8 State Space LTI Models

This lecture gives an introduction to linear time invariant (LTI) state space models of finite order.

8.1 DT LTI State Space Models

Formally, a fnite order LTI state space model is defined by specifying a time domain (either *discrete time* (DT) or *continuous time* (CT)), and four real matrices a, b, c, d of dimensions n-by-n, n-by-m, k-by-n, and k-by-m respectively. In other words, a must be a square matrix, and it should be possible to combine a, b, c, d into a 2-by-2 block matrix

$$\mathcal{M} = \mathcal{M}[a, b, c, d] = \left[\begin{array}{cc} a & b \\ c & d \end{array} \right].$$

Definition 8.1 Let a, b, c, d be real matrices of dimensions n-by-n, n-by-m, k-by-n, and k-by-m respectively. The *discrete time LTI state space model* specified by a, b, c, d is the function

$$\mathcal{S}_{DT}[a, b, c, d] : \ \mathbb{R}^n \times \ell_m \mapsto \ell_k$$

mapping $(x_0, f(\cdot)) \in \mathbb{R}^n \times \ell_m$ into $y \in \ell_k$ according to the set of *difference equations*

$$x(t+1) = ax(t) + bf(t), \ x(0) = x_0, \ y(t) = cx(t) + df(t). \tag{8.1}$$

It is understood that $\mathcal{S}_{DT}[a, b, c, d]$ defines an input - output system as the set of all pairs $(f, y) \in \ell_m \times \ell_k$ such that

$$y = \mathcal{S}_{DT}[a, b, c, d](x_0, f)$$

for some $x_0 \in \mathbb{R}^n$. The case $n = 0$, when a, b, c are *empty matrices*, is also acceptable, defining the *memoryless* relation $y(t) = df(t)$. Using the empty set notation $\emptyset$ for empty (0-by-q or q-by-0) matrices, the memoryless system is denoted by $\mathcal{S}_{DT}[\emptyset, \emptyset, \emptyset, d]$.

According to the definition, a discrete time state space model is a transformation mapping the vector $x_0 \in \mathbb{R}^n$ of *initial conditions* and an *input signal* $f \in \ell_m$ into a uniquely defined *output signal* $y \in \ell_k$. The signal $x \in \ell_n$ involved in the intermediate calculations is called the *state vector* of the system.

Example 8.1 The transformation mapping two real numbers y_0, y_1 and a sequence $f \in \ell_1$ to the sequence $y \in \ell_1$ defined by

$$y(t+2) = y(t+1) + y(t) + f(t) \quad (t \in \mathbb{Z}_+), \quad y(0) = y_0, \ y(1) = y_1,$$

