

Sliding Mode Control Using MATLAB

Sliding Mode Control Using MATLAB

Jinkun Liu

Beihang University, Beijing, China



ACADEMIC PRESS

An imprint of Elsevier

Sliding mode control MATLAB simulation basic theory and design method

In the formulation of any control problem, there will typically be discrepancies between the actual plant and the mathematical model developed for controller design. This mismatch may be due to unmodelled dynamics, variation in system parameters or the approximation of complex plant behavior by a straightforward model. The engineer must ensure that the resulting controller has the ability to produce the required performance levels in practice despite such plant /model mismatches. This has led to intense interest in the development of robust control methods that seek to solve this problem. One particular approach to robust controller design is the sliding mode control methodology.

One of the most intriguing aspects of sliding mode is the discontinuous nature of the control action, whose primary function of each of the feedback channels is to switch between two distinctively different system structures (or components) such that a new type of system motion, called the sliding mode, exists in a manifold. This peculiar system characteristic is claimed to result in superb system performance, which includes insensitivity to parameter variations, and complete rejection of disturbances.

Sliding mode control is a particular type of variable structure control. In sliding mode control, the control system is designed to drive and then constrain the system state to lie within a neighborhood of the switching function. There are two main advantages to this approach. Firstly, the dynamic behavior of the system may be tailored by the particular choice of switching function. Secondly, the closed-loop response becomes totally insensitive to a particular class of uncertainty. The latter invariance property clearly makes the methodology an appropriate candidate for robust control. In addition, the ability to specify performance directly makes sliding mode control attractive from a design perspective.

The sliding mode design approach consists of two components. The first involves the design of a switching function so that the sliding motion satisfies design specifications. The second is concerned with the selection of a control law that will make the switching function attractive to the system state. Note that this control law is not necessarily discontinuous.

The chattering phenomenon is generally perceived as motion that oscillates about the sliding manifold. There are two possible mechanisms that produce such a motion. Firstly, in the absence of switching nonidealities such as delays, i.e., the switching device is switching ideally at an infinite frequency, the presence of parasitic dynamics in series with the plant causes a small amplitude high-frequency oscillation to appear in the neighborhood of the

sliding manifold. These parasitic dynamics represent the fast actuator and sensor dynamics. Secondly, the switching nonidealities alone can cause such high-frequency oscillations.

In this book, we aim to accomplish these objectives:

- Provide reasonable methods of the chattering phenomenon alleviating.
- Offer a catalogue of implementable robust sliding mode control design solutions for engineering applications.
- Provide advanced sliding mode controller design methods and their stability analysis.
- For each sliding mode control algorithm, we offer its simulation example and Matlab program.

This book provides the reader with a thorough grounding in sliding mode controller design. From this basis, more advanced theoretical results are developed. Typical sliding mode controller design is emphasized using Matlab simulation. In this book, concrete case studies, which present the results of sliding mode controller implementations, are used to illustrate the successful practical application of the theory.

The book is structured as follows.

Chapter 1, Basic sliding mode control principle and design, introduces the concept of sliding mode control and illustrates the attendant features of robustness and performance specification using a straightforward example and graphical exposition. Several typical sliding mode controllers for continuous system are introduced, and concrete stability analysis, simulation examples and Matlab programs are given.

In Chapter 2, Sliding mode control with high performance, firstly an adaptive sliding mode control is introduced for mechanical systems with tanh function; to avoid a control input value that is too big, a projection algorithm is used. Secondly, the problem of tracking control with prescribed performance guarantees is considered, the error evolution within prescribed performance bounds in both problems of regulation and tracking.

In Chapter 3, Sliding mode control based on a state observer, several kinds of state observer such as high gain observer, K observer, high gain differentiator, robust observer and separation theorem are introduced, and based on the different observer, sliding mode controller is designed.

In Chapter 4, Sliding mode control based on disturbance and a delayed observer, an exponential disturbance observer, delayed output observer for linear system and delayed output observer for nonlinear system are introduced, and closed system stability and convergence are analyzed.

In Chapter 5, Sliding mode control based on LMI, several kinds of sliding mode controller based on LMI technology are introduced, closed system stability and convergence are analyzed, and simulation examples are given.

In Chapter 6, Sliding mode control based on the RBF neural network, firstly, a simple adaptive sliding mode control based on RBF is introduced, then an adaptive sliding mode control based on RBF compensation is discussed. Sliding mode control based on RBF neural network with minimum parameter learning method is introduced, and finally, a sliding mode controller based on RBF with MPL is introduced.

In Chapter 7, Sliding mode control based on a fuzzy system, firstly, sliding mode control based on the fuzzy system approximation is introduced. Then, based on the fuzzy system with minimum parameter learning method, a sliding mode controller based on fuzzy system is designed.

In Chapter 8, Sliding mode control of a class of underactuated systems, considering several kinds of underactuated system, sliding mode controllers are designed, and the Lyapunov function and the Hurwitz method are used to analyze closed system stability.

In Chapter 9, Sliding mode control for underactuated system with decoupling algorithm, a general decoupling algorithm for underactuated system is introduced. With this decoupling algorithm, sliding mode control is designed, and the Lyapunov function and the Hurwitz method are used to analyze closed system stability.

All the control algorithms are described separately and classified by chapter name; all the programs can be run successfully in MATLAB and can be downloaded via <http://shi.buaa.edu.cn/liujinkun>.

If you have questions about algorithms and simulation programs, please contact the author at ljk@buaa.edu.cn.

Basic sliding mode control principle and design

Sliding mode techniques are one approach to solving control problems and are an area of increasing interest.

This book provides the reader with classical sliding mode control design examples, based on [1] and [2].

Variable Structure Control (VSC) with Sliding Mode Control (SMC) was first proposed and elaborated in early 1950s in the Soviet Union by Emelyanov and several coresearchers such as Utkins and Itkis [3]. During the last decades significant interest on VSC and SMC have been generated in the control research community.

SMC has been applied including nonlinear system, multiinput multioutput (MIMO) systems, discrete-time models, large-scale and infinite-dimension systems, and stochastic systems. The most eminent feature of SMC is it is completely insensitive to parametric uncertainty and external disturbances during sliding mode [4].

Essentially, VSC utilizes a high-speed switching control law to drive the nonlinear plant's state trajectory onto a specified and user-chosen surface in the state space, which is called the sliding or switching surface, and to maintain the plant's state trajectory on this surface for all subsequent time. This surface is called the switching surface because if the state trajectory of the plant is “above” the surface, a control path has one gain and a different gain if the trajectory drops “below” the surface. During the process, the control system's structure varies from one to another, thus earning the name VSC. To emphasize the important role of the sliding mode, the control is also called SMC [5].

In SMC, the system is designed to drive and then constrain the system state to lie within a neighborhood of the switching function. Its two main advantages are (1) the dynamic behavior of the system may be tailored

2 CHAPTER 1 Basic sliding mode control principle and design

by the particular choice of switching function, and (2) the closed-loop response becomes totally insensitive to a particular class of uncertainty. Also, the ability to specify performance directly makes SMC attractive from the design perspective.

Trajectory of a system can be stabilized by a sliding mode controller. After the initial reaching phase, the system states “slides” along the line $s = 0$. The particular $s = 0$ surface is chosen because it has desirable reduced-order dynamics when constrained to it. In this case, the $s = cx_1 + \dot{x}_1, c > 0$. Surface corresponds to the first-order LTI system $\dot{x}_1 = -cx_1$, which has an exponentially stable origin.

There are two steps in the SMC design. The first step is designing a sliding surface so that the plant restricted to the sliding surface has a desired system response. This means the state variables of the plant dynamics are constrained to satisfy another set of equations which define the so-called switching surface. The second step is constructing a switched feedback gains necessary to drive the plant’s state trajectory to the sliding surface. These constructions are built on the generalized Lyapunov stability theory.

Now we give a simple sliding mode controller design example as follows.

1.1 A SIMPLE SLIDING MODE CONTROLLER DESIGN

Consider a plant as

$$J\ddot{\theta}(t) = u(t) + d(t), \quad (1.1)$$

where J is the inertia moment, $\ddot{\theta}(t)$ is the angle signal, $u(t)$ is the control input, $d(t)$ is the disturbance and $|d(t)| \leq \eta$.

Design the sliding mode function as

$$s(t) = ce(t) + \dot{e}(t) \quad (1.2)$$

where c must satisfy Hurwitz condition, $c > 0$.

The tracking error and its derivative value is

$$e(t) = \theta(t) - \theta_d(t), \quad \dot{e}(t) = \dot{\theta}(t) - \dot{\theta}_d(t)$$

where $\theta_d(t)$ is the ideal position signal.

From Eq. (1.2), we can see that if $s(t) = 0$, then $ce(t) + \dot{e}(t) = 0$, and we can get $e(t) = e(0)\exp(-ct)$. That is, when $t \rightarrow \infty$, position tracking

error and speed tracking error will tend to zero exponentially with c value.

From Eq. (1.2), we have

$$\dot{s}(t) = c\dot{e}(t) + \ddot{e}(t) = c\dot{e}(t) + \ddot{\theta}(t) - \ddot{\theta}_d(t) = c\dot{e}(t) + \frac{1}{J}(u + d(t)) - \ddot{\theta}_d(t) \quad (1.3)$$

and

$$s\dot{s} = s\left(c\dot{e} + \frac{1}{J}(u + d(t)) - \ddot{\theta}_d\right).$$

To guarantee $s\dot{s} < 0$, design the sliding mode controller as

$$u(t) = J\left(-c\dot{e} + \ddot{\theta}_d - \frac{1}{J}(ks + \eta \text{sgns})\right) \quad (1.4)$$

Define the Lyapunov function as

$$V = \frac{1}{2}s^2.$$

We obtain

$$\dot{V} = s\dot{s} = s\left(-ks - \frac{1}{J}\eta \text{sgns} + \frac{1}{J}d(t)\right) = \frac{1}{J}(-ks^2 - \eta|s| + sd(t)) \leq -\frac{1}{J}ks^2.$$

Then we have $\dot{V} \leq -\frac{2}{J}kV$. Use Lemma 1.3, choose $\alpha = \frac{2}{J}k$ and $f = 0$; we can obtain

$$V(t) \leq e^{-\frac{2}{J}kt}V(t_0)$$

If k is a positive constant value, $V(t)$ will tend to zero exponentially with k value.

In control law (1.4), the switch term ηsgns is a robust term, which is used to overcome $d(t)$. However, larger value of η may cause chattering of control input, how to eliminate the chattering is an important problem in SMC theory.

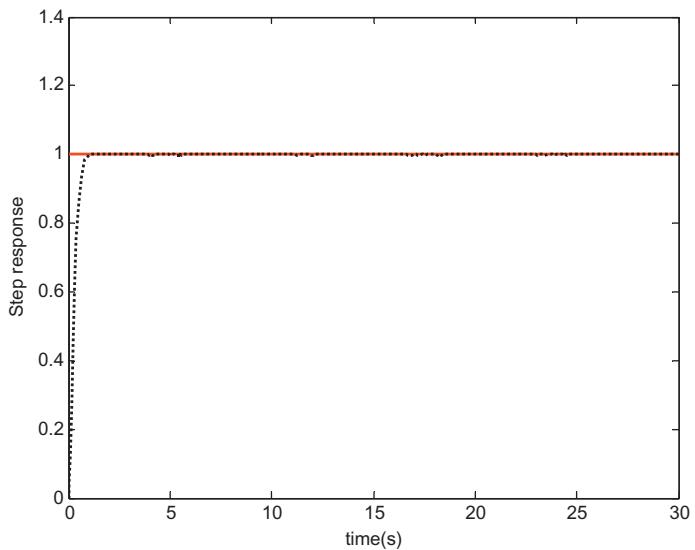
Now we give a simulation example to explain, consider the plant as

$$J\ddot{\theta}(t) = u(t) + d(t),$$

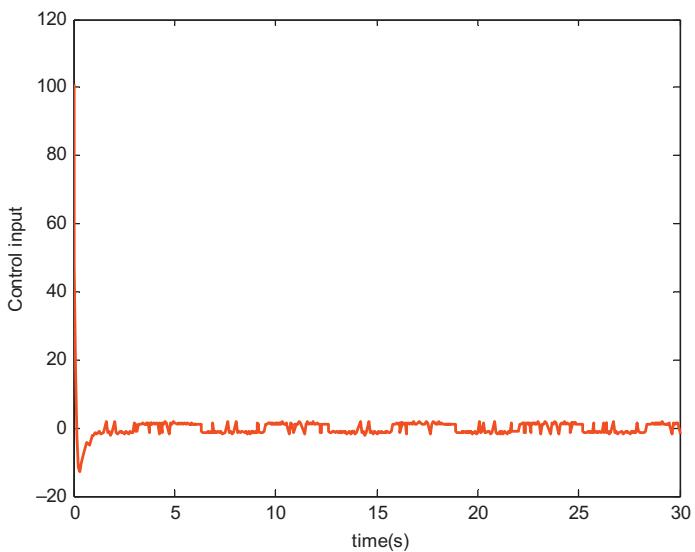
where $J = 2$, $d(t) = \sin t$, the initial states are $\theta(0) = 0$, $\dot{\theta}(0) = 0$.

Choose the position ideal signal $\theta_d = 1.0$, use controller (1.4), choose $c = 10$, $\eta = 1.1$, and set $k = 0$ and $k = 10$, respectively. The results are shown in Figs. 1.1–1.6.

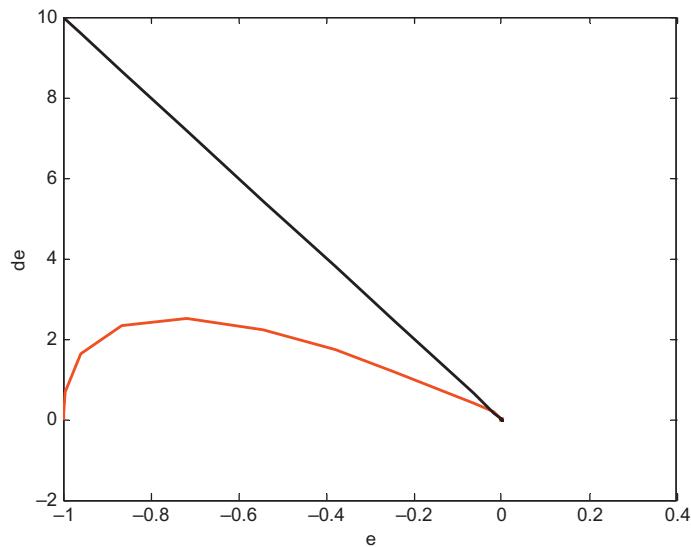
4 CHAPTER 1 Basic sliding mode control principle and design



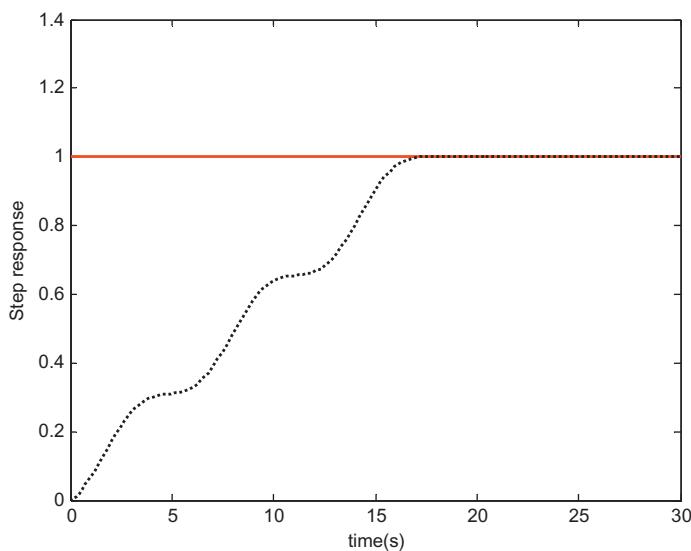
■ FIGURE 1.1 Position tracking with $k = 10$.



■ FIGURE 1.2 Control input with $k = 10$.

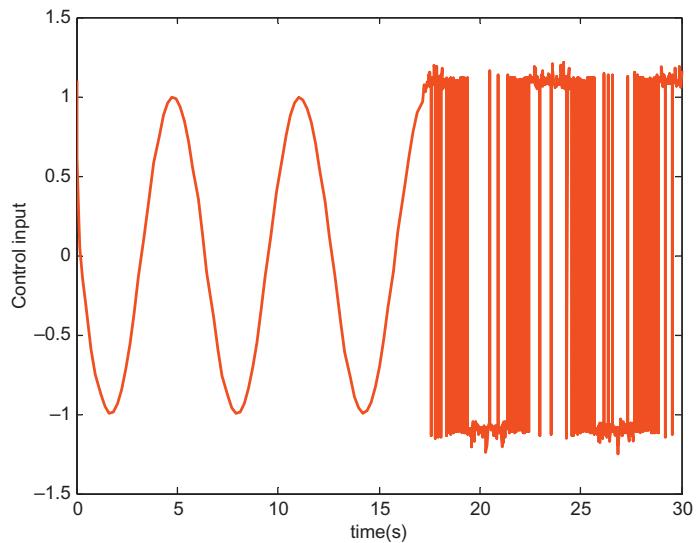


■ FIGURE 1.3 Phase trajectory with $k = 10$.

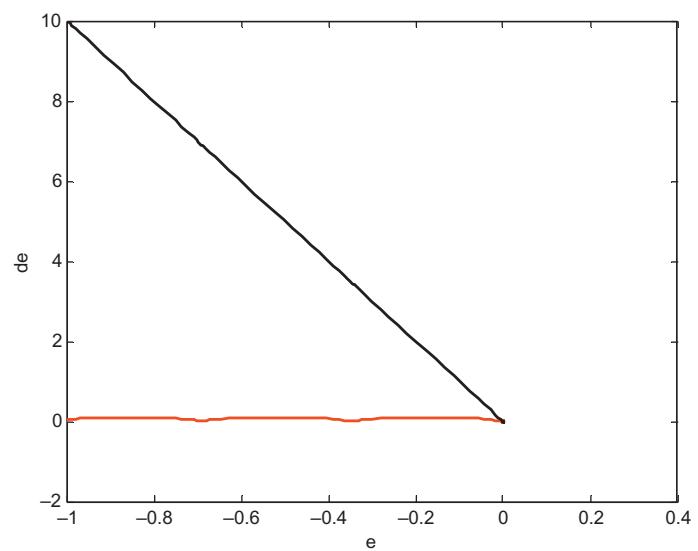


■ FIGURE 1.4 Position tracking with $k = 0$.

6 CHAPTER 1 Basic sliding mode control principle and design



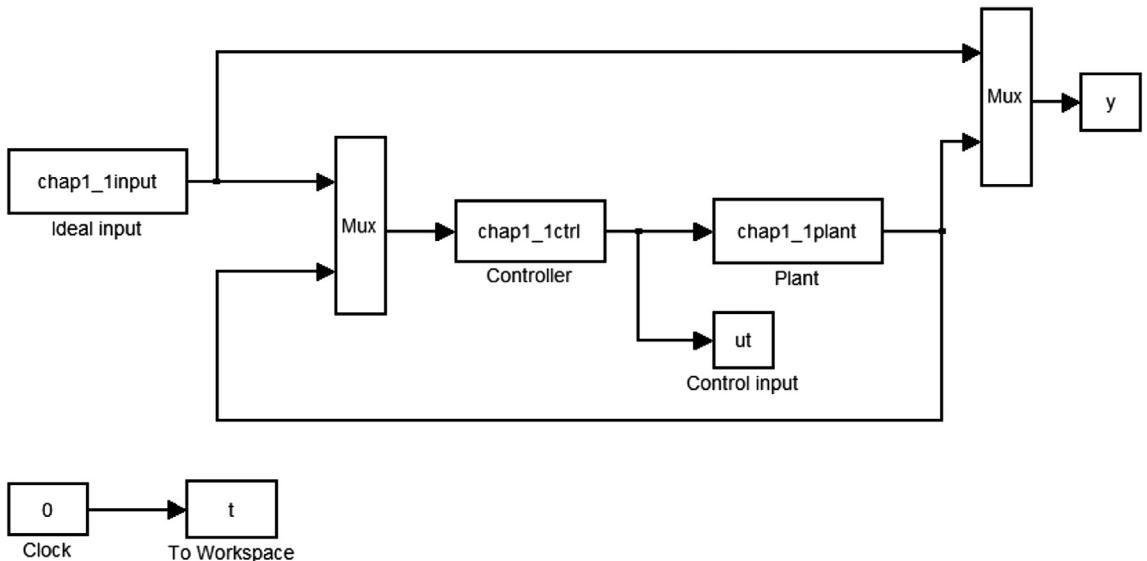
■ FIGURE 1.5 Control input with $k = 0$.



■ FIGURE 1.6 Phase trajectory with $k = 0$.

From the results, we can see that the gain k plays an important part, and the control chattering is caused by the term $\eta sgn(s)$.

1. Simulink main program: chap1_1sim.mdl



2. Controller: chap1_1ctrl.m

```

function [sys,x0,str,ts]=spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag=',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=0;
sizes.NumDiscStates=0;
sizes.NumOutputs=1;
sizes.NumInputs=3;

```

8 CHAPTER 1 Basic sliding mode control principle and design

```
sizes.DirFeedthrough=1;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[];
function sys=mdlOutputs(t,x,u)
J=2;
thd=u(1);
th=u(2);
dth=u(3);

e=th-thd;
de=dth;
c=10;
s=c*e+de;
xite=1.1;

k=0;
%k=10;
ut=J*(-c*dth-1/J*(k*s+xite*sign(s)));
sys(1)=ut;
3. Plant: chap1_1plant.m
function [sys,x0,str,ts]=spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=2;
sizes.NumDiscStates=0;
sizes.NumOutputs=2;
sizes.NumInputs=1;
sizes.DirFeedthrough=0;
```

```

sizes.NumSampleTimes=1;
sys=simsizes(sizes);
x0=[0,0];
str=[];
ts=[0 0];
function sys=mdlDerivatives(t,x,u)
J=2;
dt=sin(t);
ut=u(1);
sys(1)=x(2);
sys(2)=1/J*(ut+dt);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);

```

4. Plot program: chap1_1plot.m

```

close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'k','linewidth',2);
xlabel('time(s)');ylabel('Step response');

figure(2);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input');

c=10;
figure(3);
e=y(:,2)-y(:,1);
de=y(:,3);
plot(e,de,'r',e,-c.*e,'k','linewidth',2);
xlabel('e');ylabel('de');

```

1.2 PARAMETERS DESIGN OF SLIDING MODE FUNCTION

For a nonlinear system

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u + d, \mathbf{x} \in R^n, u \in R, \quad (1.5)$$

a sliding variable can be designed as

$$\mathbf{s}(\mathbf{x}) = \mathbf{C}^T \mathbf{x} = \sum_{i=1}^n c_i x_i = \sum_{i=1}^{n-1} c_i x_i + x_n, \quad (1.6)$$

where \mathbf{x} is a state vector, $x_i = x_i^{n-1}$, $i = 1, \dots, n$ and $\mathbf{C} = [c_1 \dots c_{n-1} 1]^T$.

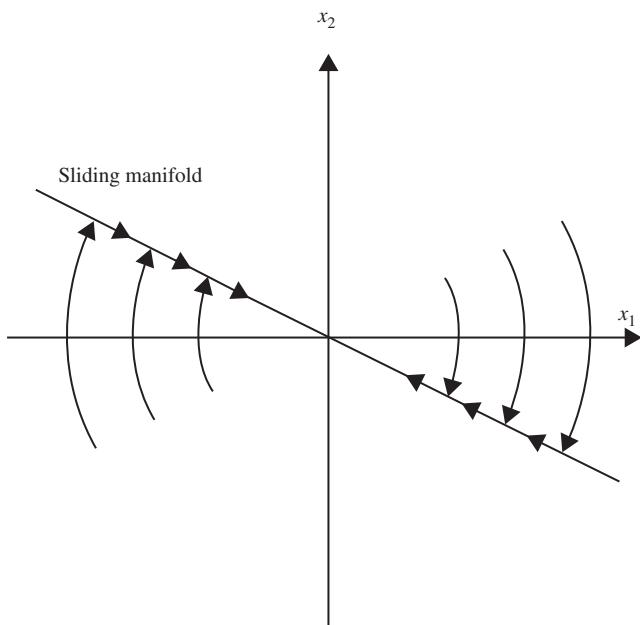
If $s(\mathbf{x}) \rightarrow 0$, to guarantee $x_i \rightarrow 0$, parameters c_1, c_2, \dots, c_{n-1} should be selected so that the polynomial $p^{n-1} + c_{n-1}p^{n-2} + \dots + c_2p + c_1$ is Hurwitz, where p is Laplace operator.

For example, $n = 2$, $s(\mathbf{x}) = c_1x_1 + x_2$, to guarantee the polynomial $p + c_1$ Hurwitz, the eigenvalue of $p + c_1 = 0$ should have negative real part, i.e., $c_1 > 0$.

For another example, $n = 3$, $s(\mathbf{x}) = c_1x_1 + c_2x_2 + x_3$, to guarantee the polynomial $p^2 + c_2p + c_1$ Hurwitz, the eigenvalue of $p^2 + c_2p + c_1 = 0$ should have negative real part. For $\lambda > 0$ in $(p + \lambda)^2 = 0$, we can get $p^2 + 2\lambda p + \lambda^2 = 0$. Therefore, we have $c_2 = 2\lambda$, $c_1 = \lambda^2$.

1.3 SLIDING MODE CONTROL BASED ON REACHING LAW

Sliding mode based on reaching law include reaching phase and sliding phase. The reaching phase drive system to stable manifold, the sliding phase drive system slide to equilibrium. The idea of sliding mode can be described as Fig. 1.7.



■ FIGURE 1.7 The idea of sliding mode.

Four kind of classical reaching laws are given as follows [6]:

1. Reaching law with constant rate

$$\dot{s} = -\varepsilon \text{sgn}s, \quad \varepsilon > 0, \quad (1.7)$$

where ε represents a constant rate.

This law forces the switching variable to reach the switching manifold s at a constant rate ε . The merit of this reaching law is its simplicity. But, as will be shown later, if ε is too small, the reaching time will be too long. On the other hand, too large ε will cause severe chattering.

2. Exponential reaching law

$$\dot{s} = -\varepsilon \text{sgn}s - ks \quad \varepsilon > 0, k > 0, \quad (1.8)$$

where $\dot{s} = -ks$ is an exponential term, and its solution is $s = s(0)e^{-kt}$.

Clearly, by adding the proportional rate term $-ks$, the state is forced to approach the switching manifolds faster when s is larger.

If we use exponential reaching law to design SMC controller, define the Lyapunov function as $V = \frac{1}{2}s^2$, then we can get $\dot{V} \leq -\varepsilon|s| - ks^2 = -\frac{k}{2}V - \varepsilon|s| \leq -\frac{k}{2}V$.

Using Lemma 1.1 to solve the equation $\dot{V} \leq -\frac{k}{2}V$, set $\alpha = \frac{k}{2}, f = 0$; we can get

$$V(t) \leq e^{-\frac{k}{2}(t-t_0)}V(t_0).$$

If k is positive constant value, $V(t)$ will tend to zero exponentially with k value.

In the exponential reaching law, to guarantee faster convergence speed, especially when s is near to zero, the term $\dot{s} = -\varepsilon \text{sgn}(s)$ is used. Moreover, to reduce chattering, we should design a bigger value of k and use a smaller value of ε .

3. Reaching law with power rate

$$\dot{s} = -k|s|^\alpha \text{sgn}s \quad k > 0, 1 > \alpha > 0. \quad (1.9)$$

This reaching law increases the reaching speed when the state is far away from the switching manifold, but reduces the rate when the state is near the manifold. The result is a fast reaching and low chattering reaching mode.

4. General reaching law

$$\dot{s} = -\varepsilon \text{sgn}s - f(s) \quad \varepsilon > 0, \quad (1.10)$$

where $f(0) = 0$, and when $s \neq 0$, $f(s) > 0$.

Obviously, the above four reaching laws can satisfy the sliding mode arrived condition $ss' < 0$.

1.4 ROBUST SLIDING MODE CONTROL BASED ON REACHING LAW

1.4.1 System description

The plant is

$$\ddot{x}(t) = f(x, t) + bu(t) + d(t), \quad (1.11)$$

where $f(x, t)$ and b are known and $b > 0$, $d(t)$, $d(t)$ is unknown disturbance, $|d(t)| \leq D$.

1.4.2 Controller design

The sliding mode function is

$$s(t) = ce(t) + \dot{e}(t), \quad (1.12)$$

where c must satisfy the Hurwitz condition, $c > 0$.

The tracking error and its derivative value is

$$e = x_d - x_1, \quad \dot{e}(t) = \dot{x}_d - \dot{x}_1,$$

where $x_1 = x, x_d$ is ideal position signal.

Therefore, we have

$$\begin{aligned} \dot{s} &= c\dot{e}(t) + \ddot{e}(t) = c(\dot{x}_d - \dot{x}_1) + (\ddot{x}_d - \ddot{x}) \\ &= c(\dot{x}_d - \dot{x}_1) + (\ddot{x}_d - f - bu - d). \end{aligned} \quad (1.13)$$

Adopting the exponential reaching law, we have

$$\dot{s} = -\varepsilon sgn s - ks \quad \varepsilon > 0, k > 0. \quad (1.14)$$

From Eqs. (1.13) and (1.14), we have

$$c(\dot{x}_d - \dot{x}_1) + (\ddot{x}_d - f - bu - d) = -\varepsilon sgn s - ks.$$

If we design the sliding mode controller as

$$u(t) = \frac{1}{b}(-\varepsilon sgn s - ks + c(\dot{x}_d - \dot{x}_1) + \ddot{x}_d - f - d). \quad (1.15)$$

Obviously, all quantities on the right-hand side of (1.15) are known except the disturbance d , which is unknown. Thus control law (1.15) cannot be realized.

Design the sliding mode controller as

$$u(t) = \frac{1}{b}(-\varepsilon sgn s - ks + c(\dot{x}_d - \dot{x}_1) + \ddot{x}_d - f + Dsgns). \quad (1.16)$$

Substituting Eq. (1.16) into Eq. (1.13) and simplifying the result, we have

$$\dot{s}(t) = -\varepsilon sgn s - ks - Dsgns - d. \quad (1.17)$$

We then get

$$s\dot{s}(t) = s(-\varepsilon \text{sgn} s - ks - D \text{sgn} s - d) = -ks^2 - \varepsilon|s| - D|s| - ds.$$

Define the Lyapunov function as $V = \frac{1}{2}s^2$, then we can get

$$\dot{V} = s\dot{s}(t) = -ks^2 - \varepsilon|s| - D|s| - ds \leq -2kV.$$

Using Lemma 1.3, we have

$$V(t) \leq e^{-2kt}V(0).$$

We can see that the sliding mode function $s(t)$ will tend to zero exponentially with k value.

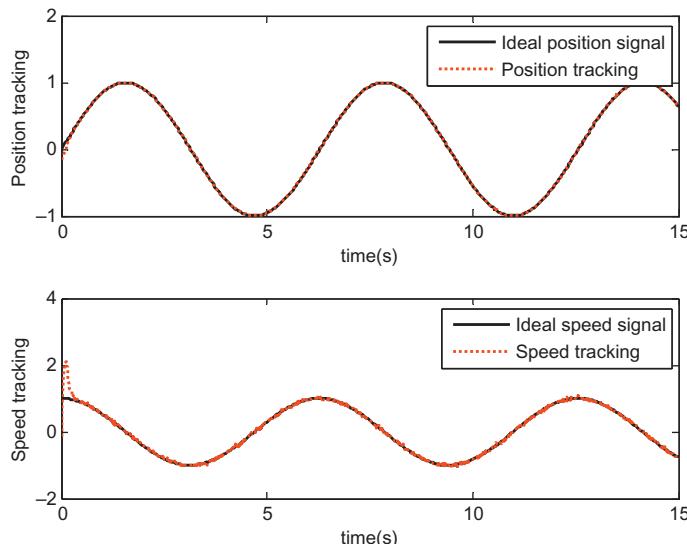
1.4.3 Simulation example

Consider the plant as

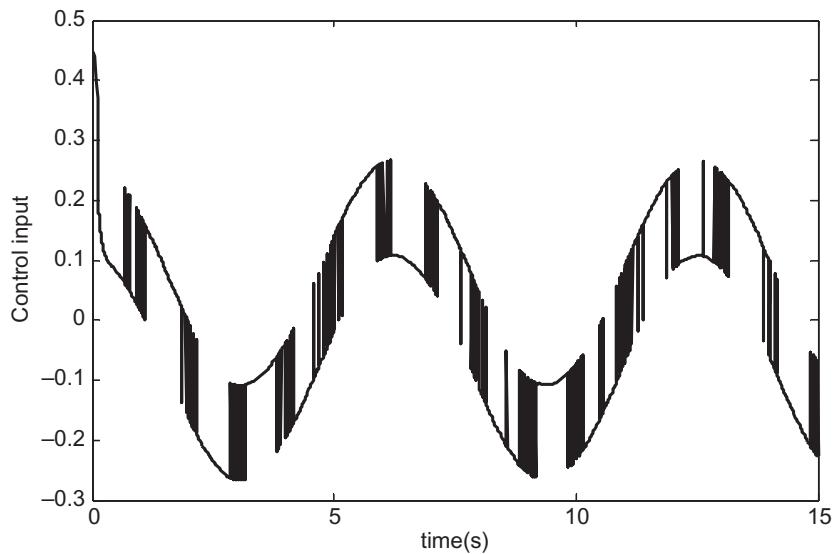
$$\ddot{x}(t) = f(x, t) + bu(t) + d(t),$$

where $f = -25\dot{x}$, $b = 133$ and $d(t) = 10\sin(\pi t)$.

