input set to a fixed value, one also obtains a discrete-time, regularly sampled version of the continuous-time system. Of course, by using this device we are forgetting the nice algebraic and computational properties potentially associated with the differential equation, hence no progress has been made unless we can simplify again the resulting transition system, say to a computationally tractable finite system. This idea is explored later when we discuss notions of equivalence and approximation of transition systems, in particular simulation and bisimulation relations.

Also, keep in mind that the output set Y in definition 2.2.2 can be quite general. In particular, it can consist of atomic logic propositions that are true for the associated states. L is then called a *labeling function*. Then, given Φ a propositional logic formula, the state x verifies Φ if the evaluation of $L(x)$ makes the formula true (denoted $L(x) \models \Phi$). This view can be exploited to specify and verify properties along the possible trajectories of a system modeled

as a transition system, an idea discussed in a later chapter and known as *model-checking* [BK08].

Remark. Another widely used term is *Kripke structure*. Normally, a Kripke structure is a finite state Moore transition system with a single label in the input vocabulary (i.e., the input labels are irrelevant) and with L a labeling function (i.e., the outputs are logic propositions).

We finish this section on transition systems with a few general definitions. For $x \in X$ and $u \in U$, define the sets of direct u-successors and direct successors of x as

$$Post(x,u) = \{x' \in X | x \xrightarrow{u} x'\}, \;\; Post(x) = \bigcup_{u \in U} Post(x,u).$$

Similarly, the sets of direct u-predecessors and direct predecessors of x are defined as

$$Pre(x,u) = \{s' \in X | x' \xrightarrow{u} x\}, \;\; Pre(x) = \bigcup_{u \in U} Pre(x,u).$$

The notation is extended to sets in a straightforward way. For $S \subset X$,

$$Post(S,u) = \bigcup_{x \in S} Post(x,u), \;\; Post(S) = \bigcup_{x \in S} Post(x),$$
$$Pre(S,u) = \bigcup_{x \in S} Pre(x,u), \;\; Pre(S) = \bigcup_{x \in S} Pre(x).$$

A state x of a transition system is called *terminal* is it does not have any outgoing transition, i.e., if $Post(x) = \emptyset$. For reactive systems, the types of systems we are interested in for control applications and are supposed to run continuously, this is generally something that must be detected and avoided. In such a state, the system is said to be in *deadlock*. Finally, we have the following notions of determinism.

Definition 2.2.3. A transition system is called *input-deterministic* if $L|_{X_0}$ is injective, and $|Post(x,u)| \leq 1$ for all states $x \in X$ and inputs $u \in U$. Hence observing the initial output and all the subsequent inputs allows us to reconstruct the state trajectory with certainty. It is called *output-deterministic* if $L|_{X_0}$ is injective and $|Post(x) \bigcap \{x' \in X | L(x') = A\}| \leq 1$ for all states $x \in X$ and output sets $A \subset Y$. In other words, for any state x and inputs u', u'' with $x \xrightarrow{u'} x'$ and $x \xrightarrow{u''} x''$ and the same observations $L(x') = L(x'')$, we must have $x' = x''$. Hence if L is a single valued map, observing the sequence of outputs allows us to reconstruct the state trajectory with certainty. Similarly if L is a labeling function and we can observe in each state all the propositions that are true.

Timed and Hybrid Automata

For all practical purposes, the transition systems of definition 2.2.2 cannot be manipulated at this level of generality. Indeed, in general as modeling expressiveness increases, so do the associated computational costs. Because the definition places no restriction on the state space or the transitions, essentially any type of system can be modeled in this framework, but one cannot develop computational tools to answer basic questions about or even just store such general models. Finite transition systems can be manipulated relatively easily on a computer, but seem a priori insufficient for control purposes. In particular, it is not immediately clear how to record the precise timing of asynchronous events using state transition systems with a finite or even countable state space. When used directly for modeling purposes, finite state transition systems usually only record the ordering of events, in *untimed models* or perhaps regularly sampled models with fixed sampling period. Note however that such models are sufficient for some purposes, e.g. reachability analysis. Moreover it turns out that certain types of hybrid transition systems with state variables evolving both in continuous-time and via instantaneous discrete updates (events) are for various purposes equivalent to finite transition systems, a topic that we revisit later when we discuss simulation relations between automata. In this case, the original hybrid system formalism is used for modeling purposes, and assuming that certain restrictions are enforced on the type of admissible models, the model can sometimes be automatically transformed into a (usually very large) finite state system, adapted to computer operations.