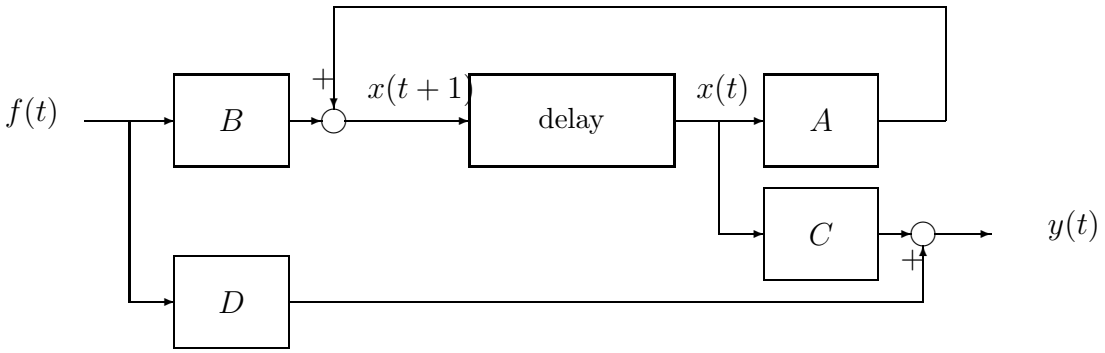

Figure 8.1: Discrete time state space model

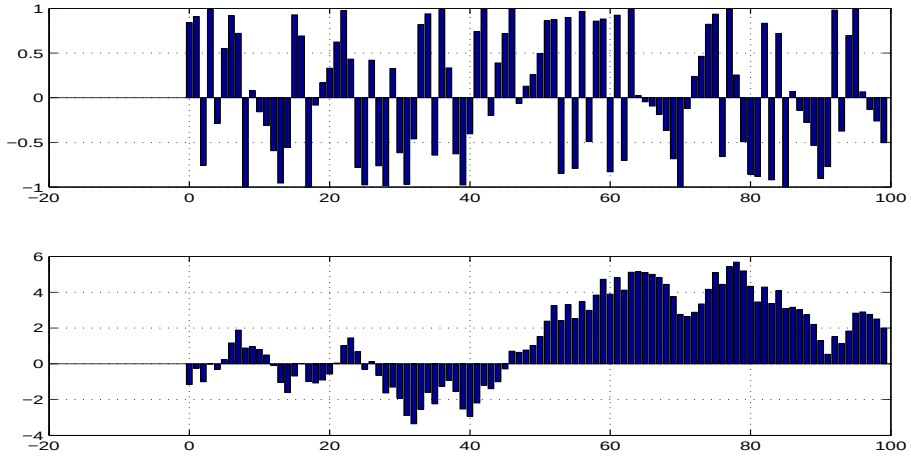

Figure 8.2: System input and output from Example 8.2

can be interpreted as a state space model (8.1) with

$$x(t) = \begin{bmatrix} y(t+1) \\ y(t) \end{bmatrix}, \quad a = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad c = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad d = 0, \quad x_0 = \begin{bmatrix} y_1 \\ y_0 \end{bmatrix}.$$

For numerical calculations with MATLAB, state space models are specified using the function `ss.m`, and the result of applying DT system transformations to signals are computed using `lsim.m`.

Example 8.2 The summation relation

$$y(t) = x_0 + \sum_{\tau=0}^{t} f(\tau), \quad t = 0, 1, 2, \ldots,$$

can be interpreted as a DT LTI state space model with input f, output y,

$$x(t) = y(t-1), \quad a = 1, \quad b = 1, \quad c = 1, \quad d = 1.$$

To calculate the first 100 values of system response to $f(t) = \sin(2^t)$ with $x_0 = -2$, one can use the script

```
a=1;b=1;c=1;d=1;                    % state space matrices
S=ss(a,b,c,d,-1);                   % DT state space model
N=100;t=0:N-1;v=sin(2.^t);          % values of v
x0=-2;                              % initial state
y=lsim(S,v,[],x0);                  % simulate
close(gcf);                         % close current plot
subplot(2,1,1);bar(t,v);grid        % subplot of input
subplot(2,1,2);bar(t,y);grid        % subplot of output
```

which produces the plot shown on Figure 8.2.

8.2 CT LTI State Space Models

The continuous time definition is similar to its discrete time alternative.

Definition 8.2 Let a, b, c, d be real matrices of dimensions n-by-n, n-by-m, k-by-n, and k-by-m respectively. The *continuous time LTI state space model* specified by a, b, c, d is the function

$$\mathcal{S}_{CT}[a, b, c, d] : \ \mathbb{R}^n \times \mathcal{L}_m \mapsto \mathcal{L}_k$$

which maps $(x_0, f(\cdot)) \in \mathbb{R}^n \times \mathcal{L}_m$ into $y(\cdot) \in \mathcal{L}_k$ defined according to

$$\dot{x}(t) = ax(t) + bf(t), \ x(0) = x_0, \ y(t) = cx(t) + df(t), \tag{8.2}$$

where the *differential equation* in (8.2) is interpreted as

$$x(t) = x_0 + \int_0^t (ax(\tau) + bf(\tau))d\tau.$$

The case $n = 0$, when a, b, c are *empty matrices*, is also acceptable, defining the *memoryless* relation $y(t) = df(t)$, denoted by $\mathcal{S}_{CT}[\emptyset, \emptyset, \emptyset, d]$.

