LECTURE 1

State-Space Linear Systems

This lecture introduces state-space linear systems, which are the main focus of this book.

Contents

1. State-Space Linear Systems
2. Block Diagrams
3. Exercises

1.1 State-Space Linear Systems

A *continuous-time state-space linear system* is defined by the following two equations:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t), \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^k, \tag{1.1a}$$
$$y(t) = C(t)x(t) + D(t)u(t), \qquad y \in \mathbb{R}^m. \tag{1.1b}$$

Notation. A function of time (either continuous $t \in [0, \infty)$ or discrete $t \in \mathbb{N}$) is called a *signal*.

The signals

$$u : [0, \infty) \to \mathbb{R}^k, \qquad x : [0, \infty) \to \mathbb{R}^n, \qquad y : [0, \infty) \to \mathbb{R}^m,$$

are called the *input*, *state*, and *output* of the system. The first-order differential equation (1.1a) is called the *state equation* and (1.1b) is called the *output equation*.

The equations (1.1) express an *input-output* relationship between theinput signal $u(\cdot)$ and the output signal $y(\cdot)$. For a given input $u(\cdot)$, we need to solve the state equation to determine the state $x(\cdot)$ and then replace it in the output equation to obtain the output $y(\cdot)$.

Notation 1. We write $u \overset{P}{\rightsquigarrow} y$ to mean that "y is one of the outputs that corresponds to u," the (optional) label P specifies the system under consideration.

Attention! For the same input $u(\cdot)$, different choices of the initial condition $x(0)$ on the state equation will result in different state trajectories $x(\cdot)$. Consequently, *one* input $u(\cdot)$ generally corresponds to *several* possible outputs $y(\cdot)$. □

1.1.1 Terminology and Notation

When the input signal u takes scalar values ($k = 1$) the system is called *single-input (SI)*, otherwise it is called *multiple-input (MI)*. When the output signal y takes scalar values ($m = 1$) the system is called *single-output (SO)*, otherwise it is called *multiple-output (MO)*.

When there is no state equation ($n = 0$) and we have simply

$$y(t) = D(t)u(t), \quad u \in \mathbb{R}^k, \quad y \in \mathbb{R}^m,$$

the system is called *memoryless*.

Note. The rationale behind this terminology is explained in Lecture 3.

When all the matrices $A(t)$, $B(t)$, $C(t)$, $D(t)$ are constant $\forall t \geq 0$, the system (1.1) is called a *Linear Time-Invariant (LTI)* system. In the general case, (1.1) is called a *Linear Time-Varying (LTV)* system to emphasize that time invariance is not being assumed. For example, Lecture 3 discusses *impulse responses of LTV systems* and *transfer functions of LTI systems*. This terminology indicates that the *impulse response* concept applies to both LTV and LTI systems, but the *transfer function* concept is meaningful only for LTI systems.

To keep formulas short, in the following we abbreviate (1.1) to

$$\dot{x} = A(t)x + B(t)u, \qquad y = C(t)x + D(t)u, \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^k,\ y \in \mathbb{R}^m \tag{CLTV}$$

MATLAB® Hit 1. `ss(A,B,C,D)` creates the continuous-time LTI state-space system (CLTI). ► p. 7

and in the time-invariant case, we further shorten this to

$$\dot{x} = Ax + Bu, \qquad y = Cx + Du, \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^k,\ y \in \mathbb{R}^m. \tag{CLTI}$$

Since these equations appear in the text numerous times, we use the special tags (CLTV) and (CLTI) to identify them.

1.1.2 Discrete-Time Case

A *discrete-time state-space linear system* is defined by the following two equations:

$$x(t+1) = A(t)x(t) + B(t)u(t), \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^k, \tag{1.2a}$$

$$y(t) = C(t)x(t) + D(t)u(t), \qquad y \in \mathbb{R}^m. \tag{1.2b}$$

Attention. *One* input generally corresponds to *several* outputs, because one may consider several initial conditions for the state equation.

All the terminology introduced for continuous-time systems also applies to discrete time, except that now the domain of the signals is $\mathbb{N} \coloneqq \{0, 1, 2, \dots\}$, instead of the interval $[0, \infty)$.