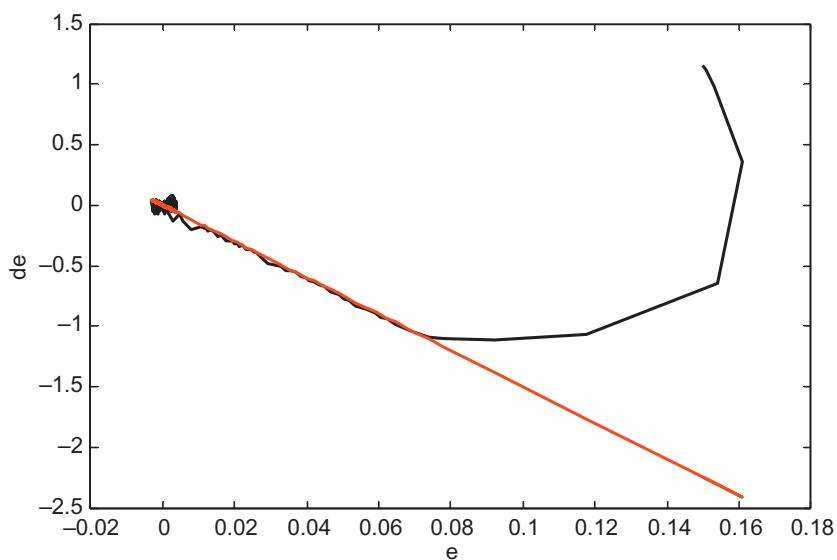
Choose the ideal position signal $x_d = \sin t$, set the initial states as $[-0.15 \ -0.15]$, use controller (1.16), set $c = 15$, $\varepsilon = 0.5$, $k = 10$, $D = 10.1$, the results can be seen in Figs 1.8–1.10.



■ FIGURE 1.8 Position and Speed tracking.

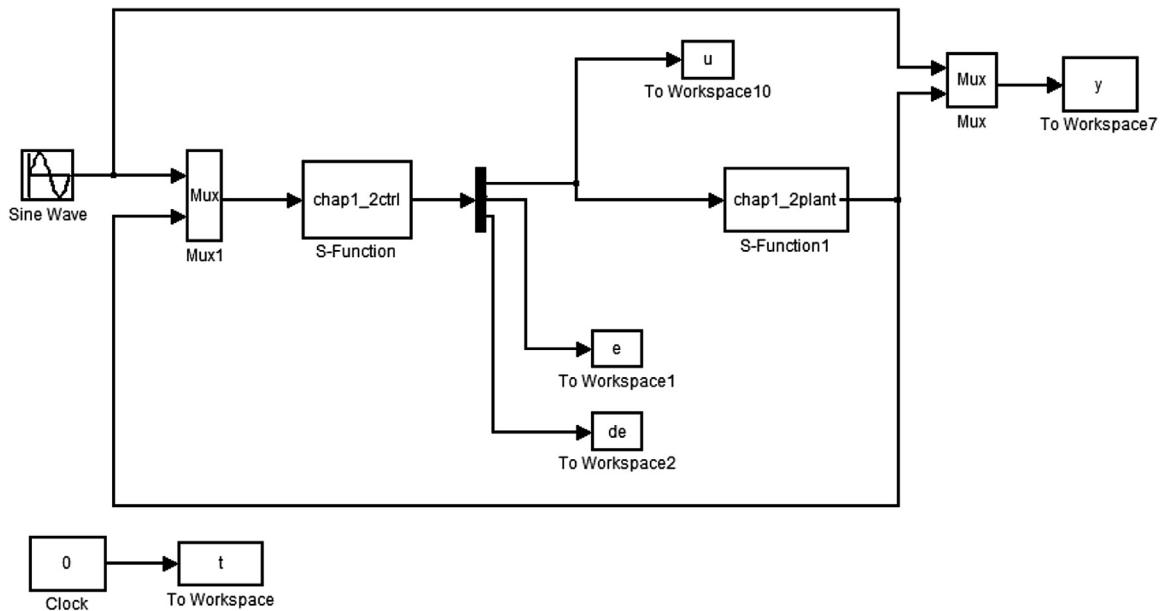


■ FIGURE 1.9 Control input.



■ FIGURE 1.10 Phase trajectory.

1. Simulink main program: chap1_2sim.mdl



2. Controller: chap1_2ctrl.m

```

function [sys,x0,str,ts]=spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=0;
sizes.NumDiscStates=0;
sizes.NumOutputs=3;
sizes.NumInputs=3;
sizes.DirFeedthrough=1;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
  
```

16 CHAPTER 1 Basic sliding mode control principle and design

```
x0 = [];
str = [];
ts = [];
function sys = mdlOutputs(t,x,u)
xd = u(1);
dxd = cos(t);
ddxd = - sin(t);

x1 = u(2);
dx1 = u(3);

c = 15;
e = xd - x1;
de = dxd - dx1;
s = c*e + de;

fx = - 25*dx1;b = 133;
D = 10.1;

epc = 0.5;k = 10;
ut = 1/b*(epc*sign(s) + k*s + c*de + ddxd - fx + D*sign(s));

sys(1)=ut;
sys(2)=e;
sys(3)=de;
```

3. Plant: chap1_2plant.m

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=2;
sizes.NumDiscStates=0;
sizes.NumOutputs=2;
sizes.NumInputs=1;
```

```

sizes.DirFeedthrough=0;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[-0.15 -0.15];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
sys(1)=x(2);
sys(2)=-25*x(2)+133*u+10*sin(pi*t);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
4. Plot: chap1_2plot.m
close all;
figure(1);
subplot(211);
plot(t,y(:,1),'k',t,y(:,2),'r','LineWidth',2);
legend('Ideal position signal','Position tracking');
xlabel('time(s)');ylabel('Position tracking');
subplot(212);
plot(t,cos(t),'k',t,y(:,3),'r','LineWidth',2);
legend('Ideal speed signal','Speed tracking');
xlabel('time(s)');ylabel('Speed tracking');
figure(2);
plot(t,u(:,1),'k','LineWidth',.2);
xlabel('time(s)');ylabel('Control input');
c=15;
figure(3);
plot(e,de,'k',e,-c.*e,'r','LineWidth',2);
xlabel('e');ylabel('de');

```

1.5 SLIDING MODE CONTROL BASED ON A QUASI-SLIDING MODE

Consider the plant as

$$\ddot{x}(t) = f(x, t) + bu(t) + d(t). \quad (1.18)$$

Use controller (1.16) in Section 1.4 as follows:

$$u(t) = \frac{1}{b}(\varepsilon sgn s + ks + c(\dot{x}_d - \dot{x}_1) + \ddot{x}_d - f + Dsgns), \quad (1.19)$$

where $f(x, t)$ and b are known and $b > 0$, $d(t)$ is unknown disturbance, $|d(t)| \leq D$.

In sliding mode controller, due to the switch term $\varepsilon \text{sgn}s$, the chattering can be caused, especially for a large disturbance, which will damage to system components such as actuators.

One way to alleviate the chattering is to use quasi-sliding mode method, which can make the state stay in a certain range at Δ neighborhood. Often we name Δ as the boundary layer.

Instead of the sign function, in continuous system, there are commonly two methods for quasi-sliding mode design.

1. Saturation function

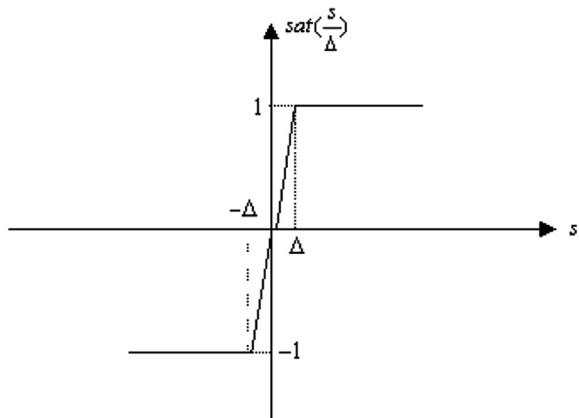
$$\text{sat}(s) = \begin{cases} 1 & s > \Delta \\ ks & |s| \leq \Delta \\ -1 & s < -\Delta \end{cases} \quad k = \frac{1}{\Delta}, \quad (1.20)$$

where Δ is called “the boundary layer”, which is shown in Fig. 1.11. Outside the boundary layer, we use switch control, inside the boundary layer, we use linear feedback control, then chattering can be eliminated.

2. Relay function

$$\theta(s) = \frac{s}{|s| + \delta}, \quad (1.21)$$

where δ is a very small positive constant.



■ FIGURE 1.11 Saturation function.

Several examples with quasi-sliding mode are given in [1,2]. Recently, there is a new quasi-sliding mode function, i.e., tanh function, which is given as follows.

1.6 SLIDING MODE CONTROL BASED ON CONTINUOUS TANH FUNCTION

1.6.1 Characteristics of tanh function

The discontinuity of the sign function in the SMC law can cause the chattering. Therefore, in order to avoid the chattering, the discontinuous sign function can be replaced by the continuous tanh function [7]. In other word, the function is used as an approximator of the sign function. The steepness of the tanh function determines that how the tanh can approximate the sign function.

The hyperbolic tangent function is

$$\tanh\left(\frac{x}{\varepsilon}\right) = \frac{e^{\frac{x}{\varepsilon}} - e^{-\frac{x}{\varepsilon}}}{e^{\frac{x}{\varepsilon}} + e^{-\frac{x}{\varepsilon}}}, \quad (1.22)$$

where $\varepsilon > 0$, the steepness of the tanh function is determined by ε value.

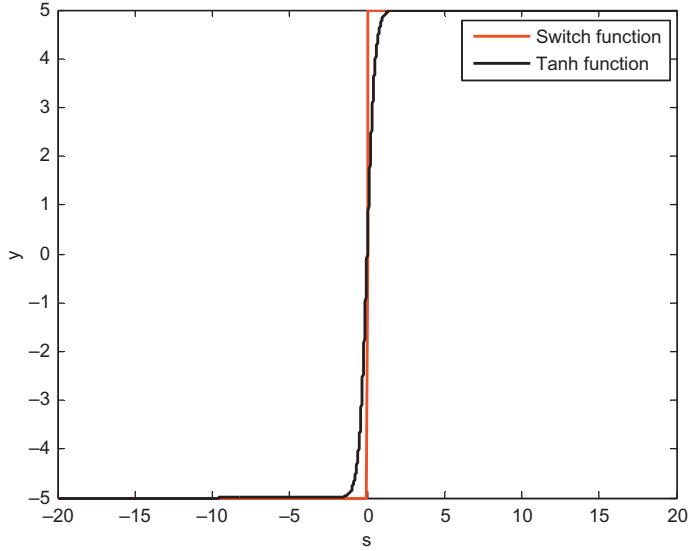
1.6.2 Simulation example

Choose $\varepsilon = 0.50$, the comparison of the tanh function and switch function are given in Fig. 1.12.

```
Simulation example: tanh_test.m
clearall;
closeall;

xite=5.0;
ts=0.01;
for k=1:1:4000;
    s(k)=k*ts-20;
    y1(k)=xite*sign(s(k));
    epc=0.5;
    y2(k)=xite*tanh(s(k)/epc);
end

figure(1);
plot(s,y1,'r',s,y2,'k','linewidth',2);
xlabel('s');ylabel('y');
legend('Switch function','Tanh function');
```



■ FIGURE 1.12 Tanh function and switch function.

1.6.3 Sliding mode control based on tanh function

Consider the following plant as

$$J\ddot{\theta}(t) = u(t) + d(t), \quad (1.23)$$

where J is the moment of inertia, $\theta(t)$ is the angle signal, $u(t)$ is the control input, $d(t)$ is disturbance and $|dt| \leq D$.

Then the sliding mode function is:

$$s(t) = ce(t) + \dot{e}(t),$$

where c must satisfy Hurwitz condition, $c > 0$.

Tracking error and its derivatives value is

$$e(t) = \theta(t) - \theta_d(t), \quad \dot{e}(t) = \dot{\theta}(t) - \dot{\theta}_d(t),$$

where $\theta_d(t)$ is ideal angle signal.

Define the Lyapunov function as

$$V = \frac{1}{2}s^2,$$

then

$$\dot{s}(t) = c\dot{e}(t) + \ddot{e}(t) = c\dot{e}(t) + \ddot{\theta}(t) - \ddot{\theta}_d(t) = c\dot{e}(t) + \frac{1}{J}(u + d(t)) - \ddot{\theta}_d(t)$$

and

$$s\dot{s} = s \left(c\dot{e} + \frac{1}{J}(u + d(t)) - \ddot{\theta}_d \right)$$

To guarantee $s\dot{s} < 0$, we consider two controllers as follows.

1. SMC based on switch function

$$u(t) = J(-c\dot{e} + \ddot{\theta}_d - \eta s) - D\text{sgn}(s), \quad (1.24)$$

then

$$\begin{aligned} s\dot{s} &= s \left(c\dot{e} + (-c\dot{e} + \ddot{\theta}_d - \eta s) - \frac{1}{J}D\text{sgn}(s) + \frac{1}{J}d(t) - \ddot{\theta}_d \right) \\ &= s \left(-\eta s - \frac{1}{J}D\text{sgn}(s) + \frac{1}{J}d(t) \right) \\ &= -\eta s^2 - \frac{1}{J}D|s| + \frac{1}{J}sd(t) \leq -\eta s^2 = -2\eta V \end{aligned}$$

Then we have $\dot{V} \leq -2\eta V$; using Lemma 1.3, we get the solution as

$$V(t) \leq e^{-2\eta(t-t_0)}V(t_0).$$

According to the above inequality, we can conclude that the tracking error are exponential convergence, and the convergent precision is determined by η .

2. SMC based on tanh function

$$u(t) = J(-c\dot{e} + \ddot{\theta}_d - \eta s) - D\tanh\left(\frac{s}{\varepsilon}\right). \quad (1.25)$$

According to Lemma 1.2, we have

$$|s| - \text{stanh}\left(\frac{s}{\varepsilon}\right) \leq \mu\varepsilon. \quad (1.26)$$

Then $D|s| - D\tanh\left(\frac{s}{\varepsilon}\right) \leq D\mu\varepsilon$, that is

$$-D\tanh\left(\frac{s}{\varepsilon}\right) \leq -D|s| + D\mu\varepsilon,$$

therefore

$$\begin{aligned} s\dot{s} &= s \left(c\dot{e} + \frac{1}{J}(u + d(t)) - \ddot{\theta}_d \right) \\ &= s \left(c\dot{e} + (-c\dot{e} + \ddot{\theta}_d - \eta s) - \frac{1}{J}D\tanh\left(\frac{s}{\varepsilon}\right) + \frac{1}{J}d(t) - \ddot{\theta}_d \right) \\ &= s \left(-\eta s - \frac{1}{J}D\tanh\left(\frac{s}{\varepsilon}\right) + \frac{1}{J}d(t) \right) \\ &= -\eta s^2 + \frac{1}{J} \left(-Ds\tanh\left(\frac{s}{\varepsilon}\right) + sd(t) \right) \\ &\leq -\eta s^2 + \frac{1}{J}(-D|s| + D\mu\varepsilon + sd(t)) \\ &\leq -\eta s^2 + \frac{1}{J}D\mu\varepsilon = -2\eta V + b \end{aligned}$$

where $b = \frac{1}{J}D\mu\varepsilon$.

Then we have $\dot{V} \leq -2\eta V + b$, use Lemma 1.3, we get the solution as

$$\begin{aligned} V(t) &\leq e^{-2\eta(t-t_0)} V(t_0) + b e^{-2\eta t} \int_{t_0}^t e^{2\eta\tau} d\tau = e^{-2\eta(t-t_0)} V(t_0) + \frac{b e^{-2\eta t}}{2\eta} (e^{2\eta t} - e^{2\eta t_0}) \\ &= e^{-2\eta(t-t_0)} V(t_0) + \frac{b}{2\eta} (1 - e^{-2\eta(t-t_0)}) \\ &= e^{-2\eta(t-t_0)} V(t_0) + \frac{D\mu\varepsilon}{2\eta J} (1 - e^{-2\eta(t-t_0)}) \end{aligned}$$

then

$$\lim_{t \rightarrow \infty} V(t) \leq \frac{D\mu\varepsilon}{2\eta J}. \quad (1.27)$$

According to inequality (1.27), we can conclude that the tracking error are asymptotically convergence, and the convergent precision is determined by D , η and ε .

1.6.4 Simulation example

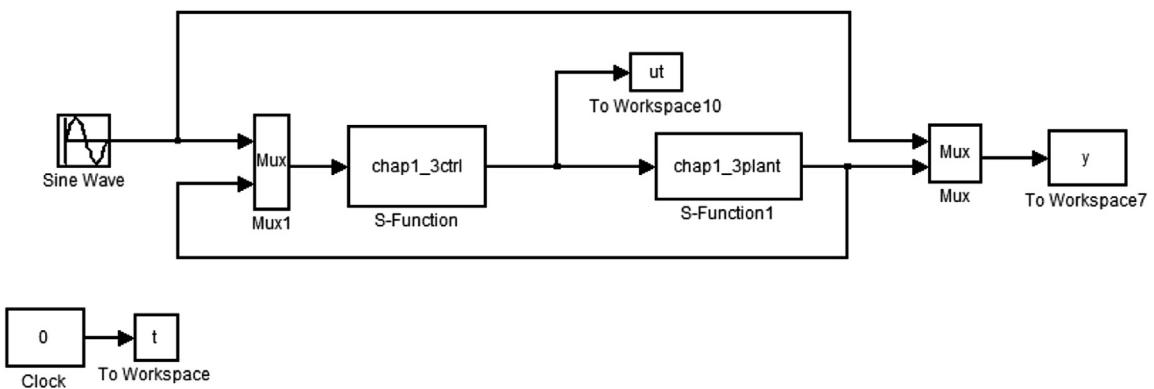
Consider the plant as

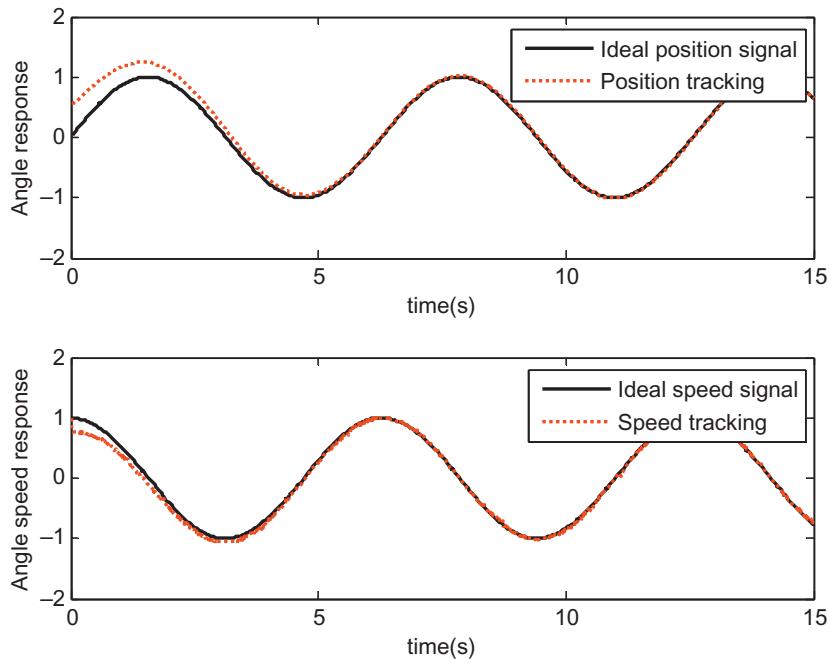
$$J\ddot{\theta}(t) = u(t) + d(t).$$

We choose $J = 10$, ideal angle signals $\theta_d(t) = \sin t$ and $d(t) = 50 \sin t$, and we choose the initial states of the plant as $[0.5, 1.0]$. We set $c = 0.50$, $\eta = 0.50$, $D = 50$ and $\varepsilon = 0.02$, and use control laws (1.24) and (1.25), respectively. The simulation results are shown in Figs. 1.13–1.16.

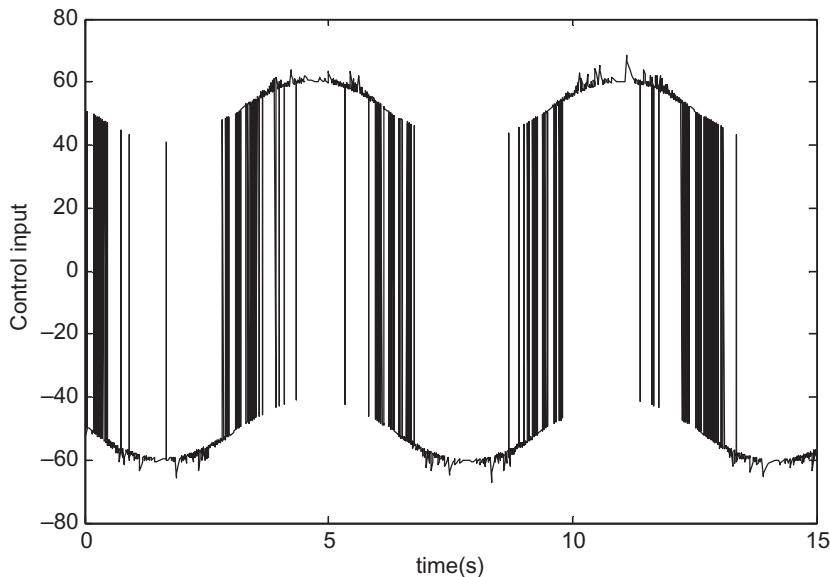
Simulation programs:

1. Simulink program: chap1_3sim.mdl

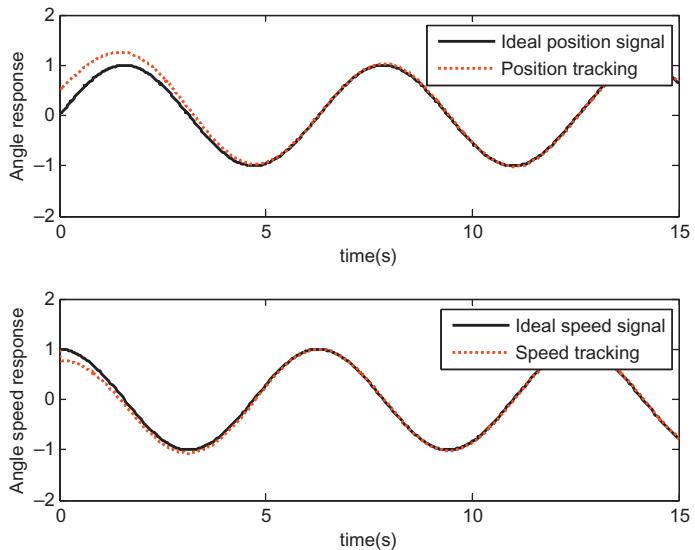




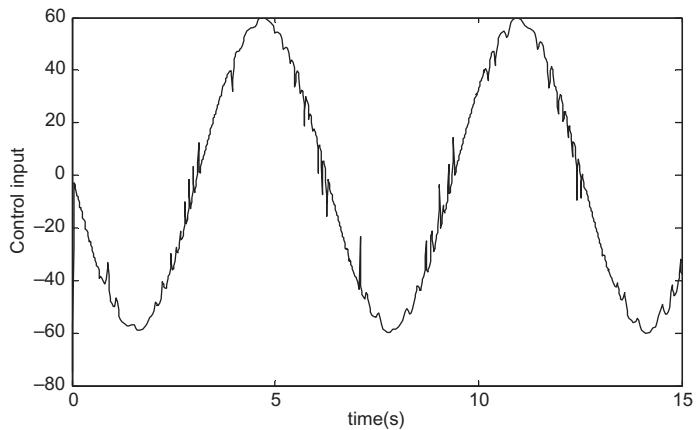
■ FIGURE 1.13 Position and speed tracking with switch function.



■ FIGURE 1.14 Control input with switch function.



■ FIGURE 1.15 Position and Speed tracking with hyperbolic tangent function.



■ FIGURE 1.16 Control input with hyperbolic tangent function.

2. Control law S function: chap1_3ctrl.m

```
function [sys,x0,str,ts]=spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
```

```
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=0;
sizes.NumDiscStates=0;
sizes.NumOutputs=1;
sizes.NumInputs=3;
sizes.DirFeedthrough=1;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[];
function sys=mdlOutputs(t,x,u)
thd=u(1);
dthd=cos(t);
ddthd=-sin(t);

th=u(2);
dth=u(3);

c=0.5;
e=th-thd;
de=dth-dthd;
s=c*e+de;

J=10;
xite=10;
D=50;
epc=0.02;

M=2;
if M==1
    ut=J*(-c*de+ddthd-xite*s)-D*sign(s);
elseif M==2
    ut=J*(-c*de+ddthd-xite*s)-D*tanh(s/epc);
end
sys(1)=ut;
```

26 CHAPTER 1 Basic sliding mode control principle and design

3. Plant S function: chap1_3plant.m

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=2;
sizes.NumDiscStates=0;
sizes.NumOutputs=2;
sizes.NumInputs=1;
sizes.DirFeedthrough=0;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[0.5 1.0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
J=10;
dt=50*sin(t);
sys(1)=x(2);
sys(2)=1/J*(u+dt);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
```

4. Plot program: chap1_3plot.m

```
close all;

figure(1);
subplot(211);
plot(t,y(:,1),'k',t,y(:,2),'r','linewidth',2);
legend('Ideal position signal','Position tracking');
```

```

xlabel('time(s)');ylabel('Angle response');
subplot(212);
plot(t,cos(t),'k',t,y(:,3),'r','linewidth',2);
legend('Ideal speed signal','Speed tracking');
xlabel('time(s)');ylabel('Angle speed response');

figure(2);
plot(t,ut(:,1),'k','linewidth',0.01);
xlabel('time(s)');ylabel('Control input');

```

REFERENCES

- [1] J. Liu, Matlab Simulation for Sliding Mode Control, third edition, Tsinghua & Springer Press, Beijing, 2015, in Chinese.
- [2] J. Liu, X. Wang, Advanced Sliding Mode Control for Mechanical Systems Design, Analysis and Matlab Simulation, Tsinghua & Springer Press, Beijing, 2011.
- [3] U. Itkis, Control System of Variable Structure, Wiley, New York, 1976.
- [4] J.Y. Hung, W. Gao, J.C. Hung, Variable structure control: a survey, IEEE Trans. Ind. Electronics 40 (1) (1993) 2–22.
- [5] C. Edwards, S. Spurgeon, Sliding Mode Control: Theory and Applications, Taylor and Francis, London, 1998.
- [6] W. Gao, J.C. Hung, Variable structure control of nonlinear systems: a new approach, IEEE Trans. Ind. Electronics 40 (1) (1993) 45–55.
- [7] M.P. Aghababa, M.E. Akbari, A chattering-free robust adaptive sliding mode controller for synchronization of two different chaotic systems with unknown uncertainties and external disturbances, Appl. Math. Computat. 218 (2012) 5757–5768.
- [8] M.M. Polycarpou, P.A. Ioannou, A robust adaptive nonlinear control design, IEEE Am. Control Conf. (1993) 1365–1369.
- [9] P.A. Ioannou, J. Sun, Robust Adaptive Control, PTR Prentice-Hall, Upper Saddle River, 1996, pp. 75–76.

APPENDIX

Lemma 1.1: [7] For every given scalar x and positive scalar ε , the following inequality holds:

$$xtanh\left(\frac{x}{\varepsilon}\right) = \left|xtanh\left(\frac{x}{\varepsilon}\right)\right| = |x|\left|\tanh\left(\frac{x}{\varepsilon}\right)\right| \geq 0. \quad (\text{A1})$$

Lemma 1.1 can be proved as follows: according to the definition of tanh function, we have

$$xtanh\left(\frac{x}{\varepsilon}\right) = x \frac{e^{\frac{x}{\varepsilon}} - e^{-\frac{x}{\varepsilon}}}{e^{\frac{x}{\varepsilon}} + e^{-\frac{x}{\varepsilon}}} = \frac{1}{e^{2\frac{x}{\varepsilon}} + 1} x(e^{2\frac{x}{\varepsilon}} - 1).$$

Since

$$\begin{aligned} e^{2\frac{x}{\varepsilon}} - 1 &\geq 0 \quad \text{if } x \geq 0 \\ e^{2\frac{x}{\varepsilon}} - 1 &< 0 \quad \text{if } x < 0, \end{aligned}$$

then

$$x(e^{2\frac{x}{\varepsilon}} - 1) \geq 0,$$

therefore

$$x \tanh\left(\frac{x}{\varepsilon}\right) = \frac{1}{e^{2\frac{x}{\varepsilon}} + 1} x(e^{2\frac{x}{\varepsilon}} - 1) \geq 0,$$

and

$$x \tanh\left(\frac{x}{\varepsilon}\right) = \left| \tanh\left(\frac{x}{\varepsilon}\right) \right| = |x| \left| \tanh\left(\frac{x}{\varepsilon}\right) \right| \geq 0.$$

Lemma 1.2: [8] For every given scalar $\chi \in R$ and positive scalar ε , the following inequality holds:

$$0 \leq |\chi| - \chi \tanh\left(\frac{\chi}{\varepsilon}\right) \leq \mu\varepsilon, \mu = 0.2785. \quad (\text{A2})$$

Lemma 1.3: [9] Let $f, V: [0, \infty) \in R$, then $\dot{V} \leq -\alpha V + f$, $\forall t \geq t_0 \geq 0$ implies that

$$V(t) \leq e^{-\alpha(t-t_0)} V(t_0) + \int_{t_0}^t e^{-\alpha(t-\tau)} f(\tau) d\tau$$

For any finite constant α .

According to [9], we have the proof as follows:

Let $\omega(t) \triangleq \dot{V} + \alpha V - f$, we have $\omega(t) \leq 0$, and

$$\dot{V} = -\alpha V + f + \omega$$

implies that

$$V(t) = e^{-\alpha(t-t_0)} V(t_0) + \int_{t_0}^t e^{-\alpha(t-\tau)} f(\tau) d\tau + \int_{t_0}^t e^{-\alpha(t-\tau)} \omega(\tau) d\tau.$$

Because $\omega(t) < 0$ and $\forall t \geq t_0 \geq 0$, we have

$$V(t) \leq e^{-\alpha(t-t_0)} V(t_0) + \int_{t_0}^t e^{-\alpha(t-\tau)} f(\tau) d\tau.$$

Moreover, if we choose $f = 0$, then we have $\dot{V} \leq -\alpha V$, implies that

$$V(t) \leq e^{-\alpha(t-t_0)} V(t_0).$$

If α is a positive constant value, $V(t)$ will tend to zero exponentially with α value.

Sliding mode control with high performance

In this chapter, two sections are introduced as follows: in Section 2.1, an adaptive sliding mode control for mechanical systems with tanh function is introduced, to avoid control input value too big, a projection algorithm is used. In Section 2.2, the problem of tracking control with prescribed performance guarantees is considered; the sliding mode control objective is to realize that the error evolution is within prescribed performance bounds.

2.1 ADAPTIVE SLIDING MODE CONTROL FOR MECHANICAL SYSTEMS WITH TANH FUNCTION

2.1.1 System description

The uncertain mechanical system is described as

$$\frac{dx_1}{dt} = x_2, \quad (2.1)$$

$$m \frac{dx_2}{dt} = u(t) + \Delta, \quad (2.2)$$

where x_1 and x_2 are the position and velocity respectively. m is the unknown moment of inertia and m is a positive constant value, and Δ the uncertainty including matched and unmatched disturbances.