Since the right side of the differential equation in (8.2) is continuous, a continuous time state space model defines a transformation mapping the vector $x_0 \in \mathbb{R}^n$ of *initial conditions* and an *input signal* $f \in \mathcal{L}_m$ into a uniquely defined *output signal* $y \in \mathcal{L}_k$. This results in an input - output model consisting of all pairs (f, y) such that $y = \mathcal{S}_{CT}[a, b, c, d](x_0, f)$ for some $x_0 \in \mathbb{R}^n$. The signal $x \in \mathcal{L}_n$ involved in the intermediate calculations is called the *state vector* of the system.

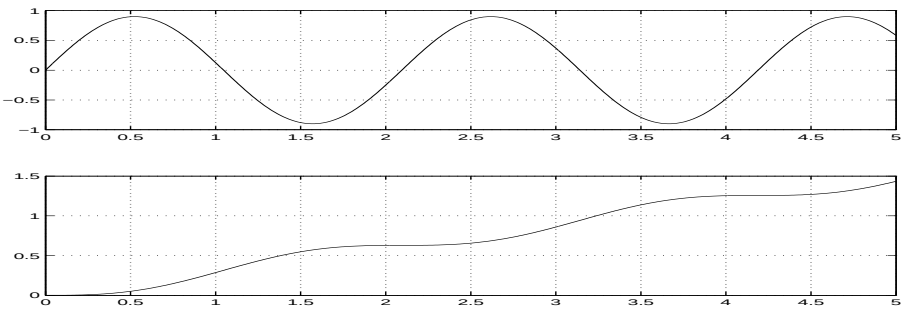

Figure 8.3: System input and output from Example 8.4

Example 8.3 The Newton's second law relation $m\ddot{y} = f$ between mass m, coordinates y, and force f can be written in the state space form (8.2) with

$$x(t) = \begin{bmatrix} \dot{y}(t) \\ y(t) \end{bmatrix}, \quad a = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad c = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad d = 0.$$

For numerical calculations with MATLAB, state space models are specified using the function `ss.m`, and the result of applying CT system transformations to signals are computed using `lsim.m`.

Example 8.4 Differential equation

$$\ddot{y}(t) = f(t), \quad t \in \mathbb{R}_+$$

relating position $y(t)$ to acceleration $f(t)$ can be interpreted as a CT LTI state space model with

$$a = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \; b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \; c = \begin{bmatrix} 1 & 0 \end{bmatrix}, \; d = 0, \; x(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}.$$

To calculate 500 samples of system response to harmonic input $f(t) = 0.9\sin(3t)$ with zero initial conditions on time interval $[0, 5]$, one can use the script

```
a=[0 1;0 0];b=[0;1];c=[1 0];d=0;            % coefficient matrices
S=ss(a,b,c,d);                              % CT state space model
N=500;T=5;t=linspace(0,T,N);v=0.9*sin(3*t); % input signal
x0=[0;0];                                   % initial condition
y=lsim(S,v,t,x0);                           % calculate response
close(gcf)
subplot(2,1,1);plot(t,v);grid               % subplot of input
subplot(2,1,2);plot(t,y);grid               % subplot of output
```

which produces the plot shown on Figure 8.3.

When the nature of the time variable t (discrete or continuous) is not important, the notation $\mathcal{S}[a, b, c, d]$ will be used.

8.3 Transfer Matrices Associated With State Space Models

Transfer matrices offer important insight into functioning of LTI models, and provide useful shortcut in LTI system notation.

For a system with m-dimensional input and k-dimensional output defined by an LTI state space model, transfer function $G = G(\lambda)$ is defined for all $\lambda \in \mathbb{C}$ (except, possibly, a finite number) as a k-by-m matrix $G(\lambda)$ such that the exponential input - output pair (f, y) with

$$f(t) = z^t, \quad y(t) = G(z)z^t$$

is admissible in the DT case, and

$$f(t) = e^{st}, \quad y(t) = G(s)e^{st}$$

is admissible in the CT case.