In discrete-time systems the state equation is a difference equation, instead of a first-order differential equation. However, the relationship between input and output is analogous. For a given input $u(\cdot)$, we need to solve the state (difference)

equation to determine the state $x(\cdot)$ and then replace it in the output equation to obtain the output $y(\cdot)$.

Note. Since this equation appears in the text numerous times, we use the special tag (DLTI) to identify it. The tag (DLTV) is used to identify the time-varying case in (1.2).

To keep formulas short to, in the sequel we abbreviate the time-invariant case of (1.2)

$$x^+ = Ax + Bu, \qquad y = Cx + Du, \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^k,\ y \in \mathbb{R}^m. \tag{DLTI}$$

1.1.3 State-Space Systems in MATLAB®

MATLAB® has several commands to create and manipulate LTI systems. The following basic command is used to create an LTI system.

Note. Initial conditions to LTI state-space MATLAB® systems are specified at simulation time.

MATLAB® Hint 1 (ss). The command `sys_ss=ss(A,B,C,D)` assigns to `sys_ss` a continuous-time LTI state-space MATLAB® system of the form

$$\dot{x} = \mathtt{A}x + \mathtt{B}u, \qquad y = \mathtt{C}x + \mathtt{D}u.$$

Optionally, one can specify the names of the inputs, outputs, and state to be used in subsequent plots as follows:

```
sys_ss=ss(A,B,C,D,...
          'InputName', {'input1', 'input2',...},...
          'OutputName',{'output1','output2',...},...
          'StateName', {'state1', 'state2',...});
```

The number of elements in the bracketed lists must match the number of inputs, outputs, and state variables.

For discrete-time systems, one should instead use the command `sys_ss=ss(A,B,C,D,Ts)`, where `Ts` is the sampling time, or `-1` if one does not want to specify it. □

1.2 Block Diagrams

Note. It is common practice to denote the input and output signals of a system by u and y, respectively. However, when dealing with interconnections, one must use different symbols for each signal, so this convention is abandoned.

It is convenient to represent systems by *block diagrams* as in Figure 1.1. These diagrams generally serve as compact representations for complex equations.

The two-port block in Figure 1.1(a) represents a system with input $u(\cdot)$ and output $y(\cdot)$, where the directions of the arrows specify which is which.

Although not explicitly represented in the diagram, one must keep in mind the existence of the state, which affects the output through the initial condition.

1.2.1 Interconnections

Interconnections of block diagrams are especially useful to highlight special structures in state-space equations. To understand what is meant by this, assume that the

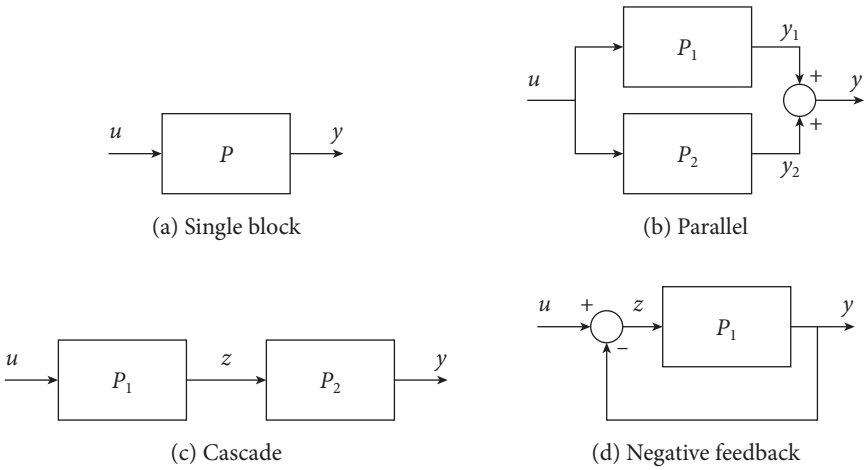


FIGURE 1.1. Block diagrams.

blocks P_1 and P_2 that appear in Figure 1.1 are the two LTI systems

$$P_1 : \dot{x}_1 = A_1x_1 + B_1u_1, \quad y_1 = C_1x_1 + D_1u_1, \quad x \in \mathbb{R}^{n_1},\ u \in \mathbb{R}^{k_1},\ y_1 \in \mathbb{R}^{m_1},$$
$$P_2 : \dot{x}_2 = A_2x_2 + B_2u_2, \quad y_2 = C_2x_2 + D_2u_2, \quad x \in \mathbb{R}^{n_2},\ u \in \mathbb{R}^{k_2},\ y_2 \in \mathbb{R}^{m_2}.$$

The general procedure to obtain the state-space for an interconnection consists of stacking the states of the individual subsystems in a tall vector x and computing $\dot{x}$ using the state and output equations of the individual blocks. The output equation is also obtained from the output equations of the subsystems.