Denote $\theta = m$, therefore, Eq. (2.2) can be written as:

$$\theta \frac{dx_2}{dt} = u + \Delta. \quad (2.3)$$

Assumption 1: The upper boundness of the uncertain parameter θ is defined as:

$$\theta \in \Omega \triangleq \{\theta : 0 < \theta_{\min} \leq \theta \leq \theta_{\max}\}. \quad (2.4)$$

Assumption 2: The uncertainty Δ is bounded, and

$$|\Delta| \leq D. \quad (2.5)$$

2.1.2 Adaptive sliding mode controller design

We select sliding variable as

$$\begin{aligned} s &= \dot{e} + ce = x_2 - q \\ q &= \dot{x}_d - ce \end{aligned}, \quad (2.6)$$

where $e = x_1 - x_d$ is the position tracking error, and $c > 0$ is a constant.

Therefore, we have $\dot{s} = \dot{x}_2 - \dot{q}$, the controller is selected as

$$u = u_a + u_{s1} + u_{s2}, \quad (2.7)$$

where

$$u_a = \hat{\theta}\dot{q}, \quad (2.8)$$

$$u_{s1} = -k_s s, \quad (2.9)$$

$$u_{s2} = -\eta \tanh\left(\frac{s}{\varepsilon}\right), \quad (2.10)$$

and u_a is the adaptive compensation, u_{s1} is the feedback item, u_{s2} is the robustness item and $k_s > 0, \eta > D, \varepsilon > 0$.

To decrease chattering, in the controller robust term, we use the tanh function instead of the switch function, i.e., use $u_{s2} = -\eta \tanh\left(\frac{s}{\varepsilon}\right)$ instead of $u_{s2} = -\eta \text{sign}(s)$.

Therefore, controller (2.7) can be rewritten as

$$u = \hat{\theta}\dot{q} - k_s s - \eta \tanh\left(\frac{s}{\varepsilon}\right).$$

Select the Lyapunov function as

$$V = \frac{1}{2}\theta s^2 + \frac{1}{2\gamma}\tilde{\theta}^2, \quad (2.11)$$

where $\tilde{\theta} = \hat{\theta} - \theta, \gamma > 0$.

We then get

$$\dot{V} = \theta s \dot{s} + \frac{1}{\gamma} \tilde{\theta} \dot{\tilde{\theta}} = s(\theta \dot{x}_2 - \theta \dot{q}) + \frac{1}{\gamma} \tilde{\theta} \dot{\tilde{\theta}}.$$

Design the adaptive law as

$$\dot{\hat{\theta}} = -\gamma \dot{q} s. \quad (2.12)$$

Therefore,

$$\begin{aligned}\dot{V} &= s(u + \Delta - \theta\dot{q}) + \frac{1}{\gamma}\tilde{\theta}\dot{\theta} \\ &= s\left(\tilde{\theta}\dot{q} - k_ss - \eta \tanh\left(\frac{s}{\varepsilon}\right) + \Delta - \theta\dot{q}\right) + \frac{1}{\gamma}\tilde{\theta}(-\gamma\dot{q}s) \\ &= s\left(\tilde{\theta}\dot{q} - k_ss - \eta \tanh\left(\frac{s}{\varepsilon}\right) + \Delta\right) - \tilde{\theta}(\dot{q}s)\end{aligned}$$

According to Eq. (1.26), we have

$$-\eta s \tanh\left(\frac{s}{\varepsilon}\right) \leq -\eta|s| + \eta\mu\varepsilon.$$

Then

$$\dot{V} = -k_ss^2 - \eta|s| + \eta\mu\varepsilon + \Delta \cdot s \leq -k_ss^2 + \eta\mu\varepsilon.$$

If $k_ss^2 \geq \eta\mu\varepsilon$, i.e., $|s| \geq \sqrt{\frac{\eta\mu\varepsilon}{k_s}}$, we have $-k_ss^2 + \eta\mu\varepsilon \leq 0$, i.e., $\dot{V} \leq 0$, then the convergence value is $|s| \leq \sqrt{\frac{\eta\mu\varepsilon}{k_s}}$, if k_s is bigger enough, η and ε is smaller enough, the convergence value will tend to be very smaller.

Since if $V \geq 0$ and $\dot{V} \leq 0$, we can get that if $t \rightarrow \infty$, V is limited, and $\hat{\theta}$ is limited.

In order to avoid $\hat{\theta}$ is too far from θ , and avoid control input value too big, we use projection algorithm [1] instead of Eq. (2.12) as follows:

$$\dot{\hat{\theta}} = \text{Proj}_{\hat{\theta}}(-\gamma\dot{q}s), \quad (2.13)$$

$$\text{Proj}_{\hat{\theta}}(\cdot) = \begin{cases} 0 & \text{if } \hat{\theta} = \theta_{\max} \text{ and } \cdot > 0 \\ 0 & \text{if } \hat{\theta} = \theta_{\min} \text{ and } \cdot < 0 \\ \cdot & \text{otherwise} \end{cases}$$

For adaptive law (2.13) and use switch function for (2.10), we have

1. if $\hat{\theta} = \theta_{\max}$ and $-\gamma\dot{q}s > 0$, set $\dot{\hat{\theta}} = 0$, then $\tilde{\theta} = \hat{\theta} - \theta > 0$, $\gamma\dot{q}s < 0$, we have $s\tilde{\theta}\dot{q} < 0$, and

$$\begin{aligned}\dot{V} &= s(\tilde{\theta}\dot{q} - k_ss - \eta\text{sign}(s) + \Delta) \\ &= s\tilde{\theta}\dot{q} - k_ss^2 - \eta|s| + \Delta \cdot s \leftarrow -k_ss^2 \leq 0\end{aligned}$$

2. if $\hat{\theta} = \theta_{\min}$ and $-\gamma\dot{q}s < 0$, set $\dot{\hat{\theta}} = 0$, then $\tilde{\theta} = \hat{\theta} - \theta < 0$, $\gamma\dot{q}s > 0$, we have $s\tilde{\theta}\dot{q} < 0$, and

$$\begin{aligned}\dot{V} &= s(\tilde{\theta}\dot{q} - k_ss - \eta\text{sign}(s) + \Delta) \\ &= s\tilde{\theta}\dot{q} - k_ss^2 - \eta|s| + \Delta \cdot s \leftarrow -k_ss^2 \leq 0\end{aligned}$$

3. if $\dot{\hat{\theta}} = -\gamma \dot{q}_s$, then Eq. (2.13) is the same as Eq. (2.12), and we have
 $\dot{V} \leq -k_s s^2 \leq 0$.

From $\dot{V} \leq -k_s s^2 \leq 0$, using LaSalle-Yoshizawa theorem [5], we have

$$\lim_{t \rightarrow \infty} k_s s^2 = 0$$

This is, when $t \rightarrow \infty$, $s \rightarrow 0$.

From the above analysis, we can get $\dot{V} \leq 0$ with adaptive law (2.13), and $\hat{\theta}$ will be limited within $[\theta_{\min}, \theta_{\max}]$.

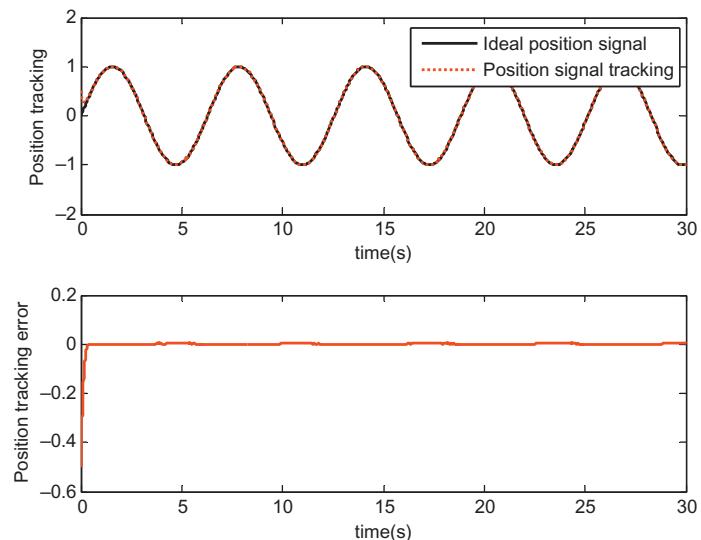
2.1.3 Simulation example

Consider a linear plant is

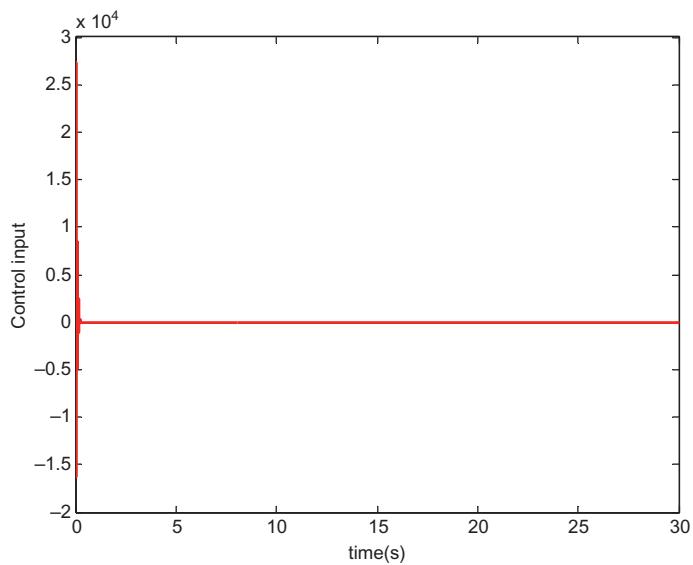
$$\begin{aligned} \frac{dx_1}{dt} &= x_2 \\ m \frac{dx_2}{dt} &= u(t) + \Delta, \end{aligned}$$

where $m = 1.0$, Δ is the friction model and denoted as $\Delta = 0.5\dot{\theta} + 1.5\text{sign}(\dot{\theta})$.

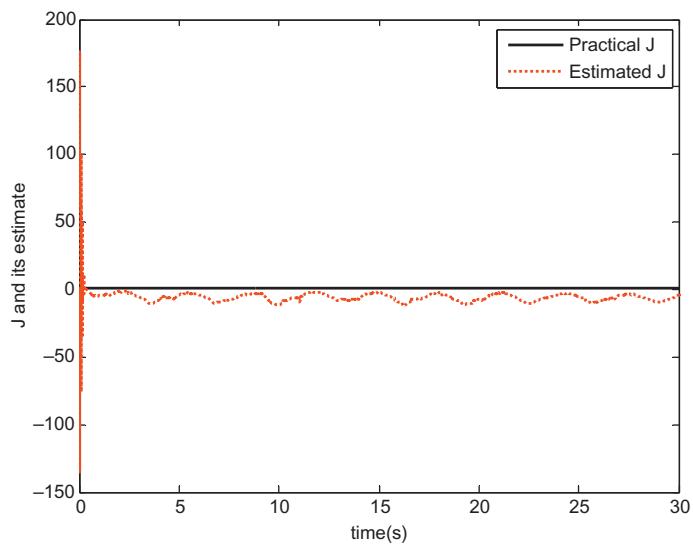
The desired trajectory is selected as $\sin t$, and the limit range of the parameter θ is assumed as $\theta_{\min} = 0.5$ and $\theta_{\max} = 1.5$. Choose $c = 15$, $k_s = 150$, $\varepsilon = 0.05$, $\gamma = 500$ and $\eta = D + 0.01 = 2.01$. Use adaptive sliding mode controller (2.7). If the adaptive law is selected as Eq. (2.12) ($M = 1$), then the simulation results are shown in Figs. 2.1–2.3. If the adaptive law is selected as Eq. (2.13) ($M = 2$), then the simulation results are shown in Figs. 2.4–2.6.



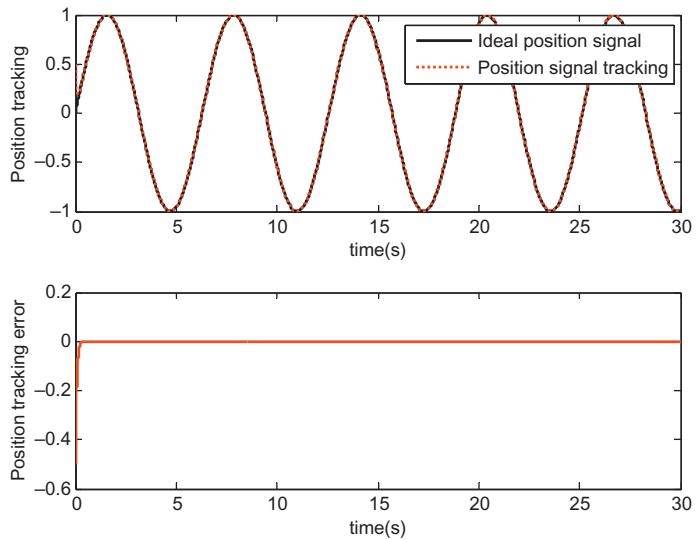
■ FIGURE 2.1 Position tracking based on adaptive law (2.12) ($M = 1$).



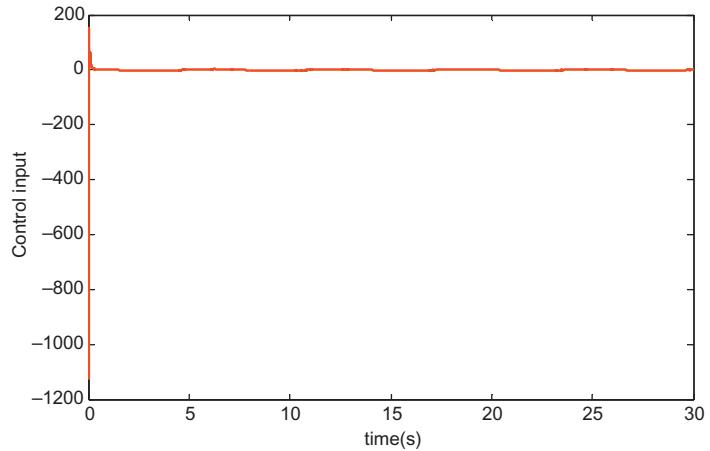
■ FIGURE 2.2 Control input based on adaptive law (2.12) ($M = 1$).



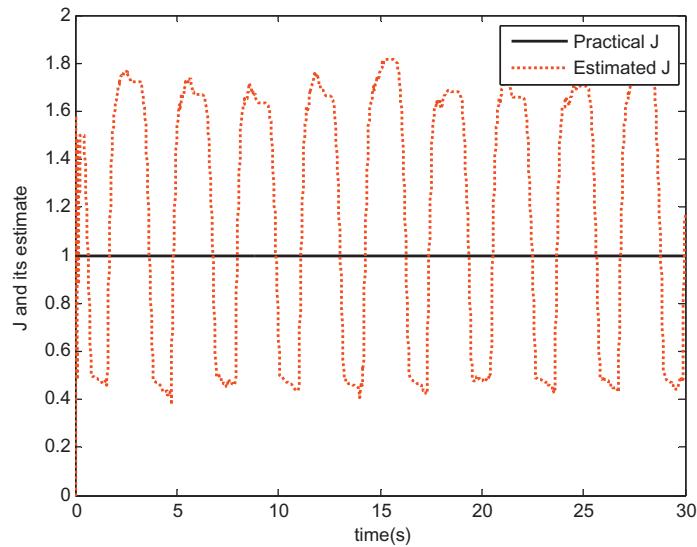
■ FIGURE 2.3 The parameter range based on adaptive law (2.12) ($M = 1$).



■ FIGURE 2.4 Position tracking based on adaptive law (2.13) ($M = 2$).



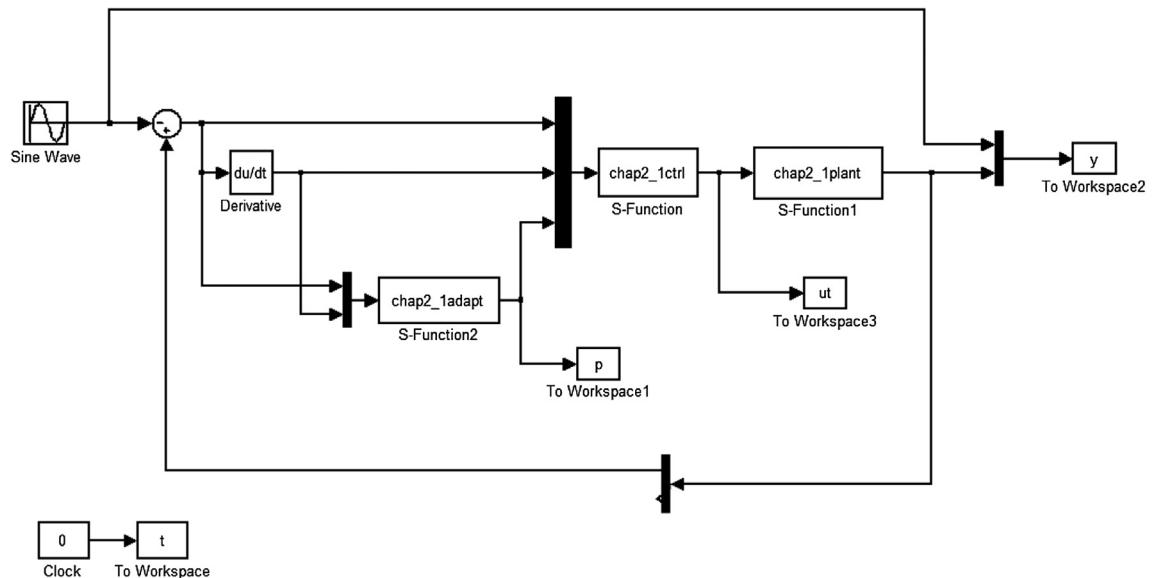
■ FIGURE 2.5 Control input based on adaptive law (2.13) ($M = 2$).



■ FIGURE 2.6 The parameter range based on adaptive law (2.13) ($M = 2$).

Simulation programs:

1. Main Simulink program: chap2_1sim.mdl



2. Control law program: chap2_1ctrl.m

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {1,2, 4, 9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[0 0];
function sys=mdlOutputs(t,x,u)
e=u(1);
de=u(2);
dxd=cos(t);
ddxd=-sin(t);
thp=u(3);

c=15;
s=de+c*e; %Sliding Mode
x2=dxd+de;
dq=ddxd-c*de;

ks=150;
xite=2.01;
epc=0.05;
ua=thp*dq;
us1=-ks*s;
us2=-xite*tanh(s/epc);
ut=ua+us1+us2;
sys(1)=ut;
```

3. Plant S function: chap2_1plant.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.5;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
m=1.0;
ut=u(1);

F=0.5*x(2)+1.5*sign(x(2));
sys(1)=x(2);
sys(2)=1/m*(ut-F);
function sys=mdlOutputs(t,x,u)
m=1.0;

sys(1)=x(1);
sys(2)=m;

```

4. Adaptive law function: chap2_1adapt.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;

```

```
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2, 4, 9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 1;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
e = u(1);
de = u(2);
dxd = cos(t);
ddxd = -sin(t);

x2 = dxd + de;

c = 15;
gama = 500;

s = de + c*e;
thp = x(1);
dq = ddxd - c*de;

th_min = 0.5;
th_max = 1.5;

alaw = -gama*dq*s; %Adaptive law

M = 1;
if M == 1      %Adaptive law
    sys(1) = alaw;
elseif M == 2    %Adaptive law with Proj
```

```

if thp >= th_max&&a1aw > 0
    sys(1)= 0;
elseif thp <= th_min&&a1aw < 0
    sys(1)= 0;
else
    sys(1)= a1aw;
end
end
function sys = mdlOutputs(t,x,u)
sys(1)=x(1); % estimate

```

5. Plot program: chap2_1plot.m

```

close all;

figure(1);
subplot(211);
plot(t,y(:,1),'k',t,y(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('Position tracking');
legend('Ideal position signal','Position signal tracking');
subplot(212);
plot(t,y(:,1)-y(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('Position tracking error');

figure(2);
plot(t,u(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,y(:,3),'k',t,p(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('J and its estimate');
legend('Practical J','Estimated J');

```

2.2 SLIDING MODE CONTROL WITH PRESCRIBED PERFORMANCE

2.2.1 Problem description

Consider a plant as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + g(x)(u + d(t))\end{aligned}\quad (2.14)$$

where x_1 is the position signal, u is the control input, $f(x)$ and $g(x)$ are known functions, $g(x) \neq 0$, $d(t)$ is disturbance and $|d(t)| \leq D$.

Define the ideal signal of x_1 as x_d , and tracking error as $e = x_1 - x_d$, then $\dot{e} = x_2 - \dot{x}_d$, $\ddot{e} = \dot{x}_2 - \ddot{x}_d = f(x) + g(x)(u + d(t)) - \dot{x}_d$.

Define the error performance function as

$$\lambda(t) = (\lambda(0)) - \lambda_\infty \exp(-lt) + \lambda_\infty, \quad (2.15)$$

where $l > 0, 0 < |e(0)| < \lambda(0), \lambda_\infty > 0, \lambda_\infty < \lambda(0)$.

Then $\lambda(t) > 0$ and $\lambda(t)$ tend to λ_∞ exponentially.

2.2.2 Prescribed error performance design

Theorem 2.1: [2–4]: In order to ensure fast convergence of tracking error, and achieve a certain convergence accuracy, tracking error is set as

$$e(t) = \lambda(t)S(\varepsilon). \quad (2.16)$$

Then the prescribed error performance function can be defined as

$$S(\varepsilon) = \frac{e(t)}{\lambda(t)}. \quad (2.17)$$

The function $S(\varepsilon)$ must satisfied as

1. $S(\varepsilon)$ is smooth continuous and monotone increasing function;
2. $-1 < S(\varepsilon) < 1$;
3. $\lim_{\varepsilon \rightarrow +\infty} S(\varepsilon) = 1$ and $\lim_{\varepsilon \rightarrow -\infty} S(\varepsilon) = -1$.

According to the above requirements, $S(\varepsilon)$ can be designed as follows:

$$S(\varepsilon) = \frac{\exp(\varepsilon) - \exp(-\varepsilon)}{\exp(\varepsilon) + \exp(-\varepsilon)}. \quad (2.18)$$

Since $-1 < S(\varepsilon) < 1$, from $\lambda(t)$ definition, we have $-\lambda(t) < \lambda(t)S(\varepsilon) < \lambda(t)$, i.e.,

$$-\lambda(t) < e(t) < \lambda(t).$$

We can then get the tracking error set as

$$\Xi = \{e \in R: |e(t)| < \lambda \forall t \geq 0 \text{ and } |e(t)| < \lambda_\infty \text{ for } t \rightarrow \infty\}. \quad (2.19)$$

2.2.3 Controller design and analysis

According to the properties of the hyperbolic tangent function, the inverse function of $S(\varepsilon)$ is

$$\varepsilon = \frac{1}{2} \ln \frac{1+S}{1-S} = \frac{1}{2} \ln \frac{1+\frac{e}{\lambda}}{1-\frac{e}{\lambda}} = \frac{1}{2} \ln \frac{\lambda+e}{\lambda-e} = \frac{1}{2} (\ln(\lambda+e) - \ln(\lambda-e)).$$

Then

$$\begin{aligned}\dot{\varepsilon} &= \frac{1}{2} \left(\frac{\dot{\lambda} + \dot{e}}{\lambda + e} - \frac{\dot{\lambda} - \dot{e}}{\lambda - e} \right) \\ \ddot{\varepsilon} &= \frac{1}{2} \left(\frac{(\ddot{\lambda} + \ddot{e})(\lambda + e) - (\dot{\lambda} + \dot{e})^2}{(\lambda + e)^2} - \frac{(\ddot{\lambda} - \ddot{e})(\lambda - e) - (\dot{\lambda} - \dot{e})^2}{(\lambda - e)^2} \right) \\ &= \frac{\ddot{\lambda}(\lambda + e) - (\dot{\lambda} + \dot{e})^2}{2(\lambda + e)^2} + \frac{\ddot{e}(\lambda + e)}{2(\lambda + e)^2} - \frac{\ddot{\lambda}(\lambda - e) - (\dot{\lambda} - \dot{e})^2}{2(\lambda - e)^2} + \frac{\ddot{e}(\lambda - e)}{2(\lambda - e)^2} \\ &= \frac{\ddot{\lambda}(\lambda + e) - (\dot{\lambda} + \dot{e})^2}{2(\lambda + e)^2} - \frac{\ddot{\lambda}(\lambda - e) - (\dot{\lambda} - \dot{e})^2}{2(\lambda - e)^2} + \left(\frac{\lambda + e}{2(\lambda + e)^2} + \frac{\lambda - e}{2(\lambda - e)^2} \right) \ddot{e}.\end{aligned}$$

Define $M_1 = \frac{\ddot{\lambda}(\lambda + e) - (\dot{\lambda} + \dot{e})^2}{2(\lambda + e)^2}$, $M_2 = -\frac{\ddot{\lambda}(\lambda - e) - (\dot{\lambda} - \dot{e})^2}{2(\lambda - e)^2}$ and $M_3 = \frac{\lambda + e}{2(\lambda + e)^2} + \frac{\lambda - e}{2(\lambda - e)^2}$, then

$$\ddot{\varepsilon} = M_1 + M_2 + M_3 \ddot{e} = M_1 + M_2 + M_3(f(x) + g(x)(u + d(t)) - \ddot{x}_d).$$

To realize $\varepsilon \rightarrow 0$, design the sliding mode function as $\sigma = \dot{\varepsilon} + c\varepsilon$, $c > 0$, then

$$\dot{\sigma} = \ddot{\varepsilon} + c\dot{\varepsilon} = M_1 + M_2 + M_3(f(x) + g(x)(u + d(t)) - \ddot{x}_d) + c\dot{\varepsilon},$$

then

$$\dot{\sigma} = M_1 + M_2 + M_3 f(x) + u_1 + M_3 g(x)d(t) - M_3 \ddot{x}_d + c\dot{\varepsilon},$$

where $u_1 = M_3 g(x)u$.

Design

$$u_1 = -k\sigma - \eta |M_3 g(x)| \operatorname{sgn}(\sigma) - M_1 - M_2 - M_3 f(x) + M_3 \ddot{x}_d - c\dot{\varepsilon},$$

where $k > 0$, then

$$\dot{\sigma} = -k\sigma - \eta |M_3 g(x)| \operatorname{sgn}(\sigma) + M_3 g(x)d(t).$$

Design the controller as

$$u = \frac{u_1}{M_3 g(x)} \quad (2.20)$$

Define the Lyapunov function as

$$V = \frac{1}{2} \sigma^2.$$

Then

$$\begin{aligned}\dot{V} &= \sigma \dot{\sigma} = \sigma(-k\sigma - \eta |M_3 g(x)| \operatorname{sgn}(\sigma) + M_3 g(x)d(t)) \\ &= -k\sigma^2 - \eta |M_3 g(x)\sigma| + M_3 g(x)d(t)\sigma \leq -k\sigma^2 = -2kV,\end{aligned}$$

where $\eta \geq D$.

From $\dot{V} \leq -2kV$, we can get

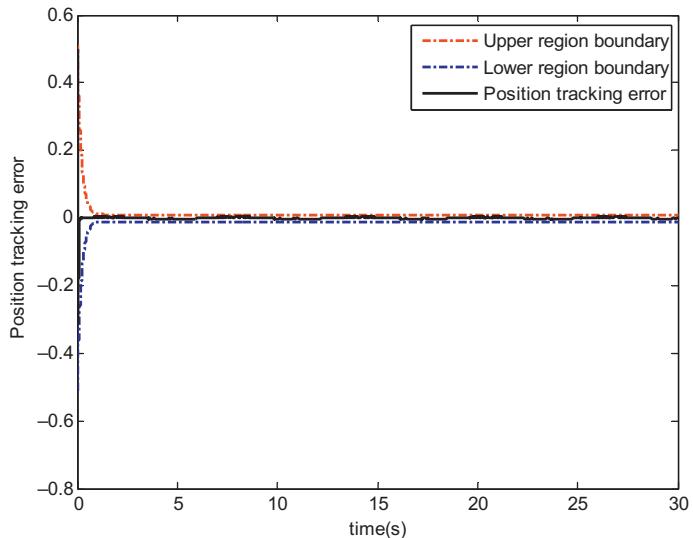
$$V(t) \leq \exp(-2k(t-t_0))V(t_0).$$

Therefore, σ converge to zero exponentially, then ε and $\dot{\varepsilon}$ converge to zero exponentially, $S(\varepsilon)$ is limited and converge to zero, from theorem (2.1), $e(t)$ will converge to zero and is limited to Eq. (2.19).

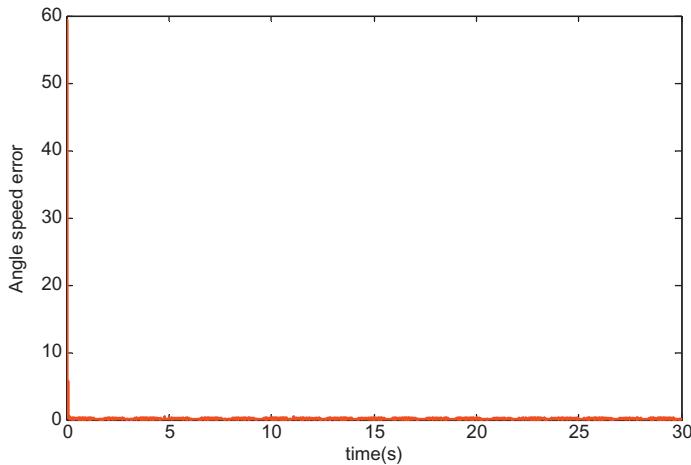
According to properties of the hyperbolic tangent function, we have $\dot{S}(\varepsilon) = (1 - S^2(\varepsilon))\dot{\varepsilon}$, then $\dot{S}(\varepsilon)$ converge to zero exponentially. Since $\dot{\lambda}$ also converge to zero exponentially, and λ converge to λ_∞ exponentially, then $\dot{e}(t) = \dot{\lambda}S + \lambda\dot{S}$ will converge to zero asymptotically.

2.2.4 Simulation example

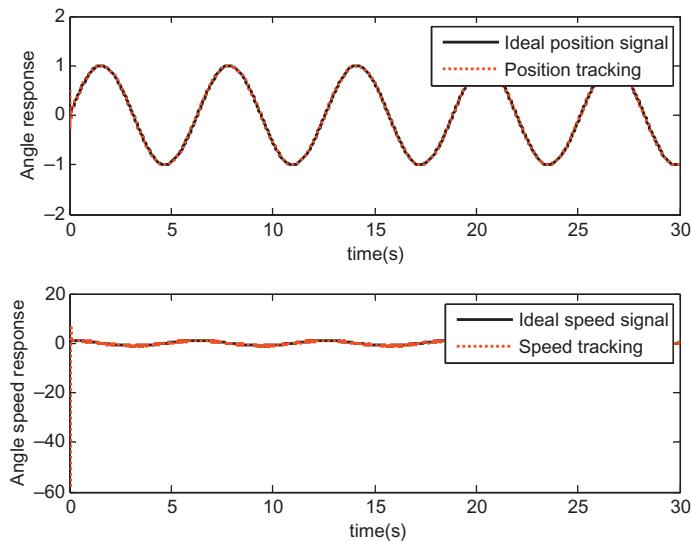
For the plant (2.14), $f(x) = -25x_2$, $g(x) = 133$ and $d(t) = 3\sin t$, and the initial states are $[0.50 \ 0]$, ideal signal is set as $x_d = \sin t$, then $e(0) = x_1(0) - x_d(0) = 0.50$, and prescribed function is set as Eq. (2.18), set $l = 5.0$, $\lambda(0) = 0.51$ and $\lambda_\infty = 0.001$, use controller (2.20), and set $c = 50$, $D = 3.0$, $\eta = |M_3g(x)|D + 0.10$, $k = 10$, the simulation examples are shown in Figs. 2.7–2.10.



■ FIGURE 2.7 Position tracking error.

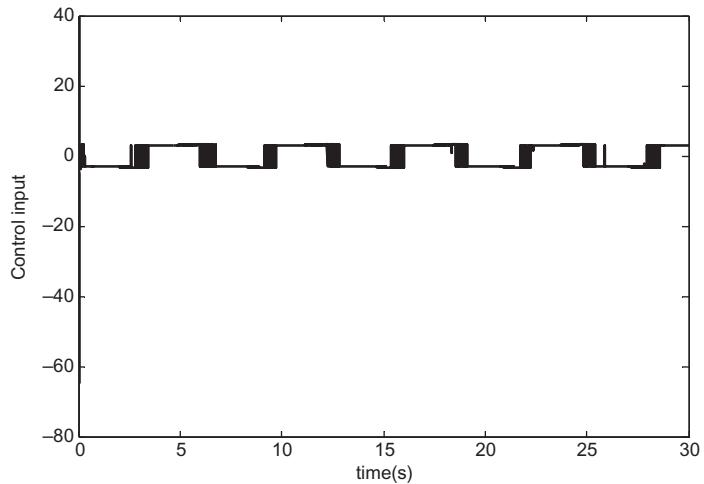


■ FIGURE 2.8 Speed tracking error.



■ FIGURE 2.9 Position and speed tracking.

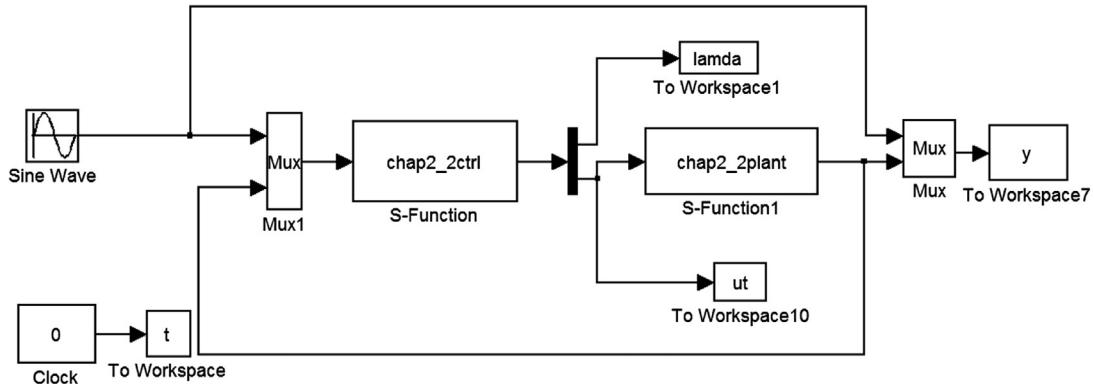
In the simulation, there are two special conditions that $\lambda(t)$ is near to $e(t)$ as follows: (1) when $t = 0$, $\lambda(0)$ is near to $e(0)$; (2) when $t \rightarrow \infty$, if λ_∞ is very small, $e(t)$ will be near to λ_∞ . Under the two conditions above, according to $S(\varepsilon) = \frac{e(t)}{\lambda(t)}$, $S(\varepsilon)$ will be near to 1.0, and $\varepsilon = \frac{1}{2} \ln \frac{1+S}{1-S}$ will be singular. To avoid the singularity of ε , $\lambda(0)$ cannot be chosen to be closed to $e(0)$, and λ_∞ cannot be chosen to be very small.



■ FIGURE 2.10 Control input.

Simulation programs:

1. Main Simulink program: chap2_2sim.mdl



2. Control law program: chap2_2ctrl.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
```

```
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];
function sys=mdlOutputs(t,x,u)
xd=u(1);
dxd=cos(t);
ddxd=-sin(t);
x1=u(2);
x2=u(3);
fx = -25*x2;
gx = 133;

e=x1-xd;
de=x2-dxd;

l=5;
M=2;
if M==1 %Fixed Step is
0.0001 in Simulink
    lamda0=0.5001; %>abs(e(0))=0.50
    lamda_inf=0.0001;
elseif M==2 %Fixed Step is 0.001 in Simulink
    lamda0=0.51; %>abs(e(0))=0.50
    lamda_inf=0.01;
end

lamda=(lamda0-lamda_i2nf)*exp(-l*t)+lamda_inf;
dlamda=-l*(lamda0-lamda_inf)*exp(-l*t);
ddlamda=l^2*(lamda0-lamda_inf)*exp(-l*t);
```

```

S = e/lamda;

epc = 0.5*log((1+S)/(1-S)); %To guarantee log effective,
must use the suitable solver method in simulink

depC = (de*lamda - e*dlamda)/((lamda + e)*lamda);

D = 3.0;
c = 50;
k = 10;

E = c*epc + depC;

M1=(ddlamda*(lamda + e)-(dlamda+de)^2)/(2*(lamda + e)^2);
M2=-(ddlamda*(lamda - e)-(dlamda - de)^2)/(2*(lamda - e)^2);
M3=(lamda + e)/(2*(lamda + e)^2)+(lamda - e)/(2*(lamda - e)^2);

xite = abs(M3*gx)*D + 0.10;

delta = 0.020;
kk = 1/delta;
if abs(E)>delta
    satE=sign(E);
else
    satE=kk*E;
end
u1 = - k*E - xite*satE - M1 - M2 - M3*fx + M3*ddxd - c*depC;
ut = u1/(M3*gx);

sys(1)=lamda;
sys(2)=ut;