Definition 8.3 Let a, b, c, d be real matrices of dimensions n-by-n, n-by-m, k-by-n, and k-by-m respectively. The *transfer matrix* $\mathcal{G}[a, b, c, d]$ of the corresponding state space model $\mathcal{S}[a, b, c, d]$ is the maximal analytical extension of the k-by-m matrix-valued function of complex scalar argument λ defined by

$$\mathcal{G}[a, b, c, d](\lambda) = d + c(\lambda I - a)^{-1}b, \quad \mathcal{G}[a, b, c, d](\infty) \stackrel{\text{def}}{=} d. \tag{8.3}$$

A 1-by-1 transfer matrix is also called *transfer function*.

Originally, (8.3) defines $\mathcal{G}[a, b, c, d](\lambda)$ only for those $\lambda \in \mathbb{C}$ for which matrix $\lambda I - a$ is invertible. However, since two rational functions are considered *equal* when they are equal for all values of their argument except, possibly, a finite set, sometimes the domain of $\mathcal{G}[a, b, c, d]$ can be extended to include some of those $\lambda \in \mathbb{C}$ for which matrix $\lambda I - a$ is not invertible.

Example 8.5 $\mathcal{G}[2, 0, 1, 1](\lambda) = \mathcal{G}[1, 1, 0, 1](\lambda) = \mathcal{G}_{\emptyset,\emptyset,\emptyset,1}(\lambda) \equiv 1$.

While the definition of transfer matrix is the same for discrete and continuous time systems, a distinction is usually made by using letter s for the argument of transfer matrices of CT models, replacing it by z in the DT case.

Example 8.6 While

$$\mathcal{G}[1, 1, 1, 1](\lambda) = 1 + 1/(\lambda - 1) = \lambda/(\lambda - 1),$$

the transfer function of $\mathcal{S}_{CT}[1, 1, 1, 1]$ is written as $G(s) = s/(s - 1)$, and transfer function of $\mathcal{S}_{DT}[1, 1, 1, 1]$ is $G(z) = z/(z - 1)$.

As can already be seen from Example 8.5, it is possible for two different LTI state space models to define the same transfer matrix. While it is possible for two state space models with identical transfer matrices to define different input - output relations, those with *zero initial conditions* are always identical.

Theorem 8.1 *Let $\mathcal{S}[a_1, b_1, c_1, d_1]$ and $\mathcal{S}[a_2, b_2, c_2, d_2]$ be two state space models. The following conditions are equivalent:*

(a) $\mathcal{G}[a_1, b_1, c_1, d_1] = \mathcal{G}[a_2, b_2, c_2, d_2]$*;*

(b) $\mathcal{S}[a_1, b_1, c_1, d_1](0, f) = \mathcal{S}[a_2, b_2, c_2, d_2](0, f)$ *for all* f*.*

Example 8.7 Transfer functions $\mathcal{G}[0, 0, 1, 0]$ and $\mathcal{G}[0, 1, 0, 0]$ are the same (zero) transfer functions, but $\mathcal{S}_{CT}[0, 0, 1, 0](x_0, f) = x_0$ is not the same as $\mathcal{S}_{CT}[0, 1, 0, 0](x_0, f) = 0$ for $x_0 \neq 0$, except when $x_0 = 0$.

8.4 Equivalence and Minimality of State Space Models

The correspondence between transfer matrices and state space models is better understood utilizing the notions of *minimality* and *equivalence*.

Definition 8.4 State space model $\mathcal{S}[a_1, b_1, c_1, d_1]$ is said to be *similar* to state space model $\mathcal{S}[a_2, b_2, c_2, d_2]$ when there exists a non-singular real matrix S of same dimensions as a_1 such that

$$a_2 = S^{-1} a_1 S, \quad b_2 = S^{-1} b_1, \quad c_2 = c_1 S, \quad d_2 = d_1. \tag{8.4}$$

Transfer matrices of similar state space models are equal, and the corresponding signal transformations are the same exept for using different coordinate systems for the state vector.