In Figure 1.1(b) we have $u = u_1 = u_2$ and $y = y_1 + y_2$, which corresponds to a *parallel interconnection*. This figure represents the LTI system

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}A_1 & 0\\ 0 & A_2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}B_1\\ B_2\end{bmatrix}u, \qquad y = \begin{bmatrix}C_1 & C_2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + (D_1 + D_2)u,$$

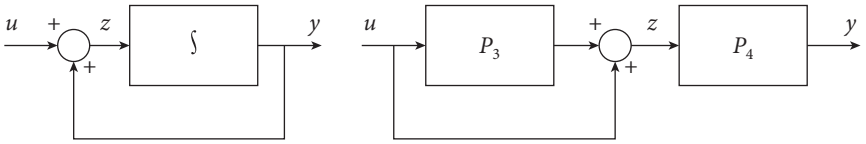
Notation. Given a vector (or matrix) x, we denote its transpose by x'.

with state $x \coloneqq [x_1'\ x_2']' \in \mathbb{R}^{n_1+n_2}$. The parallel interconnection is responsible for the block-diagonal structure in the matrix $\left[\begin{smallmatrix}A_1 & 0\\ 0 & A_2\end{smallmatrix}\right]$. A block-diagonal structure in this matrix indicates that the state-space system can be decomposed as the parallel of two state-space systems with smaller states.

In Figure 1.1(c) we have $u = u_1$, $y = y_2$, and $z = y_1 = u_2$, which corresponds to a *cascade interconnection*. This figure represents the LTI system

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}A_1 & 0\\ B_2C_1 & A_2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}B_1\\ B_2D_1\end{bmatrix}u, \quad y = \begin{bmatrix}D_2C_1 & C_2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + D_2D_1u,$$

with state $x \coloneqq [x_1'\ x_2']' \in \mathbb{R}^{n_1+n_2}$. The cascade interconnection is responsible for the block-triangular structure in the matrix $\left[\begin{smallmatrix}A_1 & 0\\ B_2C_1 & A_2\end{smallmatrix}\right]$ and, in fact, a block-triangular structure in this matrix indicates that the state-space system can be decomposed as a cascade of two state-space systems with smaller states.



(a) $\dot{x} = x + u$, $y = x$ (b) LTI system in (1.4)

FIGURE 1.2. Block diagram representation systems.

In Figure 1.1(d) we have $u_1 = u - y_1$ and $y = y_1$, which corresponds to a *negative-feedback interconnection*. This figure represents the following LTI system

$$\dot{x}_1 = (A_1 - B_1(I + D_1)^{-1}C_1)x_1 + B_1(I - (I + D_1)^{-1}D_1)u, \tag{1.3a}$$

$$y = (I + D_1)^{-1}C_1x_1 + (I + D_1)^{-1}D_1u, \tag{1.3b}$$

with state $x_1 \in \mathbb{R}^{n_1}$. Sometimes feedback interconnections are ill-posed. In this example, this would happen if the matrix $I + D_1$ were singular.

Note. How to arrive at equation (1.3)? *Hint: Start with the output equation, note that*

$y = C_1x_1 + D_1(u - y)$,

and solve for y.

MATLAB® Hint 2. To avoid ill-posed feedback interconnections, MATLAB® warns about algebraic loops when one attempts to close feedback loops around systems like P_1 with nonzero D_1 matrices (even when $I + D_1$ is invertible).

The basic interconnections in Figure 1.1 can be combined to form arbitrarily complex diagrams. The general procedure to obtain the final state-space system remains the same: Stack the states of all subsystems in a tall vector x and compute $\dot{x}$ using the state and output equations of the individual blocks.

1.2.2 System Decomposition

Block diagrams are also useful to represent complex systems as the interconnection of simple blocks. This can be seen through the following two examples:

1. The LTI system

$$\dot{x} = x + u, \qquad y = x$$

can be viewed as a feedback connection in Figure 1.2(a), where the *integrator* system $\int$ maps each input z to the solutions y of

$$\dot{y} = z.$$

MATLAB® Hint 3. This type of decomposition is especially useful to build systems in Simulink®.