A well-known class of such hybrid models is the class of *timed automata* introduced by Alur and Dill [AD94]. Other models of timed automata exist, see e.g. the references in [CL08, chapter 5]. Timed automata are finite transition systems with additional state variables called *clocks*, which are used to control the timing of the transitions. With such models, questions such as transition before a certain deadline, the number of events in a given time interval or the time spent in a particular state can be considered. Clearly, such questions are of critical importance in real-time systems such as control systems. Note that for synchronous discrete-time systems, standard discrete transition systems are sufficient. The advantage of timed automata is the possibility to model continuous-time systems in an efficient way, i.e. without employing say a very fine time discretization that would lead to a very large transition system. These models are also particularly useful for asynchronous systems.

Clock variables in timed automata are simply state variables verifying the differential equation $\dot{x} = 1$, with the additional possibility of *resetting* these variables during transitions between discrete states. These discrete states are also called *modes*. *Hybrid automata* greatly generalize timed automata by allowing general differential and algebraic equations in each mode, together with instantaneous mode switches as in timed automata. Hybrid automata are extremely expressive models but as a result many associated computational issues arise, such as the undecidability of certain reachability questions, possibility of generating non-physical behaviors such as Zeno behaviors, etc. Hence hybrid

automata should be used with care, because it is not always clear what is gained by describing a system as a hybrid automaton if computational issues are not discussed. In other words, trivializing the modeling problem generally comes at a high computational price. Still, hybrid automata form a popular class of hybrid system models and we give a formal definition below. The various elements of the model are often further constrained to try to improve the computational properties. Refer to the discussion in class for more details.

Definition 2.2.4. A hybrid automaton is a tuple

$$(Q, X, E, U, Y, f, \phi, \text{Inv}, \text{Guard}, \rho, q_0, X_0),$$

where

- Q is a set of discrete states, also called modes or locations
- X is a continous state space, e.g. $\mathbb{R}^n$
- E is a finite set of input events
- U is a set of admissible continuous controls
- f is a vector field for each mode: $f : Q \times X \times U \to X$
- ϕ is a discrete state transition function $\phi : Q \times X \times E \to Q$
- Inv is a set defining an invariant condition for each mode, $\text{Inv} \subseteq Q \times X$
- Guard is a set defining a guard condition for each transition $\text{Guard} \subseteq Q \times Q \times X$
- ρ is a reset function for each transition and input event: $\rho : Q \times Q \times X \times E \to X$
- $L : Q \times X \to 2^Y$ is an output or labeling function.
- q_0 is an initial discrete state
- x_0 is an initial continuous state.

Guards are condition under which a transition is allowed. Invariants are conditions that must be satisfied by the continuous state is a particular mode. Violation of this condition forces the state to take one of the transitions allowed by the guard conditions. Of course, certain relationships must be enforced between guards and invariants in order to have a well-posed model. Finally, we can have additional sources of non-determinism in the hybrid automaton model, by say defining discrete transitions as $\phi : Q \times X \times E \to 2^Q$, reset functions as $\rho : Q \times Q \times X \times E \to 2^X$, and allowing sets of initial conditions. Note that it is straightforward to *unfold* a hybrid automaton and view it as a transition system as in definition 2.2.2, at least as long as one allows uncountably many states and transitions.

For timed automata, in addition to the restrictions on the vector fields mentioned above, there are restrictions on the form of the guard and invariant conditions. Namely, these conditions must be of the form

$$\text{cond} ::= x < c \;\Big|\; x \leq c \;\Big|\; x > c \;\Big|\; x \geq c \;\Big|\; \text{cond} \wedge \text{cond}.$$

Moreover, the reset functions are only allowed to set clock variables back to 0.