Lemma 8.1 *Let $\mathcal{S}[a_1, b_1, c_1, d_1]$ and $\mathcal{S}[a_2, b_2, c_2, d_2]$ be two similar state space models. Then*

(a) $\mathcal{G}[a_1, b_1, c_1, d_1] = \mathcal{G}[a_2, b_2, c_2, d_2]$*;*

(b) $\mathcal{S}[a_2, b_2, c_2, d_2]$ *maps* $(x_0, f(\cdot))$ *into* $y(\cdot)$ *if and only if* $\mathcal{S}[a_1, b_1, c_1, d_1]$ *maps* $(Sx_0, f(\cdot))$ *into* $y(\cdot)$*.*

A proof of Lemma 8.1 is given in the Appendix of this section. Note that, according to Example 8.5, state space models with equal transfer matrices are not necessarily similar.

Definition 8.5 State space model $\mathcal{S}[a,b,c,d]$ is called *minimal* when the pair (a,b) is controllable and the pair (c,a) is observable, i.e. when matrices

$$M_c[a,b] = \begin{bmatrix} b & ab & \dots & a^{n-1}b \end{bmatrix}, \quad M_o[c,a] = \begin{bmatrix} c \\ ca \\ \vdots \\ ca^{n-1} \end{bmatrix} \tag{8.5}$$

are respectively right and left invertible.

Of all state space models with a given transfer matrix, the minimal ones have the smallest number of states.

Lemma 8.2 *Let $G = G(\lambda)$ be a rational matrix-valued function, which is* proper, *in the sense that the entries of $G(\lambda)$ are bounded as $|\lambda| \to \infty$. Then*

(a) there exists a minimal LTI state space model with transfer matrix G;

(b) all minimal state space models with the same transfer matrix G are similar;

(c) number of states in every non-minimal state space model with transfer matrix G is strictly larger than the number of states in a minimal state space model with transfer matrix G.

Lemma 8.2 allows one to define *order* of a rational matrix-valued function.

Definition 8.6 The *order* of a rational matrix-valued function G is the number of states in a minimal state space model with transfer matrix G.

According to the standard convention, to be used in this book as well, a *minimal* state space model can be specified by its transfer matrix, as long as the system of coordinates used for the state vector can be ignored.

A convenient way of defining a transfer matrix in MATLAB is via the symbolic variable `s` (or `z` for the discrete time case), using commands `tf('s')` or `tf('z')` respectively.

Example 8.8 MATLAB code

```
s=tf('s');
G=[(s^3+s)/(s^4+1) 1/s^2];
```

defines

$$G(s) = \begin{bmatrix} \frac{s^3+s}{s^4+1} & \frac{1}{s^2} \end{bmatrix}.$$

Conversion from transfer matrices to state space models and back is achieved using functions `ss.m`, `tf.m`, `ssdata.m`, and `tfdata.m`. Due to the imprecise nature of systems calculations with MATLAB, `tf.m` can introduce weird numerical errors, and `ss.m` will not necessarily produce a minimal state space model. To force minimality, use `minreal.m`.

Example 8.9 MATLAB code

```
z=tf('z');
H=[1/z 1/z;1/z 1/z];
HSS=ss(H);
[A,B,C,D]=ssdata(HSS)
[a,b,c,d]=ssdata(minreal(HSS))
```

produces

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

which is not a minimal state space model, and, within the achievable accuracy,

$$a = 0, \quad b = \sqrt{2} \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad c = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Example 8.10 MATLAB expression `tf(ss(1,1,1,1))` produces

$$G(s) = \frac{s - 2.027 \cdot 10^{-17}}{s - 1}$$

instead of the expected $G(s) = s/(s-1)$.

As a rule, the use of state space models is preferable to transfer matrices in numerical calculations of limited accuracy.

Example 8.11 Calculating the 200-th power of $\mathcal{G}_{CT}[-1, 1, -2, 1]$, and then evaluating its absolute value at $s_0 = 100$ is expected to produce a number approximately equal to $e^{-2} \approx 0.14$. While state space manipulations work well in this case, using transfer functions results in a failure (a `NaN`, "not a number" outcome).

```
Hss=ss(-1,1,-1,1);                % state space models
s=tf('s'); Htf=s/(s+1);           % transfer function model
s0=100;                           % evaluation point
evalfr(Hss^200,s0)                % state space calculations
evalfr(Htf^200,s0)                % transfer function calculations
```