```

3. Plant S function: chap2_2plant.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes

```

```

sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.50 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
sys(1)=x(2);
sys(2)=-25*x(2)+133*(ut+3*sin(t));
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
4. Plot program: chap2_2plot.m
close all;

figure(1);
plot(t,lambda,'-.r',t,-lambda,'-.b',t,y(:,2)-y(:,1),'k','linewidth',2);
legend('Upper region boundary','Lower region boundary','Position tracking error');
xlabel('time(s)'),ylabel('Position tracking error');

figure(2);
plot(t,abs(cos(t)-y(:,3)), 'r','linewidth',2);
xlabel('time(s)'),ylabel('Angle speed error');

figure(3);
subplot(211);
plot(t,y(:,1), 'k',t,y(:,2), 'r','linewidth',2);
legend('Ideal position signal','Position tracking');
xlabel('time(s)'),ylabel('Angle response');
subplot(212);
plot(t,cos(t), 'k',t,y(:,3), 'r','linewidth',2);
legend('Ideal speed signal','Speed tracking');
xlabel('time(s)'),ylabel('Angle speed response');

figure(4);
plot(t,ut(:,1), 'k','linewidth',2);
xlabel('time(s)'),ylabel('Control input');

```

REFERENCES

- [1] L. Xu, B. Yao, Adaptive robust control of mechanical systems with non-linear dynamic friction compensation, *Int. J. Control.* 81 (2) (2008) 167–176.
- [2] C.P. Bechlioulis, G.A. Rovithakis, Robust adaptive control of feedback linearizable MIMO nonlinear systems with prescribed performance, *IEEE Trans. Autom. Control* 53 (9) (2008) 2090–2099.
- [3] C.P. Bechlioulis, G.A. Rovithakis, Adaptive control with guaranteed transient and steady state tracking error bounds for strict feedback systems, *Automatica* 45 (2) (2009) 532–538.
- [4] C.P. Bechlioulis, G.A. Rovithakis, Prescribed performance adaptive control for multi-input multi-output affine in the control nonlinear systems, *IEEE Trans. Autom. Control* 55 (5) (2010) 1220–1226.
- [5] M. Krstic, I. Kanellakopoulos, P. Kokotovic, *Nonlinear and Adaptive Control Design*, John Wiley & Sons, INC, New York, 1995.

FURTHER READING

- J. Liu, X. Wang, *Advanced Sliding Mode Control for Mechanical Systems: Design, Analysis and Matlab Simulation*, Tsinghua & Springer Press, 2011.

Sliding mode control based on a state observer

In practical engineering, a speed signal is often difficult to measure. In this chapter, several kinds of state observer are introduced, and based on the observer, the sliding mode controller can be designed.

In [section 3.1](#), based on an exponential high gain observer, a sliding mode controller is designed, closed system exponential convergence is analyzed and an example is given.

In [section 3.2](#), a kind of low gain observer (a K observer) is designed to estimate a speed and acceleration signal, sliding mode control only by using position signal is realized, closed system exponential convergence is analyzed and an example is given.

In [section 3.3](#), it is shown how a high gain differentiator can be used to calculate the derivative value of any signal without model information, which is valuable for practical engineering, especially for a high order system. In this section, a high gain differentiator is used to estimate a speed and acceleration signal and realize sliding mode control only by using position signal, exponential convergence is realized and an example is given.

In [section 3.4](#), for a flexible-joint robot system, a robust observer is designed as a dynamic observer and a state estimation to estimate a speed and acceleration signal, sliding mode control only by using angle signal is realized, closed system exponential convergence is analyzed and an example is given.

In [section 3.5](#), based on [section 3.4](#), a sliding mode control for a flexible-joint robot system is designed, a speed signal is not needed, closed system exponential convergence is analyzed and an example is given.

In [section 3.6](#), the separation theorem is used for a single one link inverted pendulum, a high gain observer is designed and can be used in

the closed system directly, closed system stability analysis is not needed, and the closed loop control system is asymptotically stable.

3.1 SLIDING MODE CONTROL BASED ON A HIGH GAIN OBSERVER

3.1.1 System description

Consider a plant as

$$G(s) = \frac{k}{s^2 + as + b}. \quad (3.1)$$

Eq. (3.1) can be expressed as

$$\ddot{\theta} = -a\dot{\theta} - b\theta + ku(t), \quad (3.2)$$

where $\theta(t)$ is position signal and $u(t)$ is control input.

Defining $x_1 = \theta$, $x_2 = \dot{\theta}$, Eq. (3.2) can be written as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -ax_2 - bx_1 + ku(t). \\ y &= x_1 \end{aligned} \quad (3.3)$$

The control task is to design sliding mode controller without a speed signal.

3.1.2 High gain observer design

Design a high gain observer as

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 + \frac{\alpha_1}{\varepsilon}(y - \hat{x}_1) \\ \dot{\hat{x}}_2 &= -ax_2 - bx_1 + ku + \frac{\alpha_2}{\varepsilon^2}(y - \hat{x}_1), \end{aligned}$$

where α_1 and α_2 are positive values, and $\varepsilon \ll 1$.

Choosing $h_1 = \frac{\alpha_1}{\varepsilon}$, $h_2 = \frac{\alpha_2}{\varepsilon^2}$, we then have

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 + h_1(y - \hat{x}_1) \\ \dot{\hat{x}}_2 &= -a\hat{x}_2 - bx_1 + ku(t) + h_2(y - \hat{x}_1), \end{aligned} \quad (3.4)$$

where $\tilde{x} = x - \hat{x}$.

From Eqs. (3.3) and (3.4), we have

$$\begin{aligned} \dot{\tilde{x}}_1 &= -h_1\tilde{x}_1 + \tilde{x}_2 \\ \dot{\tilde{x}}_2 &= -h_2\tilde{x}_1 - a\tilde{x}_2, \\ y &= x_1 \end{aligned}$$

i.e.,

$$\dot{\tilde{x}} = A\tilde{x}, \quad (3.5)$$

$$\text{where } A = \begin{bmatrix} -h_1 & 1 \\ -h_2 & -a \end{bmatrix}, \tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}.$$

If A is Hurwitz, then \tilde{x} will converge to zero exponentially, and can be expressed as

$$||\tilde{x}(t)|| \leq \varphi_0 ||\tilde{x}(t_0)|| e^{-\sigma_0(t-t_0)}, \quad (3.6)$$

where φ_0 and σ_0 are positive constants.

[Eq. \(3.6\)](#) indicates that the convergence precision of $||\tilde{x}(t)||$ is related to σ_0 . Consider σ_0 is related to the minimum eigenvalue of A ; the smaller the ε value is, the bigger the h_i value is, and then the smaller the minimum eigenvalue of A is, the bigger the σ_0 value is. Therefore, the convergence of $||\tilde{x}(t)||$ depends on ε , and a high gain observer will improve convergence precision of $||\tilde{x}(t)||$ greatly.

The characteristic equation of matrix A is $|sI - A| = \begin{vmatrix} s + h_1 & -1 \\ h_2 & s + a \end{vmatrix} = 0$, i.e., $s^2 + (h_1 + a)s + h_2 = 0$, from $(s+p)^2 = 0$, we have $s^2 + 2ps + p^2 = 0$, then

$$\begin{cases} h_1 + a = 2p \\ h_2 = p^2 \end{cases}, \quad (3.7)$$

where $h_1 = 2p - a > 0$.

From $(s+p)^2 = 0$, we know that if we design $p > 0$, A is Hurwitz. For example, if we choose $p = 100$, then $h_1 = 200 - a$, $h_2 = 10,000$.

From [Eq. \(3.4\)](#), we define $\tilde{x}_2 = x_2 - \hat{x}_2$, then

$$\dot{\tilde{x}}_2 = \dot{x}_2 - \dot{\hat{x}}_2 = -ax_2 - bx_1 + ku - (-a\hat{x}_2 - bx_1 + ku + h_2(y - \hat{x}_1)) = -a\tilde{x}_2 - h_2\tilde{x}_1.$$

3.1.3 Sliding mode controller design

For system [\(3.1\)](#), we design the sliding mode function as

$$s = ce + \dot{e},$$

where $c > 0$.

Define

$$\hat{s} = c\hat{e} + \dot{\hat{e}}, \quad (3.8)$$

where $\hat{e} = \theta_d - \hat{\theta}$ and $\dot{\hat{e}} = \dot{\theta}_d - \dot{\hat{\theta}}$.

The design sliding mode controller is

$$u(t) = \frac{1}{k}(\ddot{\theta}_d + a\hat{\theta} + b\theta + \eta\hat{s} + c\dot{e}), \quad (3.9)$$

where $\eta > 0$, $\hat{e} = \theta_d - \hat{\theta}$ and $\hat{s} = c\hat{e} + \dot{e}$.

The Lyapunov function for the controller is designed as

$$V_s = \frac{1}{2}s^2.$$

Since

$$\begin{aligned}\ddot{e} &= \ddot{\theta}_d - \ddot{\theta} = \ddot{\theta}_d + a\dot{\theta} + b\theta - ku \\ \dot{s} &= c\dot{e} + \ddot{e} = c\dot{e} + \ddot{\theta}_d + a\dot{\theta} + b\theta - ku,\end{aligned}$$

then

$$\begin{aligned}\dot{s} &= c\dot{e} + \ddot{\theta}_d + a\dot{\theta} + b\theta - (\ddot{\theta}_d + a\dot{\theta} + b\theta + \eta\hat{s} + c\dot{e}) \\ &= c\tilde{e} + a\tilde{\theta} - \eta\hat{s} = -\eta s + \eta\tilde{s} + c\tilde{e} + a\tilde{\theta} \\ &= -\eta s + \eta(-c\tilde{\theta} - \tilde{\theta}) + c(-\tilde{\theta}) + a\tilde{\theta} \\ &= -\eta s + \eta(-c\tilde{\theta} - \tilde{\theta}) + c(-\tilde{\theta}) + a\tilde{\theta} \\ &= -\eta s - \eta c\tilde{\theta} + (a - \eta - c)\tilde{\theta}\end{aligned},$$

where $\tilde{\theta} = \theta - \hat{\theta}$, $\tilde{\theta} = \dot{\theta} - \hat{\theta}$, $\tilde{e} = e - \hat{e} = -\theta + \hat{\theta} = -\tilde{\theta}$, $\tilde{s} = -\tilde{\theta}$ and $\tilde{s} = s - \hat{s} = c\tilde{e} + \tilde{e} = -c\tilde{\theta} - \tilde{\theta}$.

Then

$$\dot{V}_s = -\eta s^2 + s(-\eta c\tilde{\theta} + (a - \eta - c)\tilde{\theta}) = -\eta s^2 + k_1 s\tilde{\theta} + k_2 s\tilde{\theta},$$

where $k_1 = -\eta c$ and $k_2 = a - \eta - c$.

Since $k_1 s\tilde{\theta} \leq \frac{1}{2}s^2 + \frac{1}{2}k_1^2\tilde{\theta}^2$ and $k_2 s\tilde{\theta} \leq \frac{1}{2}s^2 + \frac{1}{2}k_2^2\tilde{\theta}^2$, then

$$\dot{V}_s \leq -\eta s^2 + \frac{1}{2}s^2 + \frac{1}{2}k_1^2\tilde{\theta}^2 + \frac{1}{2}s^2 + \frac{1}{2}k_2^2\tilde{\theta}^2 = -(\eta - 1)s^2 + \frac{1}{2}k_1^2\tilde{\theta}^2 + \frac{1}{2}k_2^2\tilde{\theta}^2,$$

where $\eta > 1$.

Different from the book [1], the total Lyapunov function for the closed system is improved and designed as

$$V = V_s + V_o, \quad (3.10)$$

where $V_o = \frac{1}{2}\tilde{x}^T\tilde{x} = \frac{1}{2}\tilde{\theta}^2 + \frac{1}{2}\tilde{\theta}^2$.

Since $\dot{V}_o = \tilde{x}^T\dot{\tilde{x}} = \tilde{x}^TA\tilde{x}$, from section 3.1.2 analysis, \dot{V}_o converges to zero exponentially, i.e.,

$$\tilde{x}^TA\tilde{x} \leq \chi(\bullet)e^{-\sigma_0(t-t_0)}, \quad (3.11)$$

where $\chi(\bullet)$ is a K-class function of $\|\tilde{x}(t_0)\|$.

Then

$$\begin{aligned}\dot{V} &\leq -(\eta - 1)s^2 + \frac{1}{2}k_1^2\tilde{\theta}^2 + \frac{1}{2}k_2^2\tilde{\theta}^2 + \tilde{x}^T A \tilde{x} \\ &= -\eta_1 V_s - \frac{1}{2}\eta_1 \tilde{\theta}^2 - \frac{1}{2}\eta_1 \tilde{\theta}^2 + \frac{1}{2}(k_1^2 + \eta_1)\tilde{\theta}^2 + \frac{1}{2}(k_2^2 + \eta_1)\tilde{\theta}^2 + \tilde{x}^T A \tilde{x}, \\ &\leq -\eta_1 V + \chi(\bullet)e^{-\sigma_0(t-t_0)}\end{aligned}$$

where $\eta_1 = 2(\eta - 1) > 0$, $\sigma_0 > 0$, $x = [\theta \quad \dot{\theta}]^T$.

Using Lemma 1.3, the solution of $\dot{V} \leq -\eta_1 V + \chi(\bullet)e^{-\sigma_0(t-t_0)}$ is

$$\begin{aligned}V(t) &\leq e^{-\eta_1(t-t_0)}V(t_0) + \chi(\bullet) \int_{t_0}^t e^{-\eta_1(t-\tau)}e^{-\sigma_0(\tau-t_0)}d\tau \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \chi(\bullet)e^{-\eta_1 t + \sigma_0 t_0} \int_{t_0}^t e^{\eta_1 \tau}e^{-\sigma_0 \tau}d\tau \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0}e^{-\eta_1 t + \sigma_0 t_0}e^{(\eta_1 - \sigma_0)\tau}|_{t_0}^t \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0}e^{-\eta_1 t + \sigma_0 t_0}(e^{(\eta_1 - \sigma_0)t} - e^{(\eta_1 - \sigma_0)t_0}) \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0}(e^{-\sigma_0(t-t_0)} - e^{-\eta_1(t-t_0)})\end{aligned}$$

i.e.,

$$\lim_{t \rightarrow \infty} V(t) \leq 0$$

Since $V(t) \geq 0$, then we have $t \rightarrow \infty$, $V(t) = 0$ and $V(t)$ to zero exponentially, the convergence precision depends on η_1 , i.e., η .

3.1.4 Simulation example

Consider a plant as

$$G(s) = \frac{1}{s^2 + 10s + 1}$$

i.e.,

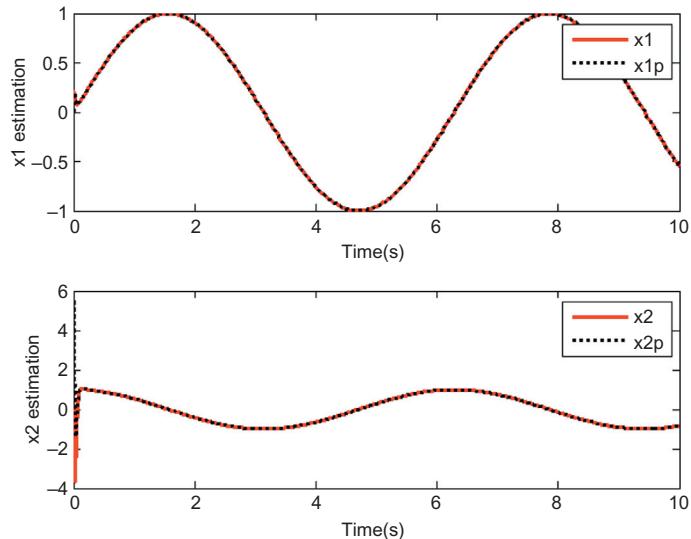
$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -10x_2 - x_1 + u(t)\end{aligned}$$

A high gain observer is designed as

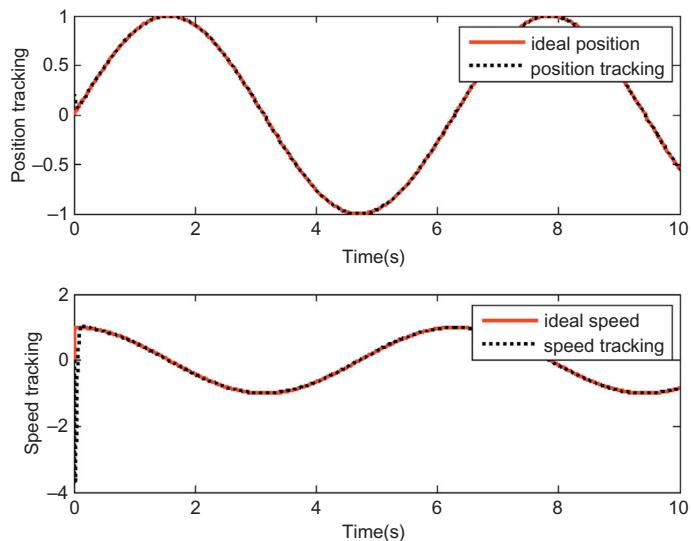
$$\begin{aligned}\dot{\hat{x}}_1 &= \hat{x}_2 + h_1(y - \hat{x}_1) \\ \dot{\hat{x}}_2 &= -10\hat{x}_2 - x_1 + u + h_2(y - \hat{x}_1)\end{aligned}$$

The initial states are $x_1(0) = 0.20$, $x_2(0) = 0$, and ideal position signal is set as $\theta_d = \sin t$.

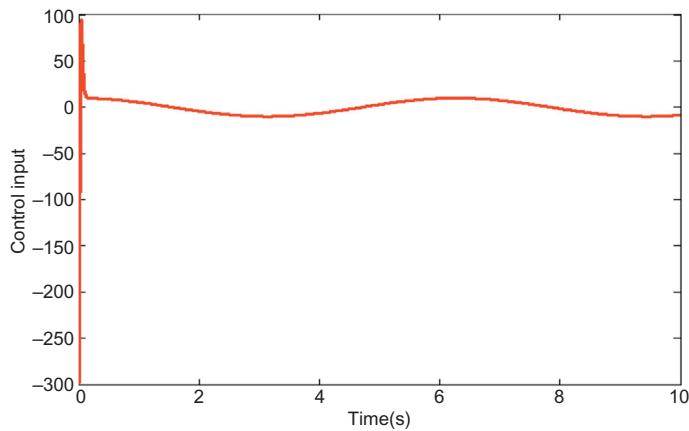
Using controller (3.9) with observer (3.4), and choosing $\varepsilon = 0.10$, $\alpha_1 = \alpha_2 = 1$, $c = 5$, $\eta = 1.5$, the simulation results are given in Figs. 3.1–3.3.



■ FIGURE 3.1 Position and speed estimation.



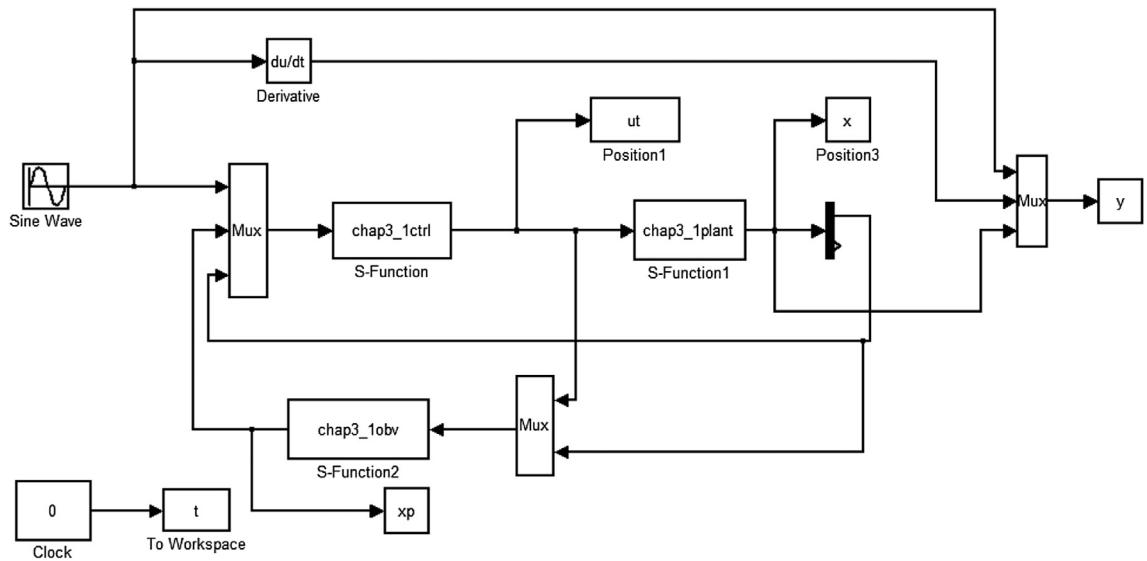
■ FIGURE 3.2 Position and speed tracking.



■ FIGURE 3.3 Control input.

Simulation programs:

1. Main Simulink: chap3_1sim.mdl



2. S function for controller: chap3_1ctrl.m

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {1,2, 4, 9 }
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[ -1 0];
function sys=mdlOutputs(t,x,u)
thd=sin(t);
wd=cos(t);
ddthd= - sin(t);

thp=u(2);
wp=u(3);
th=u(4);

elp=thd-thp;
e2p=wd-wp;

k=1;a=10;b=1;
c=10;
xite=50;
sp=c*elp+e2p;
ut=1/k*(ddthd + a*wp + b*th + xite*sp + c*e2p);

sys(1)=ut;
```

3. S function for a high gain observer: chap3_lobv.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
y=u(2);
a=10;
h1=200-a;
h2=10000;

sys(1)=x(2)+h1*(y-x(1));
sys(2)=-10*x(2)-y+ut+h2*(y-x(1));
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);

```

4. S function for plant: chap3_1plant.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,

```

```

case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0=[0.2 0];
str=[];
ts=[-1 0];
function sys=mdlDerivatives(t,x,u)
sys(1)=x(2);
sys(2)=-10*x(2)-x(1)+u(1);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);

```

5. Plot program: chap3_1plot.m

```

close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r',t,xp(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('x1 estimation');
legend('x1','x1p');
subplot(212);
plot(t,x(:,2),'r',t,xp(:,2),'k','linewidth',2);
xlabel('time(s)');ylabel('x2 estimation');
legend('x2','x2p');

```

```

figure(2);
subplot(211);
plot(t,y(:,1),'r',t,y(:,3),'k','linewidth',2);
xlabel('time(s)');ylabel('Position tracking');
legend('ideal position','position tracking');
subplot(212);
plot(t,y(:,2),'r',t,y(:,4),'k','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking');
legend('ideal speed','speed tracking');

figure(3);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input');

```

3.2 SLIDING MODE CONTROL BASED ON THE K OBSERVER FOR A HIGH ORDER SYSTEM

The K observer is a kind of low gain observer [2], which is valuable for practical engineering, especially for a high order system. In this section, we use the k observer to estimate the speed and acceleration signal, and realize sliding mode control only by using the position signal.

3.2.1 K observer design and analysis

Consider a plant as

$$G(s) = \frac{K}{(Ts+1)^3}, \quad (3.12)$$

where K is gain and T is time constant.

Eq. (3.12) can also be expressed as

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}, \quad (3.13)$$

where $\mathbf{x} = [x_1 \ x_2 \ x_3]^T$, $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{T^3} & -\frac{0}{T^2} & -\frac{1}{T} \end{bmatrix}$,

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & \frac{K}{T^3} \end{bmatrix}^T.$$

Eq. (3.13) can be written as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= -\frac{1}{T^3}x_1 - \frac{3}{T^2}x_2 - \frac{3}{T}x_3 + bu, \\ y &= \mathbf{c}^T \mathbf{x} = x_1\end{aligned}\quad (3.14)$$

where $b = \frac{K}{T^3}$, $\mathbf{c} = [1 \ 0 \ 0]^T$.

We assume only $y = x_1$ can be measured, $\mathbf{k} = [k_1 \ k_2 \ k_3]^T$ is designed to satisfy $A_0 = A - \mathbf{k}\mathbf{c}^T$ to be Hurwitz.

K observer is designed as [2]

$$\begin{aligned}\dot{\omega} &= A_0\omega + kx_1 \\ \dot{v} &= A_0v + e_3u,\end{aligned}\quad (3.15)$$

where ω and v are states vector, $e_3 = [0 \ 0 \ 1]^T$, and

$$\begin{aligned}A_0 &= A - \mathbf{k}\mathbf{c}^T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{T^3} & -\frac{3}{T^2} & -\frac{3}{T} \end{bmatrix} - \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{T^3} & -\frac{3}{T^2} & -\frac{3}{T} \end{bmatrix} - \begin{bmatrix} k_1 & 0 & 0 \\ k_2 & 0 & 0 \\ k_3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -k_1 & 1 & 0 \\ -k_2 & 0 & 1 \\ -\frac{1}{T^3} - k_3 & -\frac{3}{T^2} & -\frac{3}{T} \end{bmatrix}.\end{aligned}$$

Define

$$\hat{x} = \omega + bv. \quad (3.16)$$

Define $\tilde{x} = x - \hat{x}$, then

$$\dot{\tilde{x}} = \dot{x} - \dot{\hat{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u - \dot{\omega} - b\dot{v}$$

and

$$\begin{aligned}\dot{\tilde{x}} &= \dot{x} - \dot{\hat{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u - \dot{\omega} - b\dot{v} \\ &= (A_0 + \mathbf{k}\mathbf{c}^T)\mathbf{x} + \mathbf{B}u - (A_0\omega + kx_1) - b(A_0v + e_3u) \\ &= \mathbf{A}_0\mathbf{x} - \mathbf{A}_0(\omega + bv) \\ &= \mathbf{A}_0(x - \hat{x}) = \mathbf{A}_0\tilde{x}\end{aligned}\quad (3.17)$$

The Lyapunov function for the observer is $V_o = \frac{1}{2}\tilde{x}^T\tilde{x}$, then

$$\dot{V}_o = \tilde{x}^T\dot{\tilde{x}} = \mathbf{A}_0\tilde{x}^T\tilde{x} \leq 2\mathbf{A}_0V_o$$

Consider $A_0 = A - \mathbf{k}\mathbf{c}^T$ is Hurwitz, then \tilde{x} will tend to zero exponentially.

From $\dot{\tilde{x}} = A_0 \tilde{x}$, we can obtain

$$||\tilde{x}(t)|| \leq \varphi_0 ||\tilde{x}(t_0)|| e^{-\sigma_0(t-t_0)}, \quad (3.18)$$

where φ_0 and σ_0 are positive constants.

3.2.2 k Design

The characteristic equation of $\dot{\tilde{x}} = A_0 \tilde{x}$ is

$$|\lambda I - A_0| = \begin{vmatrix} \lambda + k_1 & -1 & 0 \\ k_2 & \lambda & -1 \\ \frac{1}{T^3} + k_3 & \frac{3}{T^2} & \lambda + \frac{3}{T} \end{vmatrix} = 0,$$

then

$$\lambda^3 + \left(\frac{3}{T} + k_1\right)\lambda^2 + \left(\frac{3k_1}{T} + \frac{3}{T^2} + k_2\right)\lambda + \frac{1}{T^3} + k_3 + \frac{3}{T^2}k_1 + \frac{3}{T}k_2 = 0.$$

For example, if we choose $\lambda = -1$, i.e., $(\lambda+1)^3 = 0$ and $\lambda^3 + 3\lambda^2 + 3\lambda + 1 = 0$, then we have

$$\begin{cases} \frac{3}{T} + k_1 = 3 \\ \frac{3k_1}{T} + \frac{3}{T^2} + k_2 = 3 \\ \frac{1}{T^3} + k_3 + \frac{3}{T^2}k_1 + \frac{3}{T}k_2 = 1 \end{cases}.$$

We can then obtain

$$\begin{cases} k_1 = 3 - \frac{3}{T} \\ k_2 = 3 - \frac{3k_1}{T} - \frac{3}{T^2} \\ k_3 = 1 - \frac{1}{T^3} - \frac{3}{T^2}k_1 - \frac{3}{T}k_2 \end{cases}. \quad (3.19)$$

3.2.3 Sliding mode control based on a K observer

The control object is that x_1 tracking x_d , x_2 tracking \dot{x}_d by using x_1 only.