Note. The upper triangular form in (1.4) gives away that this could be decomposable as a cascade.

2. Consider the LTI system

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}1 & 1\\ 0 & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}1\\ 5\end{bmatrix}u, \qquad y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}. \tag{1.4}$$

Writing these equations as

$$\dot{x}_2 = 3x_2 + 5u, \qquad y_2 = x_2 \tag{1.5}$$

Note. In general, this type of decomposition is not unique, since there may be many ways to represent a system as the interconnection of simpler systems.

and

$$\dot{x}_1 = x_1 + z, \qquad y = x_1, \tag{1.6}$$

where $z := x_2 + u$, we conclude that (1.4) can be viewed as the block diagram in Figure 1.2(b), where P_3 corresponds to the LTI system (1.5) with input u and output y_2 and P_4 corresponds to the LTI systems (1.6) with input z and output y.

1.2.3 System Interconnections with MATLAB®

Attention! Note the different order in which `sys1` and `sys2` appear in the two forms of this command.

MATLAB® Hint 4 (`series`) The command `sys=series(sys1,sys2)` or, alternatively, `sys=sys2*sys1` creates a system `sys` from the cascade connection of the system `sys1` whose output is connected to the input of `sys2`.

For MIMO systems, one can use `sys=series(sys1,sys2,outputs1, inputs2)`, where `outputs1` and `inputs2` are vectors that specify the outputs of `sys1` and inputs of `sys2`, respectively, that should be connected. These two vectors should have the same size and contain integer indexes starting at 1. □

MATLAB® Hint 5 (`parallel`) The command `sys=parallel(sys1,sys2)` or, alternatively, `sys=sys1+sys2` creates a system `sys` from the parallel connection of the systems `sys1` and `sys2`.

For MIMO systems, one can use `sys=parallel(sys1,sys2,inputs1, inputs2,outputs1,outputs2)`, where `inputs1` and `inputs2` specify which inputs should be connected and `outputs1` and `outputs2` specify which outputs should be added. All four vectors should contain integer indexes starting at 1. □

MATLAB® Hint 6 (`append`) The command `sys=append(sys1,sys2, ...,sysn)` creates a system `sys` whose inputs are the union of the inputs of all the systems `sys1`, `sys2`, ... , `sysn` and whose outputs are the union of the outputs of all the same systems. The dynamics are maintained decoupled. □

MATLAB® Hint 7 (`feedback`) The command `sys=feedback(sys1,sys2)` creates a system `sys` from the negative feedback interconnection of the system `sys1` in the forward loop, with the system `sys2` in the backward loop. A positive feedback interconnection can be obtained using `sys=feedback(sys1,sys2,+1)`.

For MIMO systems, one can use `sys=feedback(sys1,sys2,feedinputs, feedoutputs,sign)`, where `feedinputs` specify which inputs of the forward-loop system `sys1` receive feedback from `sys2`, `feedoutputs` specify which outputs of the forward-loop system `sys1` are feedback to `sys2`, and `sign` $\in \{-1, +1\}$ specifies whether a negative or positive feedback configuration should be used. More details can be obtained by using `help feedback`. □

1.3 Exercises

1.1 (Block diagram decomposition). Consider a system P_1 that maps each input u to the solutions y of

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 4 \\ 1 \end{bmatrix} u, \qquad y = \begin{bmatrix} 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Represent this system in terms of a block diagram consisting only of

- *integrator* systems, represented by the symbol $\boxed{\int}$, that map their input $u(\cdot) \in \mathbb{R}$ to the solution $y(\cdot) \in \mathbb{R}$ of $\dot{y} = u$;
- *summation* blocks, represented by the symbol $\boxed{\sum}$, that map their input vector $u(\cdot) \in \mathbb{R}^k$ to the scalar output $y(t) = \sum_{i=1}^{k} u_i(t)$, $\forall t \geq 0$; and
- *gain* memoryless systems, represented by the symbol $\boxed{g}$, that map their input $u(\cdot) \in \mathbb{R}$ to the output $y(t) = g\, u(t) \in \mathbb{R}$, $\forall t \geq 0$ for some $g \in \mathbb{R}$. □