Computational Tools

Two popular tools for creating and model-checking timed automata are UPPAAL [UPP] and KRONOS [Yov97]. Various tools for the manipulation of hybrid systems can be found on the hybrid system tools repository [Hyb].

2.3 Composing Systems

Composing Input-Output Systems

Composing input-output systems is straightforward, and the methods available in this framework tend to be compositional as well, in the sense that

- the complexity of the system grows modestly when we compose subsystems. For example, composing two systems with n inputs and n outputs in series produces a new system with n inputs and outputs. The complexity of characterizing the relationships between the new global inputs and outputs often grows reasonably with the number of subsystems (e.g. linearly). Consider for example the fact in the previous series connection, if the two subsystems have gain bounded by γ_1 and γ_2 then we immediately know that the connected system has gain bounded by $\gamma_1\gamma_2$.

- input-output analysis methods tend to focus on properties that are preserved under composition, e.g. passivity.

Note however that input-output connections typically rely on an idealization which consists in assuming that the behavior of a component is not modified when its output signals are used as inputs of another component. In practice, a component's behavior does depend on the load it is driving. However, engineers typically try to limit this phenomenon as much as possible, precisely to allow modular designs. For example, electronic systems have high input impedance and low output impedance for this purpose, and buffer circuits are available.

Review series, parallel and feedback connections, including the corresponding operations on the transfer functions for LTI systems. We will discuss in the next chapter how to connect continuous-time and discrete-time systems.

Composing State-Space and Transition Systems

The behavior of state-space models with respect to composition is much worse than for input-output models, with state space sizes that tend to grow exponentially with the number of subsystems. Note that for continuous state-space systems, the dimensions of the subsystems add. But discretizing the state space $X \subset \mathbb{R}^d$ of a continuous system for computational purposes leads to a number of state that grows exponentially with the dimension d.

We consider the following standard notion of composition of transition systems [BK08], called composition by handshaking, by synchronous message passing or simply parallel composition. Note that other types of compositions can be considered and useful.

Definition 2.3.1. Let $TS_i = (X_i, X_{0,i}, U_i, \longrightarrow_i, Y_i, L_i), i = 1, 2$ be two transition systems and $H \subset U_1 \cap U_2$. The transition system $TS_1||_H TS_2$ is defined as

$$TS_1||_H TS_2 = (X_1 \times X_2, U_1 \cup U_2, \rightarrow, X_{0,1} \times X_{0,2}, Y_1 \cup Y_2, L),$$

where $L((x_1, x_2)) = L_1(x_1) \cup L_2(x_2)$ and where the transition relation $\rightarrow$ is defined by the rules

- interleaving for $u \notin H$:

$$\frac{x_1 \xrightarrow{u} x_1'}{(x_1, x_2) \xrightarrow{u} (x_1', x_2)} \qquad \frac{x_2 \xrightarrow{u} x_2'}{(x_1, x_2) \xrightarrow{u} (x_1, x_2')}$$

- handshaking for $u \in H$:

$$\frac{x_1 \xrightarrow{u} x_1' \qquad x_2 \xrightarrow{u} x_2'}{(x_1, x_2) \xrightarrow{u} (x_1', x_2')}$$

In other words, in the system $TS_1||_H TS_2$, the transition systems synchronize on the inputs in H. We abbreviate the notation to $TS_1||TS_2$ if $H = U_1 \cap U_2$. The operation $||_H$ is obviously commutative, but also associative. Hence the composition of more than 2 transition systems $TS_1||_H TS_2||_H \cdots \|_H TS_n$ is defined immediately, for $H \subseteq U_1 \cap \cdots \cap U_n$. Note however that associativity does *not* hold if H is not the same in the different operations, i.e.

$$TS_1||_H (TS_2||_{H'} TS_3) \neq (TS_1||_H TS_2)||_{H'} TS_3$$

in general for $H \neq H'$.

Remark. Recall also that for automata the product construction has implications in terms of basic language operations, in particular for language intersection and union (the two differ only by the choice of marked or final states in the product automaton).