Define $e = x_1 - x_{1d}$, and design sliding mode function as

$$s = c_1 e + c_2 \dot{e} + \ddot{e},$$

where c_1 and c_2 must designed so that $c_1 + c_2\sigma + \sigma^2 = 0$ is Hurwitz.

Define $\alpha(\mathbf{x}) = -\frac{1}{T^3}x_1 - \frac{3}{T^2}x_2 - \frac{3}{T}x_3$, and design sliding mode controller as

$$u(t) = \frac{1}{b}(-c_1\hat{e} - c_2\hat{e} - \alpha(\hat{\mathbf{x}}) - \eta\hat{s} + \ddot{x}_{1d}), \quad (3.20)$$

where $\hat{e} = \hat{x}_1 - x_{1d}$, $\hat{s} = c_1\hat{e} + c_2\hat{e} + \ddot{e}$, $\alpha(\hat{\mathbf{x}}) = -\frac{1}{T^3}\hat{x}_1 - \frac{3}{T^2}\hat{x}_2 - \frac{3}{T}\hat{x}_3$.

Then

$$\begin{aligned} \dot{s} &= c_1\dot{e} + c_2\ddot{e} + \ddot{e} = c_1\dot{e} + c_2\ddot{e} + \dot{x}_3 - \ddot{x}_{1d} \\ &= c_1\dot{e} + c_2\ddot{e} + \alpha(\mathbf{x}) + bu - \ddot{x}_{1d} \\ &= c_1\dot{e} + c_2\ddot{e} + \alpha(\mathbf{x}) - c_1\hat{e} - c_2\hat{e} - \alpha(\hat{\mathbf{x}}) - \eta\hat{s} + \ddot{x}_{1d} - \ddot{x}_{1d} \\ &= -\eta\hat{s} + v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}}) \end{aligned}$$

where $v(\tilde{\mathbf{x}}) = c_1\dot{e} + c_2\ddot{e} - c_1\hat{e} - c_2\hat{e}$.

Design the Lyapunov function as $V = \frac{1}{2}s^2$, then

$$\begin{aligned} \dot{V} &= s\dot{s} = -\eta s\hat{s} + s(v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})) \\ &= -\eta s(s - \tilde{s}) + s(v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})) \\ &= -\eta s^2 + s(\eta\tilde{s} + v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})) \\ &= -\eta s^2 + sf(\tilde{\mathbf{x}}) \leq -\eta s^2 + \frac{1}{2}(s^2 + f(\tilde{\mathbf{x}})^2) \\ &= -(\eta - 0.5)s^2 + 0.5f(\tilde{\mathbf{x}})^2 = -\eta_1 V + 0.5f(\tilde{\mathbf{x}})^2 \end{aligned},$$

where $\eta_1 = 2\eta - 1 > 0$, $\tilde{s} = s - \hat{s}$, $f(\tilde{\mathbf{x}}) = \eta\tilde{s} + v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})$.

Since $e - \hat{e} = \tilde{x}_1$, $\dot{e} - \hat{e} = \tilde{x}_2$, $\ddot{e} - \hat{e} = \tilde{x}_3$, then

$$\begin{aligned} \tilde{s} &= s - \hat{s} = c_1\tilde{x}_1 + c_2\tilde{x}_2 + \tilde{x}_3 \\ v(\tilde{\mathbf{x}}) &= c_1\dot{e} + c_2\ddot{e} - c_1\hat{e} - c_2\hat{e} = c_1\tilde{x}_2 + c_2\tilde{x}_3 \end{aligned}$$

$$\begin{aligned} \alpha(x) - \alpha(\hat{x}) &= -\frac{1}{T^3}x_1 - \frac{3}{T^2}x_2 - \frac{3}{T}x_3 + \frac{1}{T^3}\hat{x}_1 + \frac{3}{T^2}\hat{x}_2 + \frac{3}{T}\hat{x}_3 \\ &= -\frac{1}{T^3}\tilde{x}_1 - \frac{3}{T^2}\tilde{x}_2 - \frac{3}{T}\tilde{x}_3 \end{aligned}$$

and

$$\begin{aligned} f(\tilde{\mathbf{x}}) &= \eta\tilde{s} + v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}}) = \left(\eta c_1 - \frac{1}{T^3}\right)\tilde{x}_1 \\ &\quad + \left(\eta c_2 - \frac{3}{T^2} + c_1\right)\tilde{x}_2 + \left(\eta - \frac{3}{T} + c_2\right)\tilde{x}_3. \end{aligned}$$

From Eq. (3.18), we can obtain

$$\dot{V} \leq -\eta_1 V + 0.5f(\tilde{\mathbf{x}})^2 \leq -\eta_1 V + \chi(\bullet)\exp(-\sigma_0(t - t_0)),$$

where $\chi(\bullet)$ is a K-class function of $\|\tilde{\mathbf{x}}(t_0)\|$.

Using Lemma 1.3, the solution of $\dot{V} \leq -\eta_1 V + \chi(\bullet) \exp(-\sigma_0(t-t_0))$ is

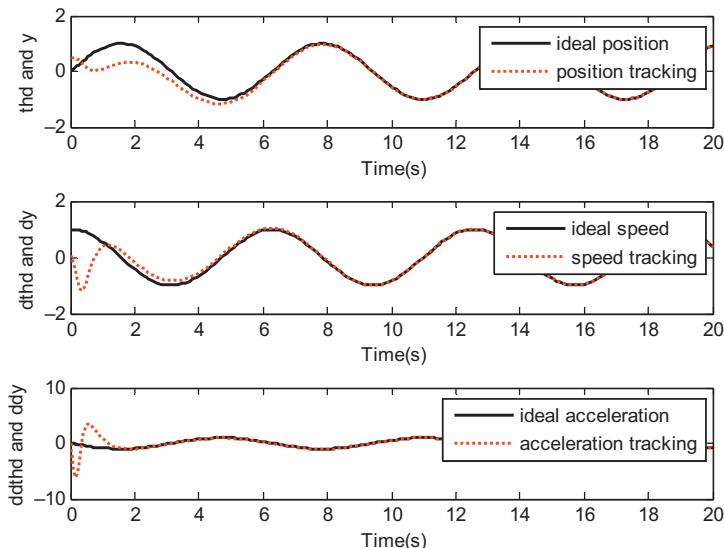
$$\begin{aligned}
 V(t) &\leq \exp(-\eta_1(t-t_0))V(t_0) + \chi(\bullet) \int_{t_0}^t \exp(-\eta_1(t-\tau)) \exp(-\sigma_0(\tau-t_0)) d\tau \\
 &= \exp(-\eta_1(t-t_0))V(t_0) + \chi(\bullet) \exp(-\eta_1 t + \sigma_0 t_0) \int_{t_0}^t \exp(\eta_1 \tau) \exp(-\sigma_0 \tau) d\tau \\
 &= \exp(-\eta_1(t-t_0))V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} \exp(-\eta_1 t + \sigma_0 t_0) \exp((\eta_1 - \sigma_0)\tau) \Big|_{t_0}^t \\
 &= \exp(-\eta_1(t-t_0))V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} \exp(-\eta_1 t + \sigma_0 t_0) (\exp((\eta_1 - \sigma_0)t) \\
 &\quad - \exp(\eta_1 - \sigma_0)t_0) \\
 &= \exp(-\eta_1(t-t_0))V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} (\exp(-\sigma_0(t-t_0)) - \exp(-\eta_1(t-t_0)))
 \end{aligned}$$

Therefore, $V(t)$ converge to zero exponentially as $t \rightarrow \infty$, and the convergence precision is related to η_1 and η .

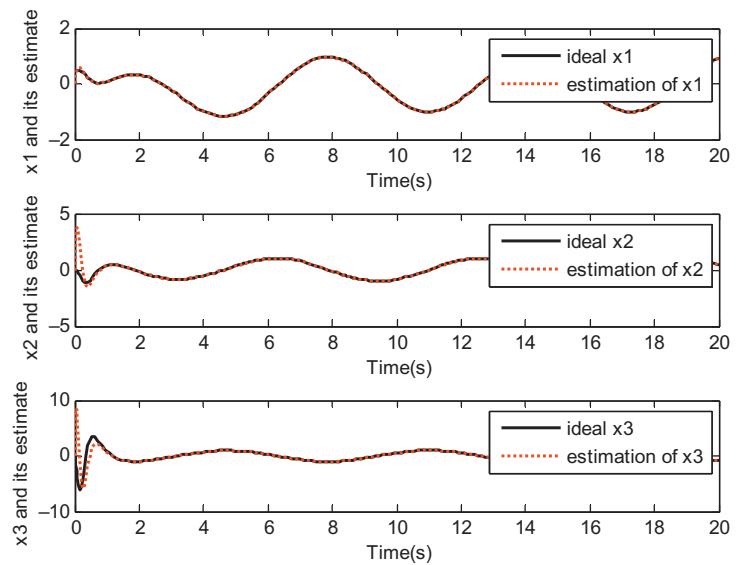
3.2.4 Simulation example

For plant (3.12), $K = 0.2$, $T = 18.5$, the initial states are $[0.5 \ 0 \ 0]$, and the ideal signal is sint .

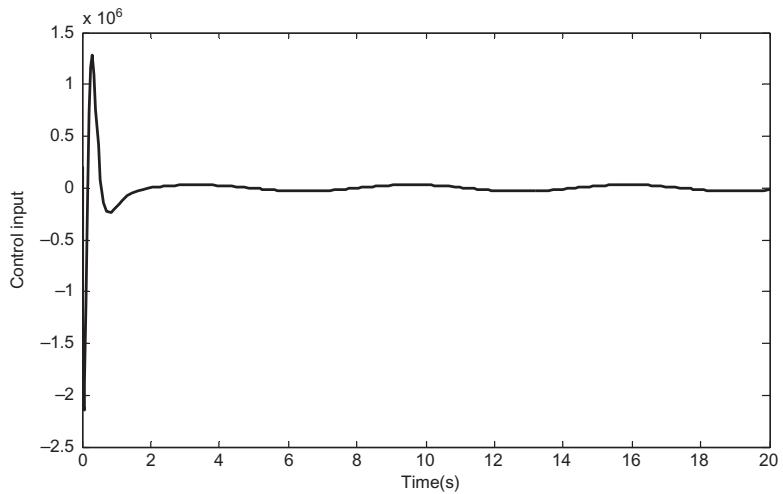
Using observers (3.15) and (3.16), $k = [k_1 \ k_2 \ k_3]^T$ is designed by Eq. (3.19), and using observer (3.20), $\eta = 0.50$, $c_1 = c_2 = 5$, simulation results are shown in Figs. 3.4–3.6.



■ FIGURE 3.4 Position, speed, and acceleration tracking.



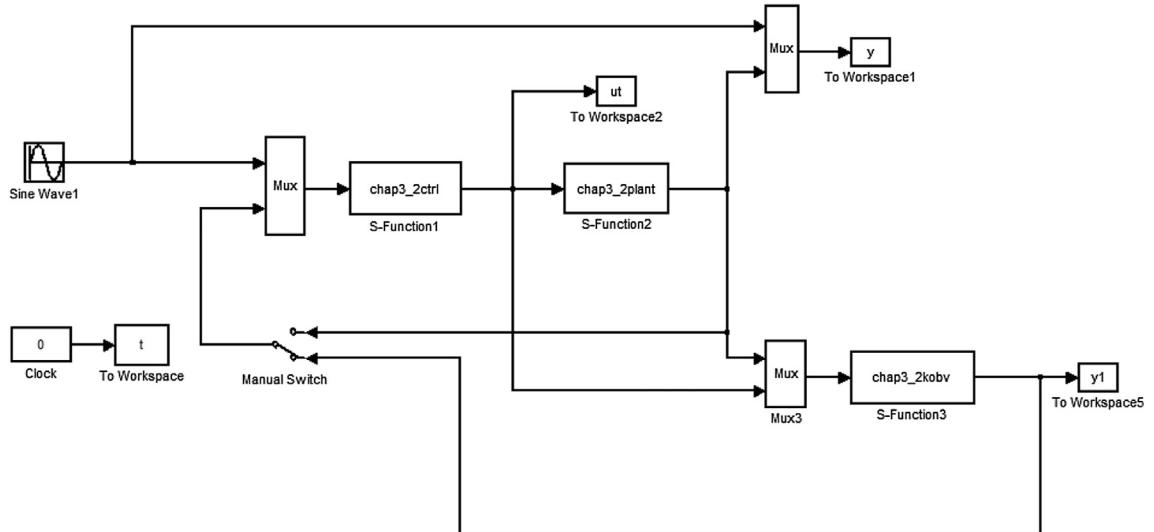
■ FIGURE 3.5 Estimation of position, speed, and acceleration signal.



■ FIGURE 3.6 Control input.

Simulation Programs:

1. Simulink main program: chap3_2sim.mdl



2. S function of Controller: chap3_2ctrl.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {1,2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [];

```

```

str = [];
ts  = [];
function sys=mdlOutputs(t,x,u)
c1=5;
c2=5;

yd=u(1);
dyd=cos(t);
ddyd=-sin(t);
dddyd=-cos(t);

x1=u(2);
x2=u(3);
x3=u(4);

e=x1-yd;
de=x2-dyd;
dde=x3-ddyd;

s=c1*e+c2*de+dde;
v=-dddyd+c1*de+c2*dde;

T=18.5;K=0.2;
b=K/(T^3);
alfa=-1/(T^3)*x1-3/(T^2)*x2-3/T*x3;
xite=0.50;
ut=-1/b*(v+alfa+xite*s);

sys(1)=ut;
3. S function of observer: chap3_2kobv.m
function [sys,x0,str,ts]=obv(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;

```

```
sizes.NumContStates = 6;
sizes.NumDiscStates = 0;
sizes.NumOutputs    = 3;
sizes.NumInputs     = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0;0;0;0;0;0];
str = [];
ts  = [0 0];
function sys=mdlDerivatives(t,x,u)
w=[x(1) x(2) x(3)]';
v=[x(4) x(5) x(6)]';

x1=u(1);
ut=u(4);

T=18.5;K=0.20;
A=[0 1 0;
   0 0 1;
   -1/T^3 -3/T^2 -3/T];

k1=30-3/T;
k2=300-3*k1/T-3/T^2;
k3=1000-1/T^3-3/T^2*k1-3/T*k2;

k=[k1 k2 k3]';
c=[1 0 0];
A0=A-k*c;

e4=[0 0 1]';

dw=A0*w+k*x1;
dv=A0*v+e4*ut;

sys(1)=dw(1);
sys(2)=dw(2);
sys(3)=dw(3);
sys(4)=dv(1);
sys(5)=dv(2);
sys(6)=dv(3);
function sys=mdlOutputs(t,x,u)
T=18.5;K=0.20;
b=K/T^3;
w=[x(1) x(2) x(3)]';
v=[x(4) x(5) x(6)]';
```

```

xp = w + b*v;

sys(1) = xp(1);
sys(2) = xp(2);
sys(3) = xp(3);

4. S function of plant: chap3_2plant.m
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[0.5,0,0];
str=[];
ts=[-1 0];
function sys=mdlDerivatives(t,x,u)
T=18.5;
K=0.20;
sys(1)=x(2);
sys(2)=x(3);
sys(3) = -1/(T^3)*x(1)-3/(T^2)*x(2)-3/T*x(3)+K/(T^3)
*u(1);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);

```

5. Plot program: chap3_2plot.m

```

close all;

figure(1);
subplot(311);
plot(t,y(:,1),'k',t,y(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('thd and y');
legend('ideal position','position tracking');
subplot(312);
plot(t,cos(t),'k',t,y(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('dthd and dy');
legend('ideal speed','speed tracking');
subplot(313);
plot(t,-sin(t),'k',t,y(:,4),'r','linewidth',2);
xlabel('time(s)');ylabel('ddthd and ddy');
legend('ideal acceleration','acceleration tracking');

figure(2);
subplot(311);
plot(t,y(:,2),'k',t,y1(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('x1 and its estimate');
legend('ideal x1','estimation of x1');
subplot(312);
plot(t,y(:,3),'k',t,y1(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('x2 and its estimate');
legend('ideal x2','estimation of x2');
subplot(313);
plot(t,y(:,4),'k',t,y1(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x3 and its estimate');
legend('ideal x3','estimation of x3');

figure(3);
plot(t,u(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('Control input');

```

3.3 SLIDING MODE CONTROL BASED ON A HIGH GAIN DIFFERENTIATOR

3.3.1 System description

Consider a plant as

$$G(s) = \frac{K}{(Ts+1)^3}, \quad (3.21)$$

where K is gain and T is time constant.

Eq. (3.21) can be expressed as

$$\dot{x}(t) = Ax(t) + Hu(t), \quad (3.22)$$

$$\text{where } \mathbf{x} = [x_1 \ x_2 \ x_3]^T, A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{T^3} & -\frac{3}{T^2} & -\frac{3}{T} \end{bmatrix},$$

$$H = \begin{bmatrix} 0 & 0 & \frac{K}{T^3} \end{bmatrix}^T.$$

Eq. (3.22) can also be written as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= \alpha(x) + bu \end{aligned}, \quad (3.23)$$

$$\text{where } \alpha(x) = -\frac{1}{T^3}x_1 - \frac{3}{T^2}x_2 - \frac{3}{T}x_3, b = \frac{K}{T^3}.$$

The control goal is as $t \rightarrow \infty$, $x_1 \rightarrow x_{1d}$, $\dot{x}_1 \rightarrow \dot{x}_{1d}$, $\ddot{x}_1 \rightarrow \ddot{x}_{1d}$.

For the model in Eq. (3.23), the design sliding mode function is

$$s = c_1e + c_2\dot{e} + \ddot{e}, \quad (3.24)$$

where $c_1 > 0$, $c_2 > 0$, $e = x_1 - x_{1d}$.

3.3.2 Traditional sliding mode control

$$\dot{s} = c_1\dot{e} + c_2\ddot{e} + \ddot{e} = c_1\dot{e} + c_2\ddot{e} + \ddot{x}_1 - \ddot{x}_{1d} = c_1\dot{e} + c_2\ddot{e} + \alpha(\mathbf{x}) + bu - \ddot{x}_{1d}.$$

Design the traditional sliding mode controller as

$$u(t) = \frac{1}{b}(-c_1\dot{e} - c_2\ddot{e} - \alpha(\mathbf{x}) - \eta sgn s + \ddot{x}_{1d}), \quad (3.25)$$

where $\eta > 0$.

The Lyapunov function is $V = \frac{1}{2}s^2$, then $\dot{V} = ss' = -\eta|s| \leq 0$.

From Eq. (3.25), we know that a speed signal and acceleration signal are needed in the controller, which is difficult in practical engineering.

3.3.3 High gain differentiator design

Three rank differentiator with a high gain can be designed as [3,4]

$$\begin{aligned}\hat{x}_1 &= \hat{x}_2 - \frac{k_1}{\varepsilon}(\hat{x}_1 - x_1(t)) \\ \hat{x}_2 &= \hat{x}_3 - \frac{k_2}{\varepsilon^2}(\hat{x}_1 - x_1(t)), \\ \hat{x}_3 &= -\frac{k_3}{\varepsilon^3}(\hat{x}_1 - x_1(t))\end{aligned}\quad (3.26)$$

where k_1 , k_2 , and k_3 are positive values and $\varepsilon \ll 1$.

Define $h_1 = \frac{k_1}{\varepsilon}$, $h_2 = \frac{k_2}{\varepsilon^2}$ and $h_3 = \frac{k_3}{\varepsilon^3}$; we then have

$$\begin{aligned}\dot{\tilde{x}}_1 &= \tilde{x}_2 - h_1 \tilde{x}_1 \\ \dot{\tilde{x}}_2 &= \tilde{x}_3 - h_2 \tilde{x}_1, \\ \dot{\tilde{x}}_3 &= -h_3 \tilde{x}_1\end{aligned}$$

where $\tilde{x} = x - \hat{x}$.

[Eq. \(3.27\)](#) can be written as $\dot{\tilde{x}} = A\tilde{x}$, $A = \begin{bmatrix} -h_1 & 1 & 0 \\ -h_2 & 0 & 1 \\ -h_3 & 0 & 0 \end{bmatrix}$, $\tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{bmatrix}$. If we design A as Hurwitz, we can get the convergence of \tilde{x} as

$$\|\tilde{x}(t)\| \leq \varphi_0 \|\tilde{x}(t_0)\| e^{-\sigma_0(t-t_0)}, \quad (3.27)$$

where φ_0 and σ_0 are positive constants.

[Eq. \(3.27\)](#) indicates that the convergence precision of $\|\tilde{x}(t)\|$ is related to σ_0 . Consider σ_0 is related to the minimum eigenvalue of A ; the smaller the ε value is, the bigger the h_i value is, then the smaller the minimum eigenvalue of A is, the bigger the σ_0 value is. Therefore, the convergence of $\|\tilde{x}(t)\|$ depends on ε , and a high gain observer will improve the convergence precision of $\|\tilde{x}(t)\|$ greatly.

To guarantee A to be Hurwitz, the characteristic equation of $\dot{\tilde{x}} = A\tilde{x}$ is

$$|\lambda I - A| = \begin{vmatrix} \lambda + h_1 & -1 & 0 \\ h_2 & \lambda & -1 \\ h_3 & 0 & \lambda \end{vmatrix} = 0, \text{ i.e., } \lambda^3 + h_1\lambda^2 + h_2\lambda + h_3 = 0.$$

Consider $\varepsilon \ll 1$, if we choose $(\lambda + 100)^3 = 0$, i.e., $\lambda^3 + 300\lambda^2 + 30,000\lambda + 1,000,000 = 0$, consider $\lambda^3 + h_1\lambda^2 + h_2\lambda + h_3 = 0$, we can design $h_1 = 300$, $h_2 = 30,000$, $h_3 = 1,000,000$, i.e., $k_1 = \varepsilon h_1$, $k_2 = \varepsilon^2 h_2$, $k_3 = \varepsilon^3 h_3$.

3.3.4 Sliding mode control based on a high gain differentiator

Using a high gain differentiator, the sliding mode controller can be designed as

$$u(t) = \frac{1}{b}(-c_1 \hat{e} - c_2 \hat{e} - \alpha(\hat{x}) - \eta \hat{s} + \ddot{x}_{1d}), \quad (3.28)$$

where $\hat{e} = \hat{x}_1 - x_{1d}$ and $\hat{s} = c_1\hat{e} + c_2\dot{\hat{e}} + \ddot{\hat{e}}$.

Then

$$\begin{aligned}\dot{s} &= c_1\dot{e} + c_2\ddot{e} + \ddot{e} = c_1\dot{e} + c_2\ddot{e} + \dot{\hat{x}}_3 - \ddot{x}_{1d} \\ &= c_1\dot{e} + c_2\ddot{e} + \alpha(\mathbf{x}) + bu - \ddot{x}_{1d} \\ &= c_1\dot{e} + c_2\ddot{e} + \alpha(\mathbf{x}) - c_1\hat{e} - c_2\dot{\hat{e}} - \alpha(\hat{\mathbf{x}}) - \eta\hat{s} + \ddot{x}_{1d} - \ddot{x}_{1d} \\ &= -\eta\hat{s} + v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})\end{aligned},$$

where $v(\tilde{\mathbf{x}}) = c_1\dot{e} + c_2\ddot{e} - c_1\hat{e} - c_2\dot{\hat{e}}$.

Design the Lyapunov function as $V = \frac{1}{2}s^2$, then

$$\begin{aligned}\dot{V} &= ss = -\eta s\hat{s} + s(v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})) \\ &= -\eta s(s - \tilde{s}) + s(v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})) \\ &= -\eta s^2 + s(\eta\tilde{s} + v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})) \\ &= -\eta s^2 + sf(\tilde{\mathbf{x}}) \leq -\eta s^2 + \frac{1}{2}(s^2 + f(\tilde{\mathbf{x}})^2) \\ &= -(\eta - 0.5)s^2 + 0.5f(\tilde{\mathbf{x}})^2 = -\eta_1 V + 0.5f(\tilde{\mathbf{x}})^2\end{aligned},$$

where $\eta_1 = 2\eta - 1 > 0$, $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$, $\tilde{s} = s - \hat{s}$, $f(\tilde{\mathbf{x}}) = \eta\tilde{s} + v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}})$.

Since $e - \hat{e} = \tilde{x}_1$, $\dot{e} - \dot{\hat{e}} = \tilde{x}_2$, $\ddot{e} - \ddot{\hat{e}} = \tilde{x}_3$, then

$$\tilde{s} = s - \hat{s} = c_1\tilde{x}_1 + c_2\tilde{x}_2 + \tilde{x}_3$$

$$v(\tilde{\mathbf{x}}) = c_1\dot{e} + c_2\ddot{e} - c_1\hat{e} - c_2\dot{\hat{e}} = c_1\tilde{x}_2 + c_2\tilde{x}_3$$

$$\begin{aligned}\alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}}) &= -\frac{1}{T^3}x_1 - \frac{3}{T^2}x_2 - \frac{3}{T}x_3 + \frac{1}{T^3}\hat{x}_1 + \frac{3}{T^2}\hat{x}_2 + \frac{3}{T}\hat{x}_3 \\ &= -\frac{1}{T^3}\tilde{x}_1 - \frac{3}{T^2}\tilde{x}_2 - \frac{3}{T}\tilde{x}_3\end{aligned},$$

then

$$\begin{aligned}f(\tilde{\mathbf{x}}) &= \eta\tilde{s} + v(\tilde{\mathbf{x}}) + \alpha(\mathbf{x}) - \alpha(\hat{\mathbf{x}}) = \left(\eta c_1 - \frac{1}{T^3}\right)\tilde{x}_1 + \left(\eta c_2 - \frac{3}{T^2} + c_1\right)\tilde{x}_2 \\ &\quad + \left(\eta - \frac{3}{T} + c_2\right)\tilde{x}_3.\end{aligned}$$

Consider $\|\tilde{\mathbf{x}}(t)\| \leq \varphi_0 \|\tilde{\mathbf{x}}(t_0)\| e^{-\sigma_0(t-t_0)}$, we can obtain

$$\dot{V} \leq -\eta_1 V + 0.5f(\tilde{\mathbf{x}})^2 \leq -\eta_1 V + \chi(\bullet)e^{-\sigma_0(t-t_0)},$$

where $\chi(\bullet)$ is a K-class function of $\|\tilde{\mathbf{x}}(t_0)\|$.

Using Lemma 1.3, the solution of $\dot{V} \leq -\eta_1 V + \chi(\bullet)e^{-\sigma_0(t-t_0)}$ is

$$\begin{aligned}
V(t) &\leq e^{-\eta_1(t-t_0)} V(t_0) + \chi(\bullet) \int_{t_0}^t e^{-\eta_1(t-\tau)} e^{-\sigma_0(\tau-t_0)} d\tau \\
&= e^{-\eta_1(t-t_0)} V(t_0) + \chi(\bullet) e^{-\eta_1 t + \sigma_0 t_0} \int_{t_0}^t e^{\eta_1 \tau} e^{-\sigma_0 \tau} d\tau \\
&= e^{-\eta_1(t-t_0)} V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} e^{-\eta_1 t + \sigma_0 t_0} e^{(\eta_1 - \sigma_0)\tau} \Big|_{t_0}^t , \\
&= e^{-\eta_1(t-t_0)} V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} e^{-\eta_1 t + \sigma_0 t_0} (e^{(\eta_1 - \sigma_0)t} - e^{(\eta_1 - \sigma_0)t_0}) \\
&= e^{-\eta_1(t-t_0)} V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} (e^{-\sigma_0(t-t_0)} - e^{-\eta_1(t-t_0)})
\end{aligned}$$

i.e.,

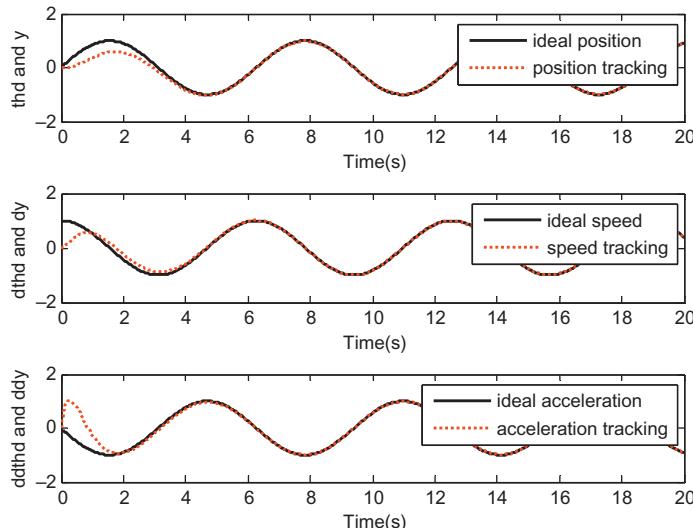
$$\lim_{t \rightarrow \infty} V(t) \leq 0.$$

Therefore, $V(t)$ converge to zero exponentially as $t \rightarrow \infty$, and the convergence precision is related to η_1 and η .

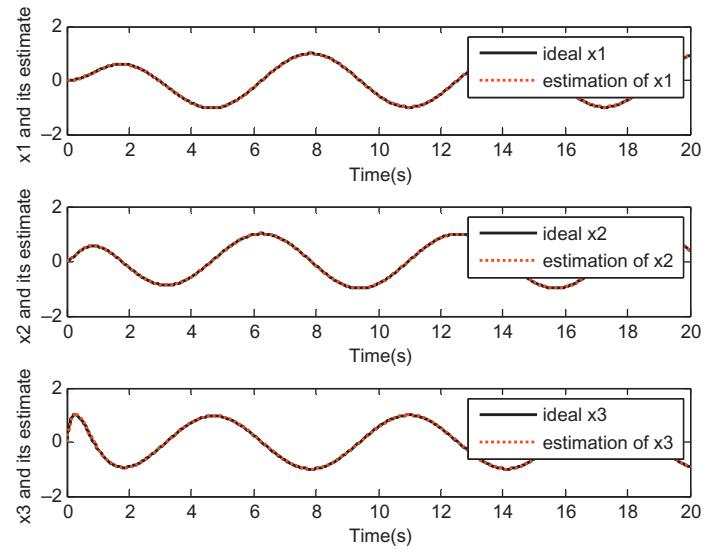
3.3.5 Simulation example

For plant (3.21), $K = 0.2$, $T = 18.5$, and initial states are $[0 \ 0 \ 0]$. The ideal position signal is $sint$.

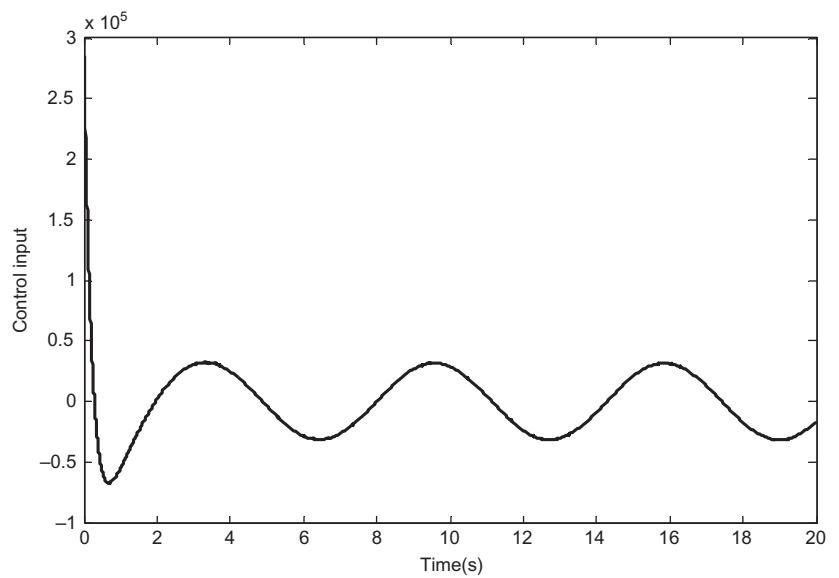
Using differentiator (3.26), $\varepsilon = 0.01$, $h_1 = 3$, $h_2 = 3$, $h_3 = 1$ is designed, and using controller (3.28), $\eta = 0.50$, $c_1 = c_2 = 5$, simulation results are shown in Figs. 3.7–3.9.



■ FIGURE 3.7 Position, speed, and acceleration tracking.



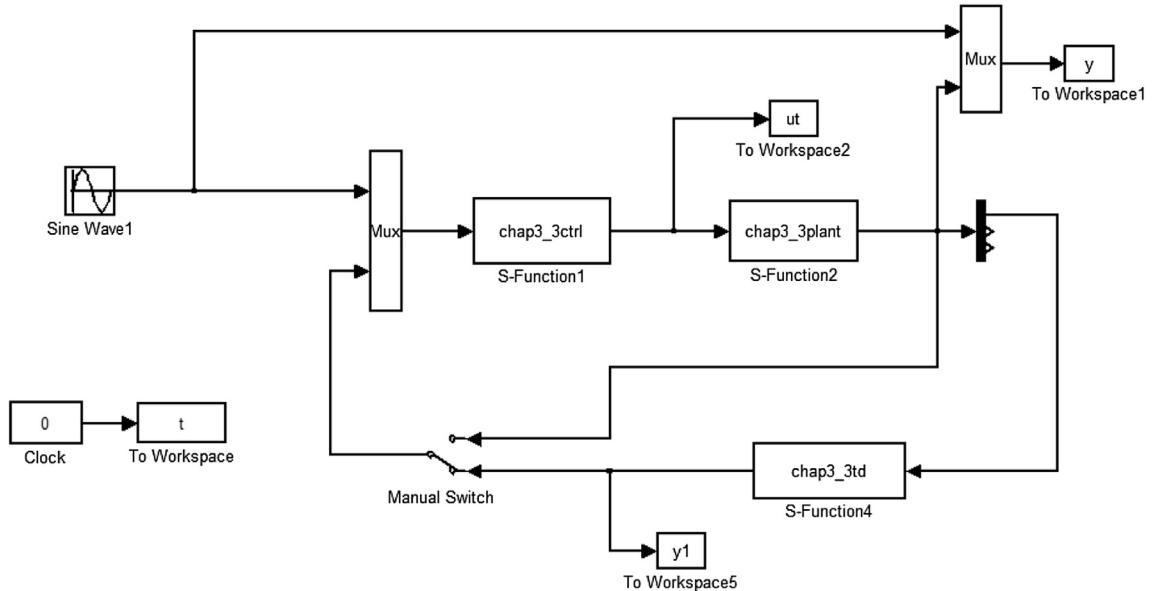
■ FIGURE 3.8 Position, speed, and acceleration estimation.



■ FIGURE 3.9 Control input.

Simulation Programs:

1. Simulink main program: chap3_3sim.mdl



2. S function of Controller: chap3_3ctrl.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {1,2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;

```

```

sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];
function sys=mdlOutputs(t,x,u)
c1=5;
c2=5;

yd=u(1);
dyd=cos(t);
ddyd=-sin(t);
dddyd=-cos(t);

x1=u(2);
x2=u(3);
x3=u(4);

e=x1-yd;
de=x2-dyd;
dde=x3-ddyd;

s=c1*e+c2*de+dde;
v=-dddyd+c1*de+c2*dde;

```

```

T=18.5;K=0.2;
b=K/(T^3);
alfa=-1/(T^3)*x1-3/(T^2)*x2-3/T*x3;
xite=0.50;
ut=-1/b*(v+alfa+xite*s);

```

```
    sys(1)=ut;
```

3. S function of observer: chap3_3td.m

```

function [sys,x0,str,ts] = Differentiator(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
    sys = [];

```

```

otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0 0 0];
str = [];
ts = [0 0];
function sys=mdlDerivatives(t,x,u)
vt=u(1);
epc=0.01;

h1=300;h2=30000;h3=1000000;

sys(1)=x(2)-h1*(x(1)-vt);%Kahlil TD
sys(2)=x(3)-h2*(x(1)-vt);
sys(3)=-h3*(x(1)-vt);
function sys=mdlOutputs(t,x,u)
sys = x;

```

4. S function of plant: chap3_3plant.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
```

```

sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[0,0,0];
str=[];
ts=[-1 0];
function sys=mdlDerivatives(t,x,u)
T=18.5;
K=0.20;

sys(1)=x(2);
sys(2)=x(3);
sys(3)=-1/(T^3)*x(1)-3/(T^2)*x(2)-3/T*x(3)+K/(T^3)
*u(1);%%+0.1*sin(t);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);

```

5. Plot program: chap3_3plot.m

```

close all;

figure(1);
subplot(311);
plot(t,y(:,1),'k',t,y(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('thd and y');
legend('ideal position','position tracking');
subplot(312);
plot(t,cos(t),'k',t,y(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('dthd and dy');
legend('ideal speed','speed tracking');
subplot(313);
plot(t,-sin(t),'k',t,y(:,4),'r','linewidth',2);
xlabel('time(s)');ylabel('ddthd and ddy');
legend('ideal acceleration','acceleration tracking');

figure(2);
subplot(311);
plot(t,y(:,2),'k',t,y1(:,1),'r','linewidth',2);

```

```

xlabel('time(s)');ylabel('x1 and its estimate');
legend('ideal x1','estimation of x1');
subplot(312);
plot(t,y(:,3),'k',t,y1(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('x2 and its estimate');
legend('ideal x2','estimation of x2');
subplot(313);
plot(t,y(:,4),'k',t,y1(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x3 and its estimate');
legend('ideal x3','estimation of x3');

figure(3);
plot(t,ut(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('Control input');

```

3.4 ROBUST STATE OBSERVER DESIGN FOR A FLEXIBLE MANIPULATOR

The flexible-joint robot has been a typical and challenging plant for control strategy research because of the highly coupled and nonlinear nature of its dynamic behavior. Compared with the rigid-joint robot, the flexible-joint robot system is far more complex. However, taking joint flexibility into account has a great effect on performance improvement [5,6]. For a real FJ robot system, physical parameters are generally unknown and some states cannot be measured. Considering parametric uncertainties and the reduction of measurement requirement, the observer design for flexible-joint robots is extremely difficult.

3.4.1 Problem statement

The dynamic model of a single-link FJ robot can be described as

$$\begin{cases} I\ddot{q} + K(q - q_m) + Mglsinq = 0 \\ J\ddot{q}_m - K(q - q_m) = u \end{cases}, \quad (3.29)$$

where q and q_m are the angular positions of the link and the motor shaft, respectively, I and J denote the link inertia and rotor inertia, respectively, K represents the joint flexibility, M , g , and l denote the link mass, gravity constant and the distance to the joint from the mass-centre, respectively, and u is the motor torque.

Define the state space variables as $x_1 = q$, $x_2 = \dot{q}$, $x_3 = q_m$, and $x_4 = \dot{q}_m$ and the single-link FJ robot system in state space is easily described as follows:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1 x_3 + f_1(x_1) + \Delta_1(t) \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = a_2 u + f_2(x_1, x_3) + \Delta_2(t) \\ y = x_1 \end{cases}, \quad (3.30)$$

where $a_1 = \frac{K}{I}$, $f_1(x_1) = -\frac{Mgl}{I} \sin x_1 - \frac{K}{I} x_1$, $a_2 = 1/J$, $f_2(x_1, x_3) = \frac{K}{J}(x_1 - x_3)$, $\Delta_1(t)$ and $\Delta_2(t)$ are uncertain parts, $|\Delta_1(t)| \leq \rho_1$, $|\Delta_2(t)| \leq \rho_2$.

3.4.2 Robust observer design

From paper [7], a robust observer is designed with the following two parts: firstly, a dynamic observer equation is designed as

$$\begin{cases} \dot{\lambda}_1 = \lambda_2 + l_1(x_1 - \lambda_1) + D_1(x_1 - \lambda_1) \\ \dot{\lambda}_2 = a_1 x_3 + f_1(x_1) + \bar{D}_2(x_1 - \lambda_1) \\ \dot{\lambda}_3 = \lambda_4 + l_2(x_3 - \lambda_3) + D_3(x_3 - \lambda_3) \\ \dot{\lambda}_4 = a_2 u + f_2(x_1, x_3) + \bar{D}_4(x_3 - \lambda_3) \end{cases}, \quad (3.31)$$

where l_1 , l_2 , D_1 , \bar{D}_2 , D_3 , and \bar{D}_4 are positive value to be designed.

Then the states estimation can be obtained as

$$\begin{cases} \hat{x}_1 = \lambda_1 \\ \hat{x}_2 = \lambda_2 + l_1(x_1 - \lambda_1) \\ \hat{x}_3 = \lambda_3 \\ \hat{x}_4 = \lambda_4 + l_2(x_3 - \lambda_3) \end{cases}, \quad (3.32)$$

where \hat{x}_i is the estimation state.

Define the estimation state error

$$\tilde{x}_i = x_i - \hat{x}_i \quad (3.33)$$

From Eqs. (3.31) and (3.33), we have

$$\begin{aligned} \dot{\tilde{x}}_1 &= \lambda_2 + l_1(x_1 - \lambda_1) + D_1(x_1 - \lambda_1) = \hat{x}_2 + D_1 \tilde{x}_1 \\ \dot{\tilde{x}}_2 &= a_1 x_3 + f_1(x_1) + \bar{D}_2(x_1 - \lambda_1) + l_1(x_2 - \hat{x}_2 - D_1 \tilde{x}_1) \\ &= a_1 x_3 + f_1(x_1) + l_1 \tilde{x}_2 + (\bar{D}_2 - l_1 D_1) \tilde{x}_1 \\ \dot{\tilde{x}}_3 &= \lambda_4 + l_2(x_3 - \lambda_3) + D_3(x_3 - \lambda_3) = \hat{x}_4 + D_3 \tilde{x}_3 \\ \dot{\tilde{x}}_4 &= a_2 u + f_2(x_1, x_3) + \bar{D}_4(x_3 - \lambda_3) + l_2(x_4 - \hat{x}_4 - D_3 \tilde{x}_3) \\ &= a_2 u + f_2(x_1, x_3) + l_2 \tilde{x}_4 + (\bar{D}_4 - l_2 D_3) \tilde{x}_3 \end{aligned}. \quad (3.34)$$

Define $D_2 = \bar{D}_2 - l_1 D_1$, $D_4 = \bar{D}_4 - l_2 D_3$, then

$$\begin{cases} \dot{\hat{x}}_1 = \hat{x}_2 + D_1 \tilde{x}_1 \\ \dot{\hat{x}}_2 = a_1 x_3 + f_1(x_1) + l_1 \tilde{x}_2 + D_2 \tilde{x}_1 \\ \dot{\hat{x}}_3 = \hat{x}_4 + D_3 \tilde{x}_3 \\ \dot{\hat{x}}_4 = a_2 u + f_2(x_1, x_3) + l_2 \tilde{x}_4 + D_4 \tilde{x}_3 \end{cases}. \quad (3.35)$$

3.4.3 Observer analysis

Design the Lyapunov function as

$$V = \frac{1}{2} \sum_{i=1}^4 \tilde{x}_i^2.$$

We then have

$$\begin{aligned} \dot{V} &= \tilde{x}_1(x_2 - \hat{x}_2 - D_1 \tilde{x}_1) + \tilde{x}_2(\Delta_1 - l_1 \tilde{x}_2 - D_2 \tilde{x}_1) \\ &\quad + \tilde{x}_3(x_4 - \hat{x}_4 - D_3 \tilde{x}_3) + \tilde{x}_4(\Delta_2 - l_2 \tilde{x}_4 - D_4 \tilde{x}_3) \\ &= (1 - D_2)\tilde{x}_1\tilde{x}_2 + (1 - D_4)\tilde{x}_3\tilde{x}_4 - D_1\tilde{x}_1^2 - l_1\tilde{x}_2^2 - D_3\tilde{x}_3^2 - l_2\tilde{x}_4^2 + \Delta_1\tilde{x}_2 + \Delta_2\tilde{x}_4 \end{aligned}$$

Set

$$D_2 = D_4 = 1 \quad (3.36)$$

From $\frac{\rho_i^2}{2} + \frac{\tilde{x}_j^2}{2} \geq \rho_i |\tilde{x}_j| \geq \Delta_i \tilde{x}_j$, we have

$$\begin{aligned} \dot{V} &\leq -D_1\tilde{x}_1^2 - l_1\tilde{x}_2^2 - D_3\tilde{x}_3^2 - l_2\tilde{x}_4^2 + \frac{\rho_1^2}{2} + \frac{\tilde{x}_2^2}{2} + \frac{\rho_2^2}{2} + \frac{\tilde{x}_4^2}{2} \\ &= - \left(D_1\tilde{x}_1^2 + D_3\tilde{x}_3^2 + \left(l_1 - \frac{1}{2} \right) \tilde{x}_2^2 + \left(l_2 - \frac{1}{2} \right) \tilde{x}_4^2 \right) + \frac{\rho_1^2}{2} + \frac{\rho_2^2}{2}. \end{aligned}$$

If we set r as

$$l_1 \geq \frac{1}{2} + r, \quad l_2 \geq \frac{1}{2} + r, \quad D_1 \geq r, \quad D_3 \geq r, \quad (3.37)$$

where r is a positive value to be designed, then

$$D_1\tilde{x}_1^2 + D_3\tilde{x}_3^2 + \left(l_1 - \frac{1}{2} \right) \tilde{x}_2^2 + \left(l_2 - \frac{1}{2} \right) \tilde{x}_4^2 \geq r(\tilde{x}_1^2 + x_3^2 + \tilde{x}_2^2 + \tilde{x}_4^2).$$

Therefore

$$\dot{V} \leq -r(\tilde{x}_1^2 + x_3^2 + \tilde{x}_2^2 + \tilde{x}_4^2) + \frac{\rho_1^2}{2} + \frac{\rho_2^2}{2} = -2rV + Q,$$

where $Q = \frac{\rho_1^2}{2} + \frac{\rho_2^2}{2}$.

Using Lemma 1.3, the solution of $\dot{V} \leq -2rV + Q$ is

$$\begin{aligned} V(t) &\leq e^{-2r(t-t_0)}V(t_0) + Qe^{-2rt} \int_{t_0}^t e^{2r\tau} d\tau = e^{-2r(t-t_0)}V(t_0) + \frac{Qe^{-2rt}}{2r}(e^{2rt} - e^{2rt_0}) \\ &= e^{-2r(t-t_0)}V(t_0) + \frac{Q}{2r}(1 - e^{-2r(t-t_0)}) \end{aligned},$$

i.e.,

$$V(t) \leq \frac{Q}{2r} + \left(V(t_0) - \frac{Q}{2r} \right) e^{-2r(t-t_0)}.$$

Obviously, all the signals in the system are globally bounded, and

$$\lim_{t \rightarrow \infty} V(t) \leq \frac{Q}{2r}.$$

From the above, we can see the convergence precision is determined by the upper bound of Δ_1 , Δ_2 and the initial error value of the observer. By taking any large size of r , the observation error can be arbitrarily small.

Moreover, if $\Delta_1(t) = 0$, $\Delta_2(t) = 0$, we have $Q = \frac{\rho_1^2}{2} + \frac{\rho_2^2}{2} = 0$, then $\dot{V} \leq -2rV$, i.e.,

$$V(t) \leq e^{-2r(t-t_0)}V(t_0).$$

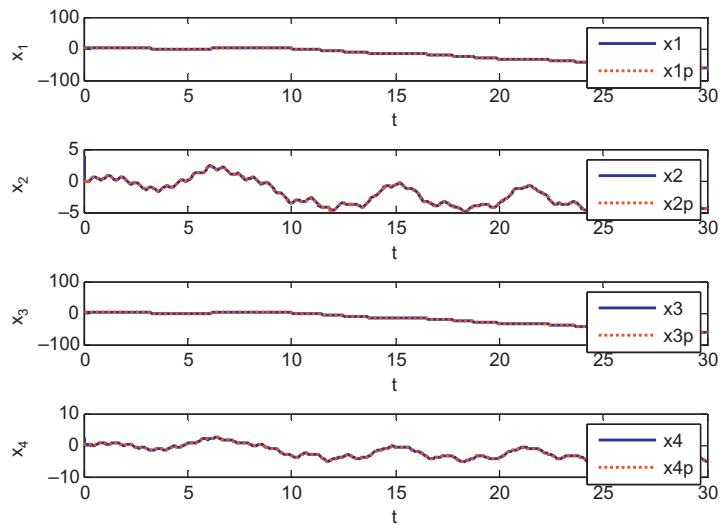
At this point, the observer is exponentially convergent.

3.4.4 Simulation example

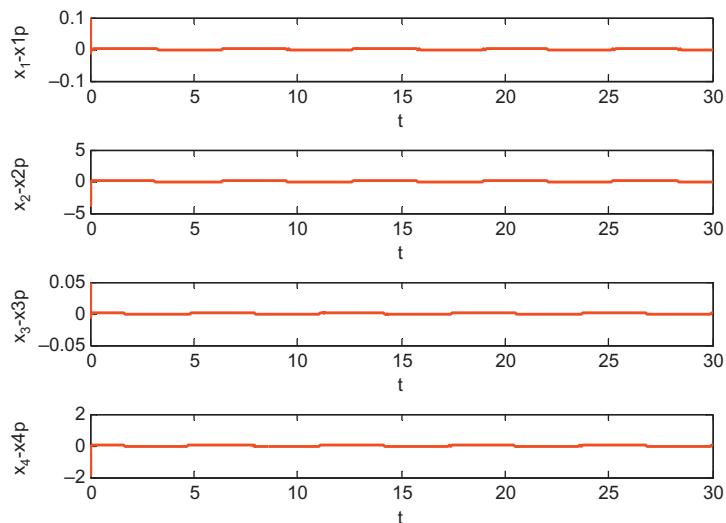
Consider the plant (3.29), the physical parameters which are only used for plant construction are chosen as $I = 1$, $J = 1$, $K = 40$ and $Mgl = 5$, $\Delta_1 = \sin t$ and $\Delta_2 = \cos t$ are assumed.

The initial states for the plant are set as $x(0) = [0.1 \ 0 \ 0.05 \ 0]^T$, and the initial states for the observer are set as $\lambda(0) = [0 \ 0 \ 0 \ 0]^T$.

Using observers (3.31) and (3.32), choose $r = 100$, according to Eqs. (3.36) and (3.37), set $l_1 = 101$, $l_2 = 101$, $D_2 = D_4 = 1.0$ and $D_1 = D_3 = 101$. Simulation results are shown in Figs. 3.10 and 3.11.



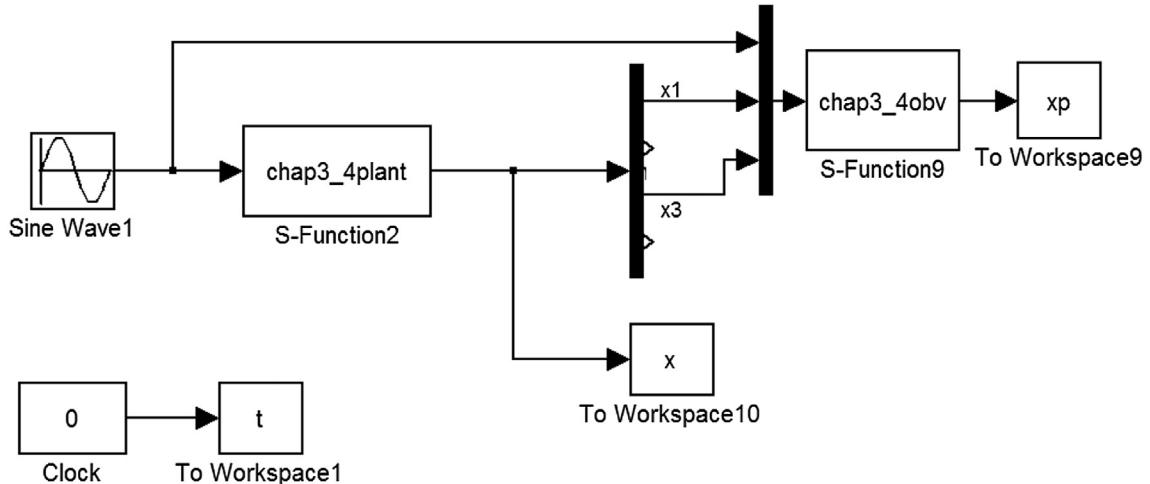
■ FIGURE 3.10 States estimation.



■ FIGURE 3.11 States estimation error.

Simulink programs

1. Simulink main program: chap3_4sim.mdl



2. S function for observer: chap3_4obv.m

```

function [sys,x0,str,ts] = obv(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

```

```

sys = simsizes(sizes);
x0  = [0;0;0;0];
str = [];
ts  = [0 0];
function sys=mdlDerivatives(t,x,u)
I=1.0;J=1.0;Mg1=5.0;K=40;

r=100;
l1=101;l2=101;
D1=101;D3=101;
D2=1;D4=1;
D2_bar=D2+l1*D1;
D4_bar=D4+l2*D3;

ut=u(1);
x1=u(2);
x3=u(3);
a1=K/I;a2=1/J;
f1=-Mg1/I*sin(x1)-K/I*x1;
f2=K/J*(x1-x3);
sys(1)=x(2)+l1*(x1-x(1))+D1*(x1-x(1));
sys(2)=a1*x3+f1+D2_bar*(x1-x(1));
sys(3)=x(4)+l2*(x3-x(3))+D3*(x3-x(3));
sys(4)=a2*ut+f2+D4_bar*(x3-x(3));
function sys=mdlOutputs(t,x,u)
l1=101;l2=101;

ut=u(1);
x1=u(2);
x3=u(3);

sys(1)=x(1);
sys(2)=x(2)+l1*(x1-x(1));
sys(3)=x(3);
sys(4)=x(4)+l2*(x3-x(3));

```

- 3. S function for plant: chap3_4plant.m**
- ```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);

```

```

case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0.1;0;0.05;0];
str = [];
ts = [0 0];
function sys=mdlDerivatives(t,x,u)
I=1.0;J=1.0;Mg1=5.0;K=40;
a1=K/I;
f1=-Mg1/I*sin(x(1))-K/I*x(1);
a2=1/J;
f2=K/J*(x(1)-x(3));
delta1=sin(t);
delta2=cos(t);

ut=u(1);
sys(1)=x(2);
sys(2)=a1*x(3)+f1+delta1;
sys(3)=x(4);
sys(4)=a2*ut+f2+delta2;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);

```

**4. Plot program: chap3\_4plot.m**

```

close all;

figure(1);

```

```

subplot(411);
plot(t,xp(:,1),'b',t,x(:,1),'r','linewidth',2);
xlabel('t');ylabel('x_1');
legend('x1','x1p');
subplot(412);
plot(t,xp(:,2),'b',t,x(:,2),'r','linewidth',2);
xlabel('t');ylabel('x_2');
legend('x2','x2p');
subplot(413);
plot(t,xp(:,3),'b',t,x(:,3),'r','linewidth',2);
xlabel('t');ylabel('x_3');
legend('x3','x3p');
subplot(414);
plot(t,xp(:,4),'b',t,x(:,4),'r','linewidth',2);
xlabel('t');ylabel('x_4');
legend('x4','x4p');

figure(2);
subplot(411);
plot(t,x(:,1)-xp(:,1),'r','linewidth',2);
xlabel('t');ylabel('x_1-x1p');
subplot(412);
plot(t,x(:,2)-xp(:,2),'r','linewidth',2);
xlabel('t');ylabel('x_2-x2p');
subplot(413);
plot(t,x(:,3)-xp(:,3),'r','linewidth',2);
xlabel('t');ylabel('x_3-x3p');
subplot(414);
plot(t,x(:,4)-xp(:,4),'r','linewidth',2);
xlabel('t');ylabel('x_4-x4p');

```

### 3.5 SLIDING MODE CONTROL FOR FLEXIBLE MANIPULATOR BASED ON A ROBUST STATE OBSERVER

Consider system (3.30) and choose  $\Delta_1(t) = \Delta_2(t) = 0$ ; we have

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1 x_3 + f_1(x_1) \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = a_2 u + f_2(x_1, x_3) \\ y = x_1 \end{cases}, \quad (3.38)$$

where  $a_1 = \frac{K}{I}$ ,  $f_1(x_1) = -\frac{Mgl}{I} \sin x_1 - \frac{K}{I}x_1$ ,  $a_2 = \frac{1}{J}$  and  $f_2(x_1, x_3) = \frac{K}{J}(x_1 - x_3)$ ,  $u$  is control input.

From Eq. (3.31), we have a robust state observer as follows:

$$\begin{cases} \dot{\lambda}_1 = \lambda_2 + l_1(x_1 - \lambda_1) + D_1(x_1 - \lambda_1) \\ \dot{\lambda}_2 = a_1 x_3 + f_1(x_1) + \bar{D}_2(x_1 - \lambda_1) \\ \dot{\lambda}_3 = \lambda_4 + l_2(x_3 - \lambda_3) + D_3(x_3 - \lambda_3) \\ \dot{\lambda}_4 = a_2 u + f_2(x_1, x_3) + \bar{D}_4(x_3 - \lambda_3) \end{cases} \quad (3.39)$$

and

$$\begin{cases} \hat{x}_1 = \lambda_1 \\ \hat{x}_2 = \lambda_2 + l_1(x_1 - \lambda_1) \\ \hat{x}_3 = \lambda_3 \\ \hat{x}_4 = \lambda_4 + l_2(x_3 - \lambda_3) \end{cases} \quad (3.40)$$

Design the Lyapunov function of the observer as  $V_o = \frac{1}{2} \sum_{i=1}^4 \hat{x}_i^2$ , from section 3.4; if  $\Delta_1(t) = \Delta_2(t) = 0$ , we have  $Q = 0$ , and

$$V_o(t) \leq e^{-2r(t-t_0)} V_o(t_0)$$

At this point, the observer is exponentially convergent.

### 3.5.1 Sliding mode controller design

The control task is  $x_1 \rightarrow x_d$  and  $x_2 \rightarrow \dot{x}_d$ , to simplify the expression, we use  $f_1$  to replace  $f_1(x_1)$ , and use  $f_2$  to replace  $f_1(x_1, x_3)$ .

The error state equation can be expressed as

$$\begin{aligned} e_1 &= x_1 - x_d \\ e_2 &= \dot{e}_1 = x_2 - \dot{x}_d \\ e_3 &= \ddot{e}_1 = \dot{x}_2 - \ddot{x}_d = a_1 x_3 + f_1 - \ddot{x}_d \\ e_4 &= \ddot{e}_1 = a_1 \dot{x}_3 + \dot{f}_1 - \ddot{x}_d = a_1 x_4 + \dot{f}_1 - \ddot{x}_d \end{aligned} \quad (3.41)$$

Define

$$\begin{aligned} \hat{e}_1 &= \hat{x}_1 - x_d \\ \hat{e}_2 &= \hat{x}_2 - \dot{x}_d \\ \hat{e}_3 &= a_1 \hat{x}_3 + \hat{f}_1 - \ddot{x}_d \\ \hat{e}_4 &= a_1 \hat{x}_4 + \dot{\hat{f}}_1 - \ddot{x}_d \end{aligned}$$

We then have

$$\begin{aligned} \tilde{e}_1 &= e_1 - \hat{e}_1 = \tilde{x}_1 \\ \tilde{e}_2 &= e_2 - \hat{e}_2 = \tilde{x}_2 \\ \tilde{e}_3 &= e_3 - \hat{e}_3 = a_1 \tilde{x}_3 + f_1 - \hat{f}_1 \\ \tilde{e}_4 &= e_4 - \hat{e}_4 = a_1 \tilde{x}_4 + \dot{f}_1 - \dot{\hat{f}}_1 \end{aligned}$$

The sliding mode function is designed as

$$s = c_1 e_1 + c_2 e_2 + c_3 e_3 + e_4, \quad (3.42)$$

where  $c_i > 0$  must be designed to be Hurwitz,  $i = 1, 2, 3$ .

Then

$$\tilde{s} = c_1 \tilde{e}_1 + c_2 \tilde{e}_2 + c_3 \tilde{e}_3 + \tilde{e}_4 = c_1 \tilde{x}_1 + c_2 \tilde{x}_2 + c_3 (a_1 \tilde{x}_3 + f_1 - \hat{f}_1) + a_1 \tilde{x}_4 + \dot{f}_1 - \hat{\dot{f}}_1.$$

Design the sliding mode controller as

$$\begin{aligned} u = -\frac{1}{a_1 a_2} & \left[ c_1 (\hat{x}_2 - \dot{x}_d) + c_2 (a_1 \hat{x}_3 + \hat{f}_1 - \ddot{x}_d) \right. \\ & \left. + c_3 (a_1 \hat{x}_4 + \hat{f}_1 - \ddot{x}_d) + a_1 \hat{f}_2 + \hat{f}_1 - \ddot{x}_d + \eta \hat{s} \right], \end{aligned} \quad (3.43)$$

where  $\eta > 0$ .

Then

$$\begin{aligned} \dot{s} &= c_1 \dot{e}_1 + c_2 \dot{e}_2 + c_3 \dot{e}_3 + \dot{e}_4 \\ &= c_1 (x_2 - \dot{x}_d) + c_2 (a_1 x_3 + f_1 - \ddot{x}_d) + c_3 (a_1 x_4 + \dot{f}_1 - \ddot{x}_d) + a_1 (a_2 u + f_2) + \ddot{f}_1 - \ddot{x}_d \\ &= c_1 (x_2 - \dot{x}_d) + c_2 (a_1 x_3 + f_1 - \ddot{x}_d) + c_3 (a_1 x_4 + \dot{f}_1 - \ddot{x}_d) \\ &\quad - [c_1 (\hat{x}_2 - \dot{x}_d) + c_2 (a_1 \hat{x}_3 + \hat{f}_1 - \ddot{x}_d) + c_3 (a_1 \hat{x}_4 + \hat{f}_1 - \ddot{x}_d) + a_1 \hat{f}_2 + \hat{f}_1 \\ &\quad - \ddot{x}_d + \eta \hat{s}] + a_1 f_2 + \ddot{f}_1 - \ddot{x}_d \\ &= c_1 \tilde{x}_2 + c_2 a_1 \tilde{x}_3 + c_3 a_1 \tilde{x}_4 + c_2 (f_1 - \hat{f}_1) + c_3 (\dot{f}_1 - \hat{f}_1) \\ &\quad + a_1 (f_2 - \hat{f}_2) + (\ddot{f}_1 - \hat{\dot{f}}_1) - \eta (s - \tilde{s}) \end{aligned}$$

Design the Lyapunov function for the controller as

$$V_c = \frac{1}{2} s^2.$$

Then

$$\begin{aligned} \dot{V}_c &= s \dot{s} = s [c_1 \tilde{x}_2 + c_2 a_1 \tilde{x}_3 + c_3 a_1 \tilde{x}_4 + c_2 (f_1 - \hat{f}_1) + c_3 (\dot{f}_1 - \hat{f}_1) \\ &\quad + a_1 (f_2 - \hat{f}_2) + (\ddot{f}_1 - \hat{\dot{f}}_1) - \eta (s - \tilde{s})] \\ &= -\eta s^2 + \eta s (c_1 \tilde{x}_1 + c_2 \tilde{x}_2 + c_3 (a_1 \tilde{x}_3 + f_1 - \hat{f}_1) + a_1 \tilde{x}_4 + \dot{f}_1 - \hat{f}_1) \\ &\quad + s [c_1 \tilde{x}_2 + c_2 a_1 \tilde{x}_3 + c_3 a_1 \tilde{x}_4 + c_2 (f_1 - \hat{f}_1) + c_3 (\dot{f}_1 - \hat{f}_1) \\ &\quad + a_1 (f_2 - \hat{f}_2) + (\ddot{f}_1 - \hat{\dot{f}}_1)] \\ &= -\eta s^2 + s [\eta c_1 \tilde{x}_1 + (\eta c_2 + c_1) \tilde{x}_2 + (\eta c_3 a_1 + c_2 a_1) \tilde{x}_3 \\ &\quad + (\eta a_1 + c_3 a_1) \tilde{x}_4 + (\eta c_3 + c_2) (f_1 - \hat{f}_1) \\ &\quad + (\eta + c_3) (\dot{f}_1 - \hat{f}_1) + a_1 (f_2 - \hat{f}_2) + (\ddot{f}_1 - \hat{\dot{f}}_1)] \\ &= -\eta s^2 + s \chi(\tilde{\mathbf{x}}) \leq -\eta s^2 + \frac{1}{2} s^2 + \frac{1}{2} \chi^2(\tilde{\mathbf{x}}) \\ &= \left( \frac{1}{2} - \eta \right) s^2 + \frac{1}{2} \chi^2(\tilde{\mathbf{x}}) = (1 - 2\eta) V_c + \frac{1}{2} \chi^2(\tilde{\mathbf{x}}) \end{aligned}$$

where

$$\begin{aligned}\chi(\tilde{\mathbf{x}}) = & [\eta c_1 \tilde{x}_1 + (\eta c_2 + c_1) \tilde{x}_2 + (\eta c_3 a_1 + c_2 a_1) \tilde{x}_3 + (\eta a_1 + c_3 a_1) \tilde{x}_4 \\ & + (\eta c_3 + c_2)(f_1 - \hat{f}_1) + (\eta + c_3)(\hat{f}_1 - \tilde{f}_1) + a_1(f_2 - \hat{f}_2) + (\tilde{f}_1 - \hat{f}_1)].\end{aligned}$$

From the observer analysis, we have if  $t \rightarrow \infty$ ,  $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4$  converge to zero exponentially. Consider  $f_1(x_1) = -\frac{Mgl}{I} \sin x_1 - \frac{K}{I} x_1$ ,  $a_2 = \frac{1}{J}$  and  $f_2(x_1, x_3) = \frac{K}{J}(x_1 - x_3)$ , then  $f_1 - \hat{f}_1, \hat{f}_1 - \tilde{f}_1, f_2 - \hat{f}_2, \tilde{f}_1 - \hat{f}_1$  will also converge to zero exponentially, therefore,  $\chi(\tilde{\mathbf{x}})$  and  $\chi^2(\tilde{\mathbf{x}})$  will also converge to zero exponentially.

For the closed system, design the Lyapunov function as

$$V = V_o + V_c. \quad (3.44)$$

Then

$$\dot{V} = \dot{V}_o + \dot{V}_c \leq -2rV_o - (2\eta - 1)V_c + \frac{1}{2}\chi^2(\tilde{\mathbf{x}}) \leq -\eta_1 V + \chi(\bullet)e^{-\sigma_0(t-t_0)},$$

where  $\eta_1 = \{2r, (2\eta - 1)\}_{\max}$ ,  $\chi(\bullet)$  is a  $K$ -class function of  $\|\tilde{\mathbf{x}}(t_0)\|$ ,  $\sigma_0 > 0$ .

Using Lemma 1.3, the solution of  $\dot{V} \leq -\eta_1 V + \chi(\bullet)e^{-\sigma_0(t-t_0)}$  is

$$\begin{aligned}V(t) &\leq e^{-\eta_1(t-t_0)}V(t_0) + \chi(\bullet) \int_{t_0}^t e^{-\eta_1(t-\tau)}e^{-\sigma_0(\tau-t_0)}d\tau \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \chi(\bullet)e^{-\eta_1 t + \sigma_0 t_0} \int_{t_0}^t e^{\eta_1 \tau} e^{-\sigma_0 \tau} d\tau \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} e^{-\eta_1 t + \sigma_0 t_0} e^{(\eta_1 - \sigma_0)\tau} \Big|_{t_0}^t \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} e^{-\eta_1 t + \sigma_0 t_0} (e^{(\eta_1 - \sigma_0)t} - e^{(\eta_1 - \sigma_0)t_0}) \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \frac{\chi(\bullet)}{\eta_1 - \sigma_0} (e^{-\sigma_0(t-t_0)} - e^{-\eta_1(t-t_0)})\end{aligned}, \quad (3.45)$$

i.e.,

$$\lim_{t \rightarrow \infty} V(t) \leq 0.$$

Since  $V(t) \geq 0$ , then if  $t \rightarrow \infty$ , we have  $V(t) = 0$  and  $V(t)$  converge to zero exponentially, and convergence precision depends on  $\eta_1$ , i.e.,  $\eta$  and  $r$ .

If  $s = 0$ , we have  $e_4 = -c_1 e_1 - c_2 e_2 - c_3 e_3$ . Define  $\mathbf{E}_1 = [e_1 \ e_2 \ e_3]^T$ , then we have  $\dot{\mathbf{E}}_1 = A\mathbf{E}_1$ . If we design  $c_1, c_2, c_3$  to make  $A$  be Hurwitz, then  $t \rightarrow \infty, \mathbf{E}_1 = [e_1 \ e_2 \ e_3]^T \rightarrow 0$ .

To design  $\mathbf{A}$  as Hurwitz, we need the roots of

$$|\mathbf{A} - \lambda\mathbf{I}| = \begin{vmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \\ -c_1 & -c_2 & -c_3 - \lambda \end{vmatrix} = \lambda^2(-c_3 - \lambda) - c_1 - c_2\lambda = -\lambda^3 - c_3\lambda^2 - c_2\lambda - c_1 = 0$$

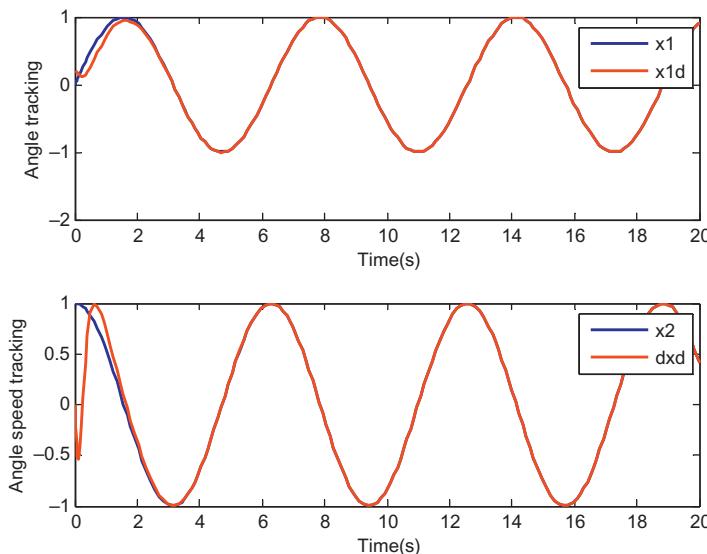
to be negative. From  $(\lambda + 10)^3 = 0$ , we have  $\lambda^3 + 30\lambda^2 + 300\lambda + 1000 = 0$ , then we can choose  $c_1 = 1000$ ,  $c_2 = 300$  and  $c_3 = 30$ .

### 3.5.2 Simulation example

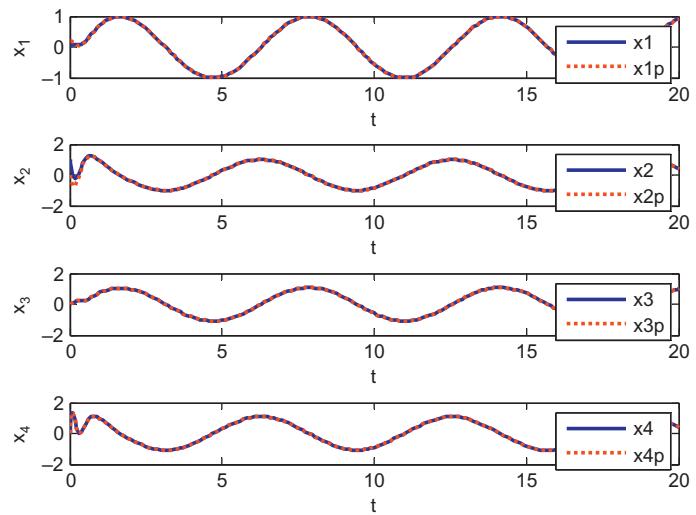
Consider the plant (3.38), the physical parameters which are only used for plant construction are chosen as  $I = 1$ ,  $J = 1$ ,  $K = 40$ , and  $Mgl = 5$ .

The initial states for the plant are set as  $x(0) = [0.2 \ 0 \ 0 \ 0]^T$ , and the initial states for the observer are set as  $\lambda(0) = [0 \ 0 \ 0 \ 0]^T$ .

Define the ideal signal as  $x_d = \sin t$ , use observers (3.39) and (3.40), choose  $r = 100$ , according to Eqs. (3.36) and (3.37), set  $l_1 = 101$ ,  $l_2 = 101$ ,  $D_2 = D_4 = 1.0$  and  $D_1 = D_3 = 101$ . Use controller (3.43), choose  $c_1 = 1000$ ,  $c_2 = 300$ ,  $c_3 = 30$ , and  $\eta = 1.5$ . Simulation results are shown in Figs. 3.12 and 3.13.



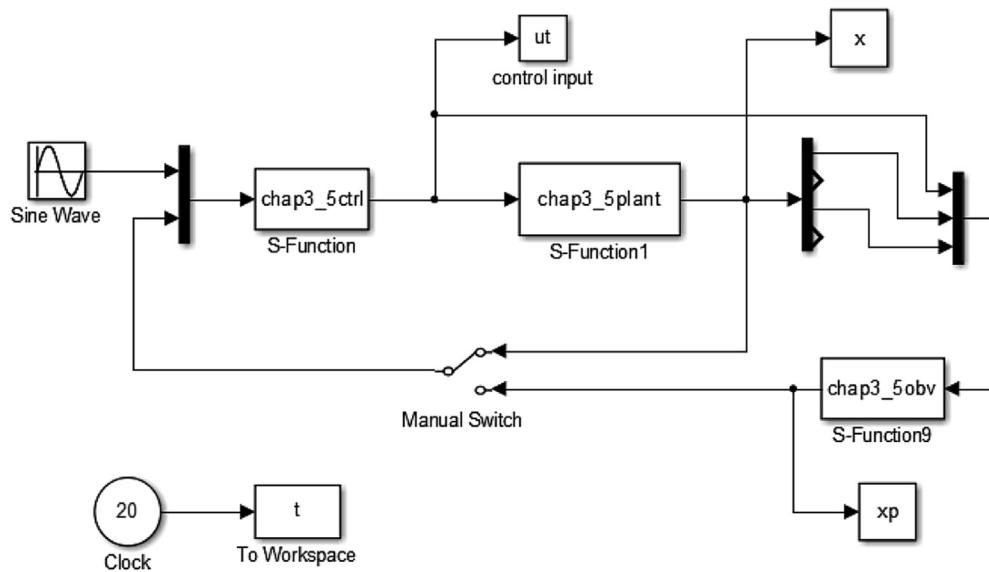
■ FIGURE 3.12 Angle and angle speed tracking.



■ FIGURE 3.13 States estimation.

#### Simulink programs

##### 1. Simulink main program: chap3\_5sim.mdl



**2. S function of controller: chap3\_5ctrl.m**

```
function [sys,x0,str,ts] = controller(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
I=1.0;J=1.0;Mg1=5.0;K=40;
a1=K/I;a2=1/J;

xd=u(1);
dxd=cos(t);
ddxd=-sin(t);
dddxd=-cos(t);
ddddxd=sin(t);

x1p=u(2);
x2p=u(3);
x3p=u(4);
x4p=u(5);

f1p = - 1/I*Mg1*sin(x1p) - K/I*x1p;
dx2p = a1*x3p + f1p;
df1p = - 1/I*Mg1*cos(x1p)*x2p - K/I*x2p;
```

```

ddf1p = - 1/I*Mg1*(- sin(x1p)*x2p^2 + cos(x1p)*dx2p)
- K/I*dx2p;
f2p = 1/J*K*(x1p - x3p);

c1 = 1000; c2 = 300; c3 = 30;
e1p = x1p - xd;
e2p = x2p - dxd;
e3p = a1*x3p + f1p - ddxsd;
e4p = a1*x4p + df1p - dddxd;

sp = c1*e1p + c2*e2p + c3*e3p + e4p;

xite = 1.50;
ut = - 1/(a1*a2)*(c1*(x2p - dxd) + c2*
(a1*x3p + f1p - ddxsd) + c3*(a1*x4p + df1p - dddxd) +
a1*f2p + ddf1p - dddxd + xite*sp);

sys(1) = ut;
3. S function for observer: chap3_5obv.m
function [sys,x0,str,ts] = obv(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
 sys = mdlDerivatives(t,x,u);
case 3,
 sys = mdlOutputs(t,x,u);
case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0;0;0;0];

```

```

str = [];
ts = [0 0];
function sys=mdlDerivatives(t,x,u)
I=1.0;J=1.0;Mgl=5.0;K=40;
r=100;
l1=101;l2=101;
D1=101;D3=101;
D2=1;D4=1;
D2_bar=D2+l1*D1;
D4_bar=D4+l2*D3;

ut=u(1);
x1=u(2);
x3=u(3);
a1=K/I;a2=1/J;
f1=-Mgl/I*sin(x1)-K/I*x1;
f2=K/J*(x1-x3);
sys(1)=x(2)+l1*(x1-x(1))+D1*(x1-x(1));
sys(2)=a1*x3+f1+D2_bar*(x1-x(1));
sys(3)=x(4)+l2*(x3-x(3))+D3*(x3-x(3));
sys(4)=a2*ut+f2+D4_bar*(x3-x(3));
function sys=mdlOutputs(t,x,u)
l1=101;l2=101;

ut=u(1);
x1=u(2);
x3=u(3);

sys(1)=x(1);
sys(2)=x(2)+l1*(x1-x(1));
sys(3)=x(3);
sys(4)=x(4)+l2*(x3-x(3));

```

**4. S function for plant: chap3\_5plant.m**

```

function [sys,x0,str,ts]=spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);

```

```

case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0.2;0;0;0];
str = [];
ts = [0 0];
function sys=mdlDerivatives(t,x,u)
I=1.0;J=1.0;Mg1=5.0;K=40;
ut=u(1);
sys(1)=x(2);
sys(2)=-(1/I)*(Mg1*sin(x(1))+K*(x(1)-x(3)));
sys(3)=x(4);
sys(4)=(1/J)*(ut+K*(x(1)-x(3)));
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);

```

**5. Plot program: chap3\_5plot.m**

```

close all;

figure(1);
subplot(211);
plot(t,sin(t),t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('angle tracking');
legend('x1','x1d');
subplot(212);
plot(t,cos(t),t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('angle speed tracking');
legend('x2','dx2');

```

```

figure(2);
subplot(411);
plot(t,xp(:,1),'b',t,x(:,1),'r','linewidth',2);
xlabel('t');ylabel('x_1');
legend('x1','x1p');
subplot(412);
plot(t,xp(:,2),'b',t,x(:,2),'r','linewidth',2);
xlabel('t');ylabel('x_2');
legend('x2','x2p');
subplot(413);
plot(t,xp(:,3),'b',t,x(:,3),'r','linewidth',2);
xlabel('t');ylabel('x_3');
legend('x3','x3p');
subplot(414);
plot(t,xp(:,4),'b',t,x(:,4),'r','linewidth',2);
xlabel('t');ylabel('x_4');
legend('x4','x4p');

figure(3);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('ut');

```

## 3.6 SLIDING MODE CONTROL WITH A HIGH GAIN OBSERVER BASED ON SEPARATION PRINCIPLE

### 3.6.1 Separation principle

For a nonlinear system as

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\Phi(\mathbf{y}, \mathbf{u}) \\ \mathbf{y} &= \mathbf{C}\mathbf{x}\end{aligned}, \quad (3.46)$$

the controller is designed as

$$\begin{aligned}\dot{\vartheta} &= \Gamma(\vartheta, \mathbf{x}) \\ \mathbf{u} &= \gamma(\vartheta, \mathbf{x})\end{aligned}. \quad (3.47)$$

System (3.46) and controller (3.47) constitute a closed loop system.

Three assumptions are given as follows [8,9]:

*Assumption 1:* Closed loop control system is asymptotically stable;

*Assumption 2:*  $\Phi(\mathbf{y}, \mathbf{u})$  meet local Lipschitz conditions, and

$$\Phi(0, 0) = 0;$$

*Assumption 3:*  $\Gamma(\vartheta, \mathbf{x})$  and  $\gamma(\vartheta, \mathbf{x})$  meet local Lipschitz conditions,

$$\text{and near the equilibrium point, } \Gamma(0, 0) = 0, \gamma(0, 0) = 0.$$

Under the above three assumptions, for the closed loop system, H.K. Khalil etc. [8,9] proposed the following separation theorem:

According to the closed loop system composed of Eqs. (3.46) and (3.47), if assumptions 1–3 are satisfied, then there exists a small enough  $\varepsilon > 0$ , a high gain observer based closed loop system is asymptotically stable near the equilibrium point.

In the actual applications, we can design closed loop control system based on a high gain observer according to the following three steps: in the first step, design the state feedback controller to guarantee the asymptotic stability of the closed loop system; in the second step, design a high gain observer; and in the third step, in the controller, use the estimation value instead of state values. The closed loop control system is asymptotically stable.

The separation theorem provides the convenience for the stability analysis of closed loop systems. However, the deficiency of the above three assumptions is more stringent, and the applications of the theorem are restricted to a certain extent.

### 3.6.2 Problem statement

The stabilization problem of the inverted pendulum system is a typical problem in the study of nonlinear control.

Consider the dynamic system of a single one link inverted pendulum as

$$\ddot{\theta} = \frac{m(m+M)gl}{(M+m)I + Mml^2} \theta - \frac{ml}{(M+m)I + Mml^2} u, \quad (3.48)$$

where the cart mass is  $M$ , the pendulum mass is  $m$  and the pendulum angle is  $\theta$ ,  $I = \frac{1}{12}mL^2$ ,  $l = \frac{1}{2}L$ .

The control task to realize the stabilization control of swing angle  $\theta$  and swing angle speed  $\dot{\theta}$ , i.e., to realize  $\theta \rightarrow 0$  and  $\dot{\theta} \rightarrow 0$  by controller design.

### 3.6.3 High gain observer design

Define  $x_1 = \theta$  and  $x_2 = \dot{\theta}$ ; we then have

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= t_1 x_1 + t_2 u, \end{aligned} \quad (3.49)$$

where  $t_1 = \frac{m(m+M)gl}{(M+m)I + Mml^2}$  and  $t_2 = -\frac{ml}{(M+m)I + Mml^2}$ .

The output is  $\mathbf{y}(t) = \theta$ , then Eq. (3.49) becomes

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{B}\Phi(\mathbf{y}, u), \\ \mathbf{y} &= \mathbf{Cx}\end{aligned}\quad (3.50)$$

where  $\mathbf{x} = [x_1 \ x_2]^T$ ,  $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ ,  $\mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ,  $\mathbf{C} = [1 \ 0]$ ,  $\Phi = t_1 x_1 + t_2 u$ .

For Eq. (3.50), high gain observer can be designed as

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\Phi(\mathbf{y}, u) + \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}), \quad (3.51)$$

where  $\mathbf{L} = \begin{bmatrix} \frac{\alpha_1}{\eta} \\ \frac{\alpha_2}{\eta^2} \end{bmatrix}$ ,  $\eta$  is a very small positive value.

Define  $\tilde{\mathbf{x}} = \hat{\mathbf{x}} - \mathbf{x}$ , combining Eq. (3.51) with  $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{B}\Phi(\mathbf{y}, u)$ , we have  $\dot{\tilde{\mathbf{x}}} = (\mathbf{A} - \mathbf{LC})\tilde{\mathbf{x}}$ , then the stability conditions of the observer is to guarantee  $(\mathbf{A} - \mathbf{LC})$  as Hurwitz.

Then

$$\overline{\mathbf{A}} = \mathbf{A} - \mathbf{LC} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \frac{\alpha_1}{\eta} & 0 \\ \frac{\alpha_2}{\eta^2} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -\frac{\alpha_1}{\eta} & 1 \\ -\frac{\alpha_2}{\eta^2} & 0 \end{bmatrix}.$$

The characteristic equation of  $\overline{\mathbf{A}}$  is

$$|\lambda I - \overline{\mathbf{A}}| = \begin{vmatrix} \lambda + \frac{\alpha_1}{\eta} & -1 \\ \frac{\alpha_2}{\eta^2} & \lambda \end{vmatrix} = \lambda^2 + \frac{\alpha_1}{\eta}\lambda + \frac{\alpha_2}{\eta^2} = 0.$$

From  $(\lambda + k)^2 = 0$ , we have  $\lambda^2 + 2k\lambda + k^2 = 0$ ,  $k > 0$ , then we can design

$$\frac{\alpha_1}{\eta} = 2k, \quad \frac{\alpha_2}{\eta^2} = k^2.$$

In the observer, we can choose  $k = \frac{k_0}{\eta}$ , then  $\alpha_1 = 2k_0$ ,  $\alpha_2 = k_0^2$ .

### 3.6.4 Sliding mode controller design and analysis

For Eq. (3.49), design the sliding mode function as

$$s = cx_1 + x_2,$$

where  $c$  is Hurwitz and  $c > 0$ .

We design the Lyapunov function as

$$V = \frac{1}{2}s^2,$$

then

$$\dot{s}(t) = cx_2 + \dot{x}_2 = cx_2 + t_1x_1 + t_2u.$$

To guarantee  $s\dot{s} < 0$ , design the sliding mode controller as

$$u(t) = \frac{1}{t_2}(-cx_2 - t_1x_1 - \eta s). \quad (3.52)$$

In the controller, the item  $\eta s$  is used instead of  $\eta \text{sgn}(s)$ , so that  $u$  meets the Lipschitz condition.

Then  $\dot{s}(t) = -\eta s$ , and

$$\dot{V} = s\dot{s} = -\eta s^2 = -2\eta V \leq 0.$$

Then we have

$$V(t) = e^{-\eta(t)}V(0).$$

Obviously, by using control law (3.51), the closed loop system is exponentially convergent. The closed loop system can be analyzed as follows:

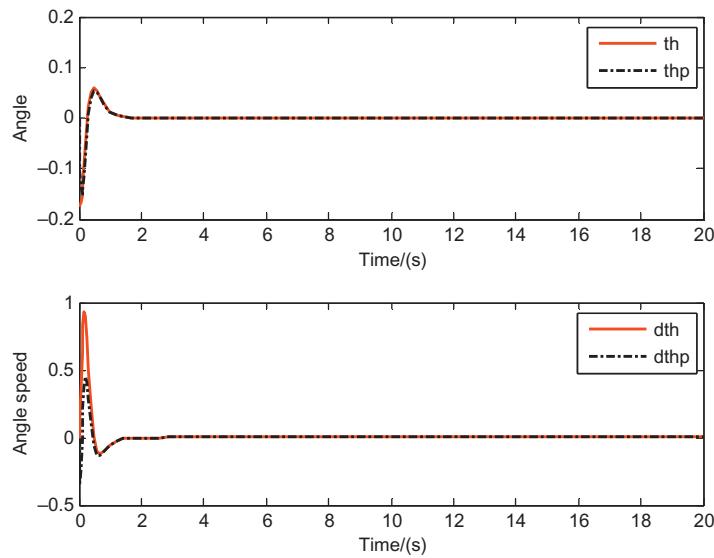
1. The closed loop system composed of control law (3.51) satisfies the asymptotic stability condition, which satisfies Assumption 1;
2. From  $\Phi = t_1x_1 + t_2u$ , we know  $\Phi(\mathbf{y}, \mathbf{u})$  meet Lipschitz, and  $\Phi(0, 0) = 0$ , which satisfies Assumption 2;
3. We have no adaptive law in the design, so  $\Gamma(\vartheta, \mathbf{x}) = 0$ , consider controller (3.52) as  $\gamma(\vartheta, \mathbf{x})$ , we get  $\gamma(0, 0) = 0$ , which satisfies Assumption 3.

In the controller design, we can use the observation values  $\hat{x}_1$  and  $\hat{x}_2$  to replace the actual values  $x_1$  and  $x_2$ , and the measurement of angular velocity is not needed.

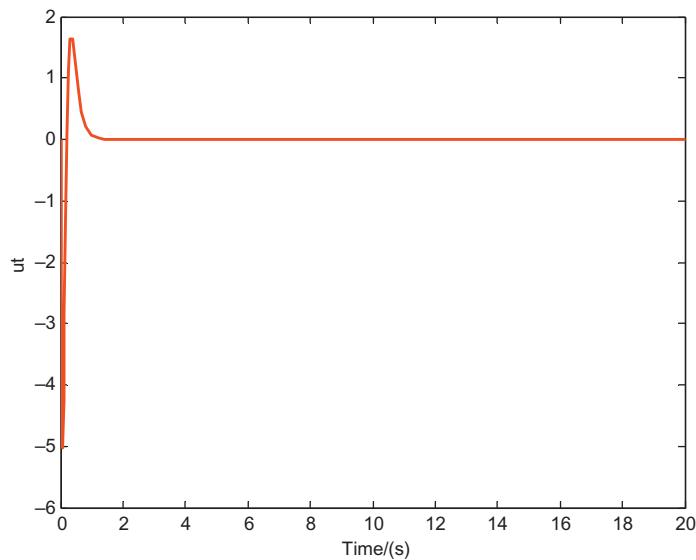
### 3.6.5 Simulation example

Consider the plant as (3.48), and choose  $M = 1$ ,  $m = 0.10$ ,  $L = 0.50$ . The initial states are set as  $\theta(0) = -10^\circ$ ,  $\dot{\theta}(0) = 0$ .

Using controller (3.52) and observer (3.51), set  $c = 10$ ,  $\hat{\mathbf{x}} = [0 \ 0]$ , and choose  $\eta = 0.10$ ,  $k_0 = 1$ . Simulation results are given in Figs. 3.14 and 3.15.



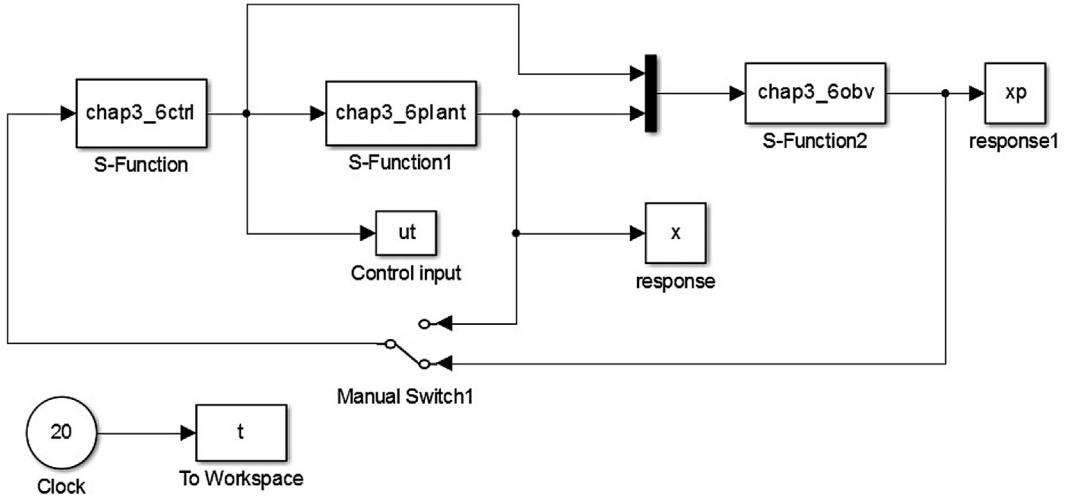
■ FIGURE 3.14 Observation and response of angle and angle speed.



■ FIGURE 3.15 Control input.

Matlab programs:

1. Simulink main program: chap3\_6sim.mdl



2. S function of controller: chap3\_6ctrl.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
```

```

x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
g=9.8;M=1.0;m=0.1;L=0.5;
I=1/12*m*L^2;
l=1/2*L;
t1=m*(M+m)*g*l/[(M+m)*I+M*m*l^2];
t2=-m*l/[(M+m)*I+M*m*l^2];
x1=u(1);%th
x2=u(2);%dth

c=10;
xite=5.0;
s=c*x1+x2;

ut=1/t2*(-c*x2-t1*x1-xite*s);

sys(1)=ut;
3. S function of observer: chap3_6obv.m
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);

```

```

x0=[0;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
g=9.8;M=1.0;m=0.1;L=0.5;
I=1/12*m*L^2;
l=1/2*L;
t1=m*(M+m)*g*[(M+m)*I + M*m*l^2];
t2=-m*l*[(M+m)*I + M*m*l^2];
ut=u(1);
th=u(2);
y=th;
A=[0 1;
 0 0];
B=[0;1];
C=[1 0];
Fai=t1*th+t2*ut;
k0=1;
alpha1= 2*k0;
alpha2= k0^2;
xite=0.10;
L=[alpha1/xite;
 alpha2/(xite^2)];
dx=A*x+B*Fai+L*(y-C*x);

sys(1)=dx(1);
sys(2)=dx(2);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);

```

**4. S function of plant: chap3\_6plant.m**

```

function [sys,x0,str,ts]=spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);

```

```

case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [-10/57.3,0];
str = [];
ts = [0 0];
function sys=mdlDerivatives(t,x,u)
g=9.8;M=1.0;m=0.1;L=0.5;

I=1/12*m*L^2;
l=1/2*L;
t1=m*(M+m)*g*l/[(M+m)*I+M*m*l^2];
t2=-m*l/[(M+m)*I+M*m*l^2];

ut=u(1);
sys(1)=x(2);
sys(2)=t1*x(1)+t2*ut; %ddth
function sys=mdlOutputs(t,x,u)
sys(1)=x(1); %th
sys(2)=x(2);

```

##### 5. Plot program: chap3\_6plot.m

```

close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r',t,xp(:,1)', '-.k','linewidth',2);
legend('th','thp'); xlabel('time/(s)'); ylabel('angle');
subplot(212);
plot(t,x(:,2),'r',t,xp(:,2)', '-.k','linewidth',2);
legend('dth','dthp'); xlabel('time/(s)'); ylabel('angle
speed');

```

```
figure(2);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time/(s)');ylabel('ut');
```

## REFERENCES

- [1] J. Liu, X. Wang, Advanced Sliding Mode Control for Mechanical Systems Design, Analysis and Matlab Simulation, Tsinghua & Springer Press, Beijing, 2011.
- [2] I. Kanellakopoulos, P.V. Kokotovic, A.S. Morse, Adaptive output-feedback control of a class of nonlinear systems, Proceedings of the 30th IEEE Conference on Decision and Control, Brighton, 1991: 1082–1087.
- [3] X. Wang, J. Liu, K.-Y. Cai, Tracking control for a velocity-sensorless VTOL aircraft with delayed outputs, Automatica 45 (2009) 2876–2882.
- [4] A.N. Atassi, H.K. Khalil, Separation results for the stabilization of nonlinear systems using different high-gain observer designs, Syst. Control Lett. 39 (3) (2000) 183–191.
- [5] F. Ghorbel, J.Y. Hung, M.W. Spong, Adaptive control of flexible-joint manipulators, Control Syst. Mag. 9 (7) (1989) 9–13.
- [6] A.C. Huang, Y.C. Chen, Adaptive sliding control for single-link flexible-joint robot with mismatched uncertainties, IEEE Trans. Control Syst. Technol. 12 (5) (2004) 770–775.
- [7] S.J. Yoo, J.B. Park, Y.H. Choi, Output feedback dynamic surface control of flexible-joint robots, Int. J. Control Autom. Syst. 6 (2) (2008) 223–233.
- [8] A.N. Atassi, H.K. Khalil, A separation principle for the stabilization of a class of nonlinear systems, IEEE Trans. Autom. Control 44 (9) (1999) 1672–1687.
- [9] H.K. Khalil, High-gain observers in nonlinear feedback control, International Conference on Control, Automation and Systems 2008, 14–17 October 2008 in COEX, Seoul, Korea.

# Sliding mode control based on disturbance and a delayed observer

Disturbance observer has been used in robotic manipulator control for a long time. In general, the main objective of the use of disturbance observer is to deduce external unknown or uncertain disturbance torques without the use of an additional sensor. In sliding mode control, by using disturbance observer, switch gain can be smaller, thus chattering phenomenon can be decreased greatly.

In practical engineering, measurement signal is often delayed, which can decrease the control performance. In this chapter, two kinds of delayed output observer are introduced, and based on the observer, sliding mode controllers are designed.

In [Section 4.1](#), based on an exponential disturbance observer, a sliding mode controller is designed, closed system stability is analyzed and an example is given. In [Section 4.2](#), consider a measurement delayed linear system, a delayed output observer is introduced, closed system stability is analyzed and exponential convergence is realized, and an example is given. In [Section 4.3](#), consider a measurement delayed nonlinear system, a delayed output observer is introduced, closed system stability is analyzed and asymptotical convergence is realized, and an example is given.

## 4.1 SLIDING MODE CONTROL BASED ON EXPONENTIAL DISTURBANCE OBSERVER

### 4.1.1 System description

Consider a linear system as

$$J\ddot{\theta} + b\dot{\theta} = u + d, \quad (4.1)$$

where  $J$  is moment of inertia,  $b$  is damping coefficient,  $u$  is control input,  $\dot{\theta}$  and  $\ddot{\theta}$  are angle and angle speed, respectively,  $d$  is disturbance,  $J > 0$ ,  $b > 0$ .

### 4.1.2 Disturbance observer design with convergence exponentially

From Eq. (4.1), we have

$$d = J\ddot{\theta} + b\dot{\theta} - u. \quad (4.2)$$

A disturbance observer can be designed as

$$\dot{\hat{d}} = K(d - \hat{d}) = -K\hat{d} + Kd = -K\hat{d} + K(J\ddot{\theta} + b\dot{\theta} - u), \quad (4.3)$$

where  $K > 0$ .

Consider disturbance vary slowly in the control system, we can assume  $\dot{d} = 0$  [1,2]. Define  $\tilde{d} = d - \hat{d}$ , then

$$\dot{\tilde{d}} = -\dot{\hat{d}} = -K(d - \hat{d}) = -K\tilde{d}$$

i.e.,

$$\dot{\tilde{d}} + K\tilde{d} = 0. \quad (4.4)$$

Obviously,  $d$  will converge to  $\hat{d}$  exponentially.

For observer (4.3), angle acceleration signal is needed, which is difficult to measure in practical engineering. Now we introduced a new disturbance observer as follows.

Define  $\dot{\hat{d}} = K(d - \hat{d})$  and auxiliary parameter as [2]

$$z = \hat{d} - KJ\dot{\theta}. \quad (4.5)$$

Then  $\dot{z} = \dot{\hat{d}} - KJ\ddot{\theta}$ , since  $\dot{\hat{d}} = K(d - \hat{d}) = K(J\ddot{\theta} + b\dot{\theta} - u) - K\hat{d}$ , then

$$\dot{z} = K(J\ddot{\theta} + b\dot{\theta} - u) - K\hat{d} - KJ\ddot{\theta} = K(b\dot{\theta} - u) - K\hat{d}.$$

Without angle acceleration signal, disturbance observer can be designed as

$$\begin{cases} \dot{z} = K(b\dot{\theta} - u) - K\hat{d}, \\ \hat{d} = z + KJ\dot{\theta} \end{cases}, \quad (4.6)$$

then

$$\dot{z} = K(b\dot{\theta} - T) - K(z + KJ\dot{\theta}) = K(b\dot{\theta} - T - KJ\dot{\theta}) - Kz.$$

In practical engineering, we always consider  $\dot{d} = 0$  [2], then

$$\tilde{d} = \dot{d} - \dot{\hat{d}} = -\dot{\hat{d}} = -\dot{z} - KJ\ddot{\theta}.$$

Substituting the  $\dot{z}$  item into the above, then

$$\begin{aligned}\dot{\tilde{d}} &= -(K(b\dot{\theta} - T - KJ\dot{\theta}) - Kz) - KJ\ddot{\theta} \\ &= -K(b\dot{\theta} - T - KJ\dot{\theta}) + Kz - KJ\dot{\theta} = K(z + KJ\dot{\theta}) - K(J\ddot{\theta} + b\dot{\theta} - T) \\ &= K\hat{d} - K(J\ddot{\theta} + b\dot{\theta} - T) = K(\hat{d} - d) = -K\tilde{d}\end{aligned}$$

and

$$\dot{\tilde{d}} + K\tilde{d} = 0.$$

The solution is

$$\tilde{d}(t) = \tilde{d}(t_0)e^{-Kt}.$$

Obviously, estimation  $\hat{d}$  converge to  $d$  exponentially, and  $|\tilde{d}(0)| = |\tilde{d}|_{\max}$ .

#### 4.1.3 Sliding mode control

In sliding mode control, to eliminate control chattering, we can design the controller with  $\hat{d}$  instead of  $d$ .

The control goal is  $\theta \rightarrow \theta_d$ ,  $\dot{\theta} \rightarrow \dot{\theta}_d$ .

For Eq. (4.1), design the sliding mode function as

$$s = ce + \dot{e}, \quad (4.7)$$

where  $c > 0$ ,  $e = \theta_d - \theta$ .

Since  $\ddot{\theta} = \frac{1}{J}(-b\dot{\theta} + u + d)$ , then

$$\ddot{e} = \ddot{\theta}_d - \ddot{\theta} = \ddot{\theta}_d - \frac{1}{J}(-b\dot{\theta} + u + d)$$

and

$$\dot{s} = c\dot{e} + \ddot{e} = c\dot{e} + \ddot{\theta}_d - \frac{1}{J}(-b\dot{\theta} + u + d) = c\dot{e} + \ddot{\theta}_d + \frac{b}{J}\dot{\theta} - \frac{1}{J}u - \frac{1}{J}d.$$

Design the controller as

$$u(t) = J\left(c\dot{e} + \ddot{\theta}_d + \frac{b}{J}\dot{\theta} - \frac{1}{J}\hat{d} + k_0s + \eta \text{sgn}s\right), \quad (4.8)$$

where  $k_0 > 0$ .

Then

$$\begin{aligned}\dot{s} &= c\dot{e} + \ddot{\theta}_d + \frac{b}{J}\dot{\theta} - \left(c\dot{e} + \ddot{\theta}_d + \frac{b}{J}\dot{\theta} - \frac{1}{J}\hat{d} + k_0s + \eta \text{sgn}s\right) - \frac{1}{J}d \\ &= \frac{1}{J}\hat{d} - k_0s - \eta \text{sgn}s - \frac{1}{J}d = -k_0s - \eta \text{sgn}s - \frac{1}{J}\tilde{d}\end{aligned}$$

The Lyapunov function for the closed system is

$$V = \frac{1}{2}s^2 + \frac{1}{2}\tilde{d}^2,$$

then

$$\dot{V} = s\dot{s} + \tilde{d}\dot{\tilde{d}} = s\left(-k_0s - \eta \text{sgn}s - \frac{1}{J}\tilde{d}\right) - K\tilde{d}^2 = -k_0s^2 - \eta|s| - \frac{1}{J}\tilde{d}s - K\tilde{d}^2,$$

$$\text{where } \eta \geq \frac{1}{J}|\tilde{d}|_{\max}.$$

Consider  $|\tilde{d}(0)| = |\tilde{d}|_{\max}$ , we have  $\eta \geq \frac{1}{J}|\tilde{d}(0)|$ , then

$$\dot{V} \leq -k_0s^2 - K\tilde{d}^2 \leq -k_1\left(\frac{1}{2}s^2 + \frac{1}{2}\tilde{d}^2\right) = -k_1V,$$

where  $k_1 = 2\min\{k_0, K\}$ .

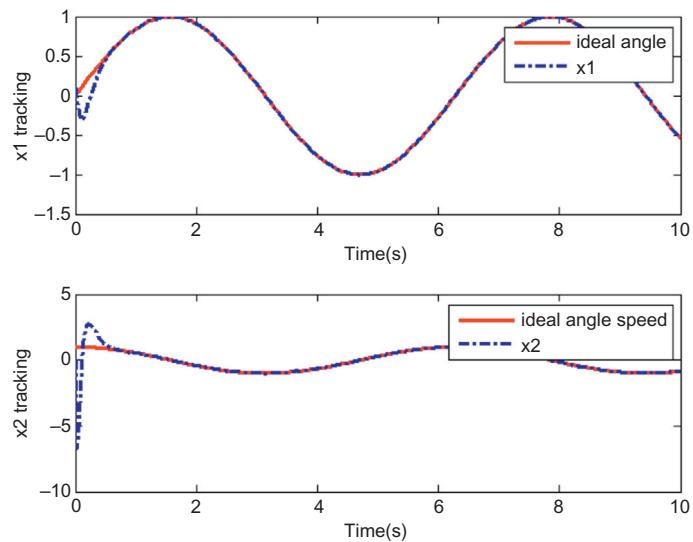
Using Lemma 1.3, the solution of  $\dot{V} \leq -k_1V$  is

$$V(t) \leq e^{-k_1(t-t_0)}V(t_0).$$

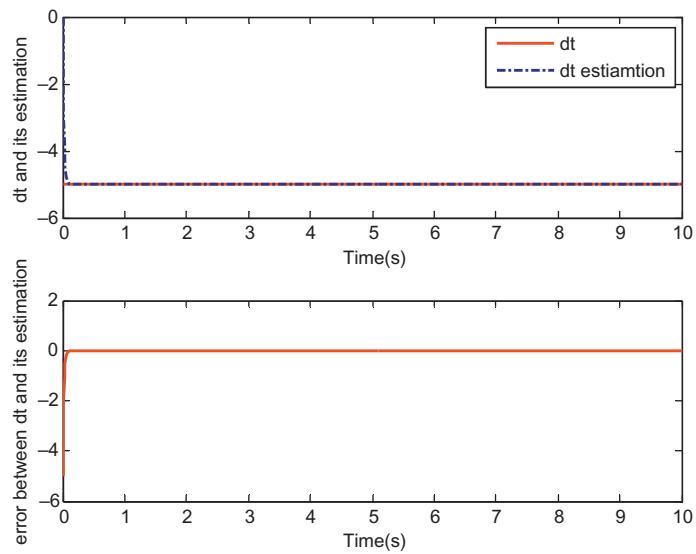
The sliding mode function will converge to zero exponentially.

#### 4.1.4 Simulation example

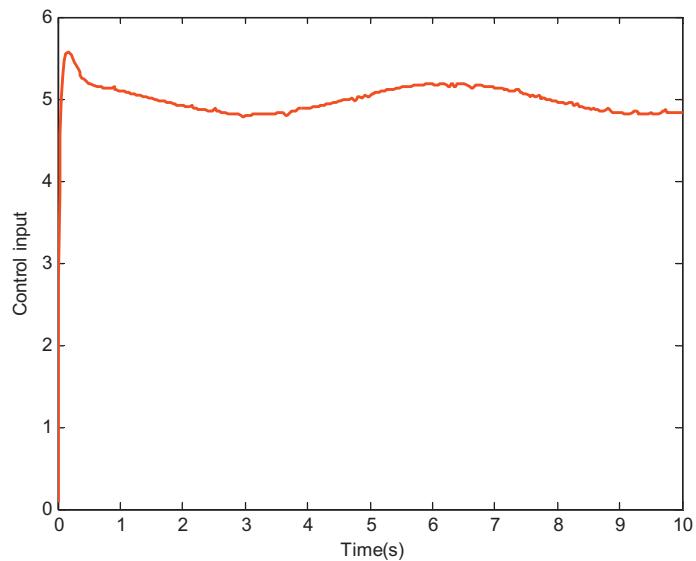
Consider a plant as  $\ddot{\theta} = -25\dot{\theta} + 133(u + d)$ , then we have  $J = \frac{1}{133}$  and  $b = \frac{25}{133}$ . We assume  $d(t) = -5$ , and ideal angle signal as  $\theta_d = \sin t$ , use controller (4.8) with disturbance observer (4.6), choose  $c = 10$ ,  $K = 50$ ,  $\eta = 5.0$ , and use the saturation function instead of the switch function, set  $\Delta = 0.05$ , the simulation results are shown in Figs. 4.1–4.3.



■ FIGURE 4.1 Angle and angle speed tracking.



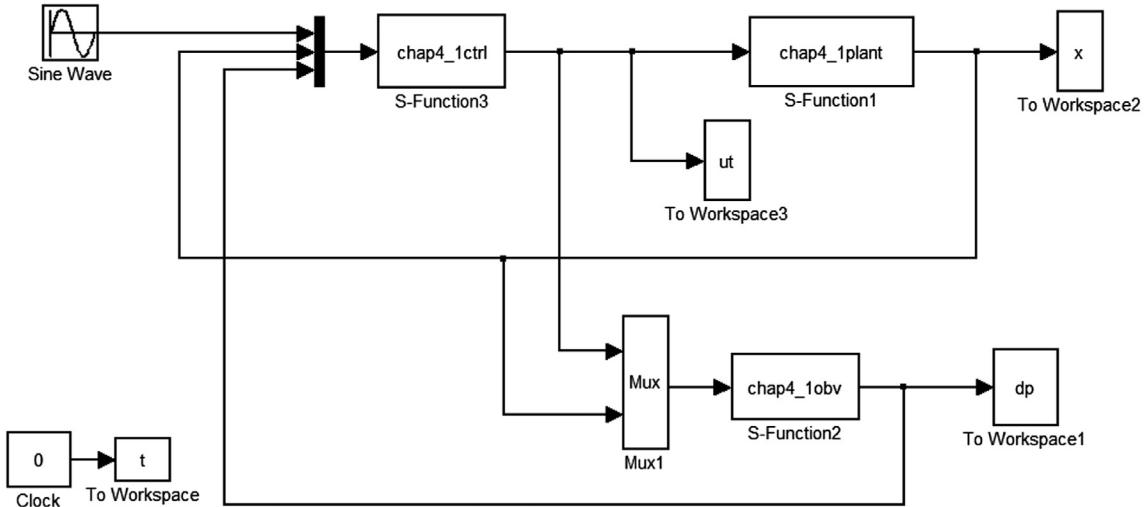
■ FIGURE 4.2 Disturbance estimation.



■ FIGURE 4.3 Control input.

Simulation programs:

(1) Main Simulink: chap4\_1sim.mdl



(2) S function for controller: chap4\_1ctrl.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {1,2,4,9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0 = [];

```

```

str=[];
ts=[-1 0];
function sys=mdlOutputs(t,x,u)
J=1/133;
b=25/133;

thd=u(1);
dthd=cos(t);
ddthd=-sin(t);

th=u(2);
dth=u(3);
dp=u(5);

e=thd-th;
de=dthd-dth;

c=10;
xite=5.0;
s=c*e+de;

%Saturated function
delta=0.05;
kk=1/delta;
if abs(s)>delta
 sats=sign(s);
else
 sats=kk*s;
end

k0=10;
%ut=J*(c*de+ddthd+b/J*dth-1/J*dp+k0*s+xite*sign
%(s));
ut=J*(c*de+ddthd+b/J*dth-1/J*dp+k0*s+xite*sats);

sys(1)=ut;

```

**(3) S function for high gain observer: chap4\_lobv.m**

```

function [sys,x0,str,ts]=NDO(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);

```

```

case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 1;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0];
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
K = 50;
J = 1/133;
b = 25/133;
ut = u(1);
dth = u(3);
z = x(1);
dp = z + K*J*dth;
dz = K*(b*dth-ut)-K*dp;
sys(1)=dz;
function sys=mdlOutputs(t,x,u)
K = 50;
J = 1/133;
dth = u(3);
z = x(1);
dp = z + K*J*dth;
sys(1)=dp;

```

**(4) S function for plant: chap4\_1plant.m**

```

function [sys,x0,str,ts]= NDO_plant (t,x,u,flag)
switch flag,
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;

```

```

case 1,
 sys = mdlDerivatives(t,x,u);
case 3,
 sys = mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.1,0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
dt = - 5;
%dt=0.05*sin(t);
sys(1)=x(2);
sys(2) = - 25*x(2)+133*(ut+dt);
function sys=mdlOutputs(t,x,u)
dt = - 5;
%dt=0.05*sin(t);
sys(1)=x(1);
sys(2)=x(2);
sys(3)=dt;

```

**(5) Plot program: chap4\_1plot.m**

```

close all;
figure(1);
subplot(211);
plot(t,sin(t),'r',t,x(:,1),'-.b','linewidth',2);
xlabel('time(s)');ylabel('x1 tracking');
legend('ideal angle','x1');

```

```

 subplot(212);
 plot(t,cos(t),'r',t,x(:,2),'-.b','linewidth',2);
 xlabel('time(s)');ylabel('x2 tracking');
 legend('ideal angle speed','x2');

 figure(2);
 subplot(211);
 plot(t,x(:,3),'r',t,dp(:,1),'-.b','linewidth',2);
 xlabel('time(s)');ylabel('dt and its estimation');
 legend('dt','dt estiamtion');
 subplot(212);
 plot(t,x(:,3)-dp(:,1),'r','linewidth',2);
 xlabel('time(s)');ylabel('error between dt and its
 estimation');

 figure(3);
 plot(t,ut(:,1),'r','linewidth',2);
 xlabel('time(s)');ylabel('Control input');

```

## 4.2 SLIDING MODE CONTROL BASED ON DELAYED OUTPUT OBSERVER

### 4.2.1 System description

Consider a plant as

$$G(s) = \frac{k}{s^2 + as + b}. \quad (4.9)$$

Eq. (4.9) can be written as

$$\ddot{\theta} = -a\dot{\theta} - b\theta + ku(t), \quad (4.10)$$

where  $\theta(t)$  is the angle signal and  $u(t)$  is the control input.

Define the output as  $z = [\theta \quad \dot{\theta}]^T$ , then Eq. (4.10) can be expressed as

$$\dot{z}(t) = Az(t) + Hu(t), \quad (4.11)$$

$$\text{where } A = \begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix}, H = [0 \quad k]^T.$$

Assuming  $\Delta$  is the time delay brought from measurement delay, the measured output is

$$y(t) = \theta(t - \Delta). \quad (4.12)$$

The goals of the observer are that  $\hat{\theta}(t) \rightarrow \theta(t)$  and  $\hat{\dot{\theta}}(t) \rightarrow \dot{\theta}(t)$  as  $t \rightarrow \infty$ .

### 4.2.2 Delayed output observer design

**Theorem 4.1:** [3,4]: Consider a linear system

$$\dot{x}(t) = Ax(t) + Bx(t - \Delta). \quad (4.13)$$

The stability of the above system can be guaranteed only if characteristic roots' real parts of

$$sI - A - Be^{-\Delta s} = 0 \quad (4.14)$$

are negative, then  $x(t)$  tend to zero exponentially.

For system (4.11), define  $\hat{z} = [\hat{\theta} \quad \dot{\hat{\theta}}]^T$  and design the output delay observer as

$$\dot{\hat{z}}(t) = A\hat{z}(t) + Hu(t) + K[\bar{y}(t) - C\hat{z}(t - \Delta)], \quad (4.15)$$

where  $\hat{z}(t - \Delta)$  is delay of  $\hat{z}(t)$ ,  $C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ .

From Eq. (4.11) to Eq. (4.15), we have

$$\dot{\delta}(t) = A\delta(t) - KC\delta(t - \Delta) \quad (4.16)$$

where  $\delta(t) = z(t) - \hat{z}(t)$ .

According to Theorem 4.1, we can firstly choose  $K$  for Eq. (4.16), and the exponential convergence of  $\delta(t)$  can be realized only if characteristic roots' real parts of

$$sI - A + KCe^{-\Delta s} = 0 \quad (4.17)$$

are in negative half part.

In simulation, we design  $K$  by experience firstly, then use Matlab function “fsolve” to solve the roots of Eq. (4.17), if all the roots are in negative half part, then  $K$  can be validated.

### 4.2.3 Sliding mode control design

The control task is  $\theta \rightarrow \theta_d$  and  $\dot{\theta} \rightarrow \dot{\theta}_d$ . Design the sliding mode function as

$$s = ce + \dot{e}, \quad (4.18)$$

where  $c > 0$ ,  $e = \theta_d - \theta$ .

Design the sliding mode controller

$$u(t) = \frac{1}{k} (\ddot{\theta}_d + a\hat{\dot{\theta}} + b\hat{\theta} + \eta\hat{s} + c\hat{\dot{e}}), \quad (4.19)$$

where  $\eta > 0$ ,  $\hat{e} = \theta_d - \hat{\theta}$ ,  $\hat{s} = c\hat{e} + \hat{\dot{e}}$ .

The Lyapunov function is designed as

$$V = \frac{1}{2}s^2.$$

Since

$$\ddot{e} = \ddot{\theta}_d - \ddot{\theta} = \ddot{\theta}_d + a\dot{\theta} + b\theta - ku$$

$$\dot{s} = c\dot{e} + \ddot{e} = c\dot{e} + \ddot{\theta}_d + a\dot{\theta} + b\theta - ku,$$

then

$$\begin{aligned} \dot{s} &= c\dot{e} + \ddot{\theta}_d + a\dot{\theta} + b\theta - (\ddot{\theta}_d + a\hat{\dot{\theta}} + b\hat{\theta} + \eta\hat{s} + c\hat{\dot{e}}) \\ &= c\dot{e} + a\hat{\dot{\theta}} + b\hat{\theta} - \eta\hat{s} = -\eta s + \eta\hat{s} + c\tilde{e} + a\tilde{\theta} + b\tilde{\theta} \\ &= -\eta s + \eta(-c\tilde{\theta} - \tilde{\theta}) + c(-\tilde{\theta}) + a\tilde{\theta} + b\tilde{\theta} \quad , \\ &= -\eta s + \eta(-c\tilde{\theta} - \tilde{\theta}) + c(-\tilde{\theta}) + a\tilde{\theta} + b\tilde{\theta} \\ &= -\eta s + (b - \eta c)\tilde{\theta} + (a - \eta - c)\tilde{\theta} \end{aligned}$$

where  $\tilde{\theta} = \theta - \hat{\theta}$ ,  $\tilde{\dot{\theta}} = \dot{\theta} - \hat{\dot{\theta}}$ ,  $\tilde{e} = e - \hat{e} = -\theta + \hat{\theta} = -\tilde{\theta}$ ,  $\tilde{\dot{e}} = -\tilde{\dot{\theta}}$ .  
 $\tilde{s} = s - \hat{s} = c\tilde{e} + \tilde{e} = -c\tilde{\theta} - \tilde{\theta}$ .

Then

$$\begin{aligned} \dot{V} &= -\eta s^2 + s((b - \eta c)\tilde{\theta} + (a - \eta - c)\tilde{\dot{\theta}}) \\ &= -\eta s^2 + k_1 s \tilde{\theta} + k_2 s \tilde{\dot{\theta}} \end{aligned}$$

where  $k_1 = b - \eta c$ ,  $k_2 = a - \eta - c$ .

Since  $k_1 s \tilde{\theta} \leq \frac{1}{2} s^2 + \frac{1}{2} k_1^2 \tilde{\theta}^2$ ,  $k_2 s \tilde{\dot{\theta}} \leq \frac{1}{2} s^2 + \frac{1}{2} k_2^2 \tilde{\dot{\theta}}^2$ , then

$$\dot{V} \leq -\eta s^2 + \frac{1}{2} s^2 + \frac{1}{2} k_1^2 \tilde{\theta}^2 + \frac{1}{2} s^2 + \frac{1}{2} k_2^2 \tilde{\dot{\theta}}^2 = -(\eta - 1)s^2 + \frac{1}{2} k_1^2 \tilde{\theta}^2 + \frac{1}{2} k_2^2 \tilde{\dot{\theta}}^2,$$

where  $\eta > 1$ .

Consider exponential convergence of  $\tilde{\theta}$  and  $\tilde{\dot{\theta}}$ , using Theorem 4.1, we have  $\frac{1}{2} k_1^2 \tilde{\theta}^2 + \frac{1}{2} k_2^2 \tilde{\dot{\theta}}^2 \leq \chi(\bullet) e^{-\sigma_0(t-t_0)}$ , then

$$\dot{V} \leq -\eta_1 V + \chi(\bullet) e^{-\sigma_0(t-t_0)} \leq -\eta_1 V + \varepsilon_0,$$

where  $\sigma_0 > 0$ ,  $\eta_1 = 2(\eta - 1) > 0$ ,  $\chi(\bullet)$  is K class function of  $\|\tilde{z}(t_0)\|$ ,  
 $z = [\theta \quad \dot{\theta}]^T$ ,  $\varepsilon_0 = \chi(\bullet)e^{-\sigma_0(t-t_0)}$  is a constant value related to  $\|\tilde{z}(t_0)\|$ .

Using Lemma 1.3, the solution of  $\dot{V} \leq -\eta_1 V + \varepsilon_0$  is

$$\begin{aligned} V(t) &\leq e^{-\eta_1(t-t_0)}V(t_0) + \varepsilon_0 e^{-\eta_1 t} \int_{t_0}^t e^{\eta_1 \tau} d\tau = e^{-\eta_1(t-t_0)}V(t_0) + \frac{\varepsilon_0 e^{-\eta_1 t}}{\eta_1} (e^{\eta_1 t} - e^{\eta_1 t_0}) \\ &= e^{-\eta_1(t-t_0)}V(t_0) + \frac{\varepsilon_0}{\eta_1} (1 - e^{-\eta_1(t-t_0)}) \end{aligned}$$

i.e.,

$$\lim_{t \rightarrow \infty} V(t) \leq \frac{1}{\eta_1} \varepsilon_0.$$

Then  $V(t)$  will tend to zero exponentially.

#### 4.2.4 Simulation example

Consider a plant as

$$G(s) = \frac{1}{s^2 + 10s + 1},$$

i.e.,

$$\ddot{\theta} = -10\dot{\theta} - \theta + u(t).$$

Let  $\Delta = 3.0$ , in the observer, set  $K = [0.1 \quad 0.1]$ , the solution of Eq. (4.17) is  $s = -0.3661$ , according to Theorem 4.1, the stability of the observer can be guaranteed.

Let the input be  $u(t) = \sin t$ , and set the initial states as  $[0.20 \quad 0]$ , use delayed output observer (4.15), set  $\hat{z}(t - \Delta) = [0 \quad 0]^T$ , the simulation results are shown in Figs. 4.4 and 4.5.

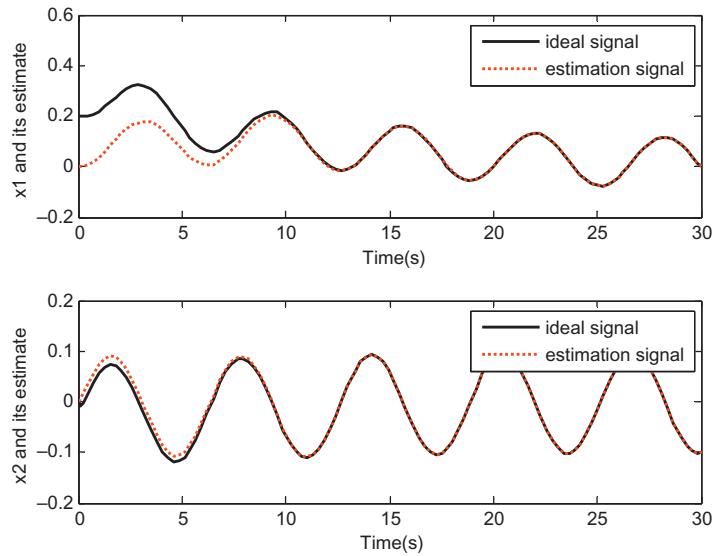
Use controller (4.19) and set  $c = 10$ ,  $k = 1$  and  $\eta = 15$ . The simulation results are shown in Figs. 4.6 and 4.7.

Simulation Programs:

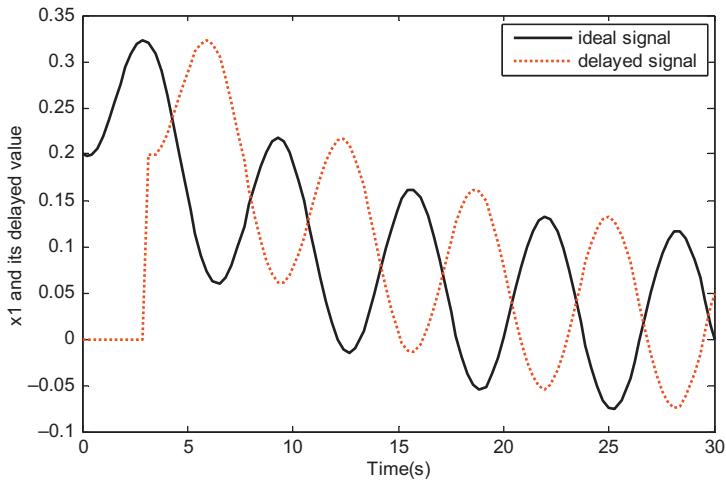
##### 1. K Validation of delayed output observer

###### (1) Main program of K design: design\_K.m

```
close all;
x0 = 0;
options = foptions;
options(1) = 1;
x = fsolve('fun_x', x0, options)
```



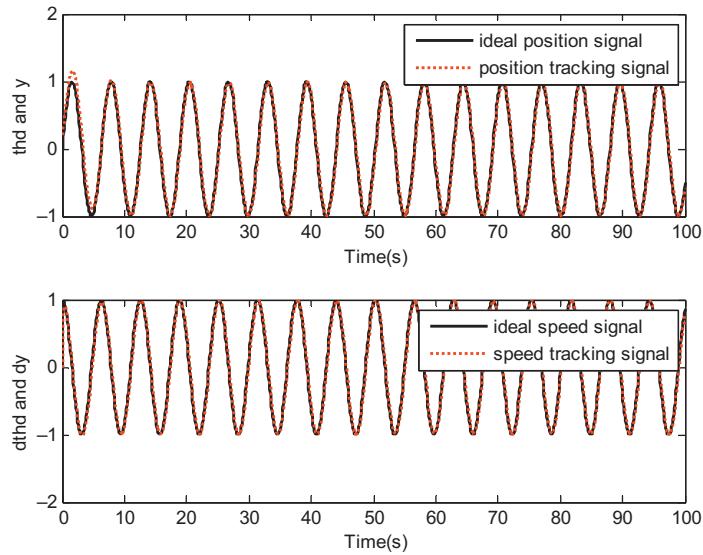
■ FIGURE 4.4  $\theta(t)$ ,  $\dot{\theta}(t)$  and their estimation.



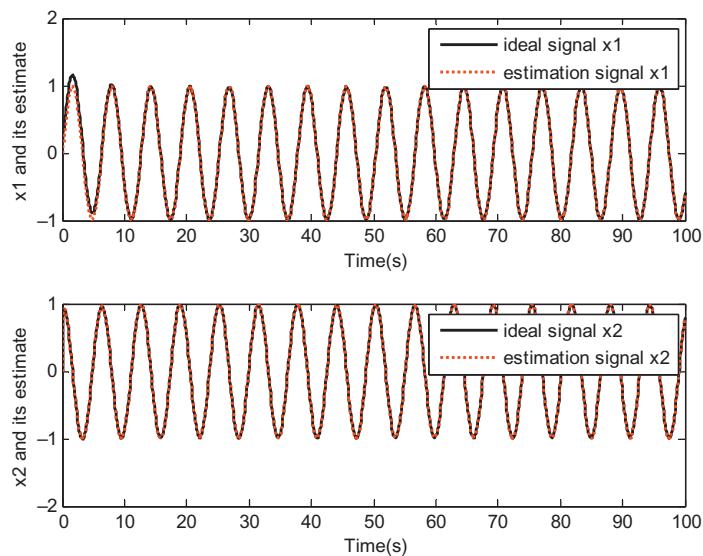
■ FIGURE 4.5 Ideal  $\theta(t)$  and its delay output  $y(t)$ .

(2) Sub program of K design: fun\_x.m

```
function F = fun(x)
tol = 3;
k1 = 0.1;k2 = 0.1;
```



■ FIGURE 4.6 Angle tracking and angle speed tracking.



■ FIGURE 4.7 Ideal value and estimation value of  $\theta(t)$  and  $\dot{\theta}(t)$ .

```

K=[k1,k2]';
C=[1,0];
A=[0 1;-1 -10];

```

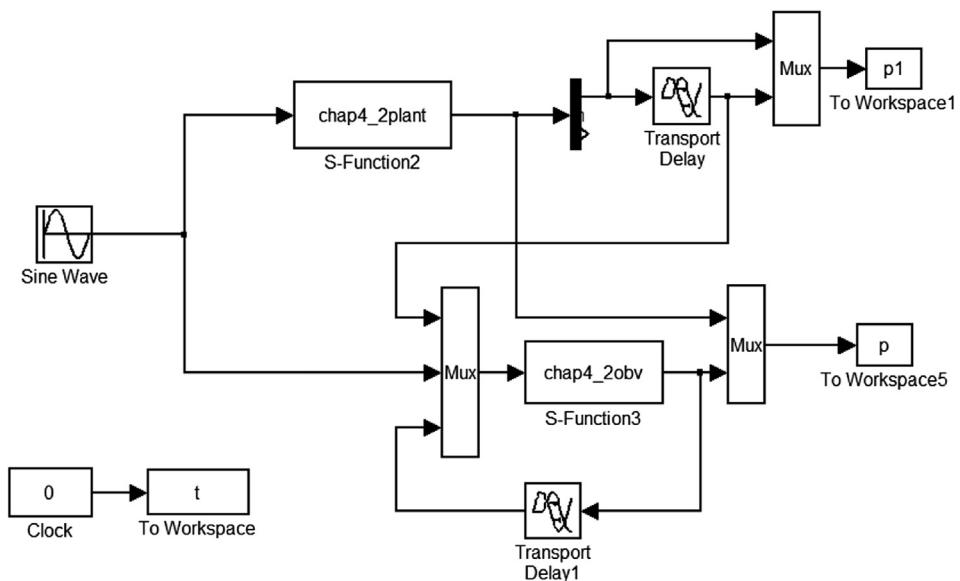
```

F=det(x*eye(2)-A+K*C*exp(-tol*x));

```

**2. Programs for delayed output observer**

**(1) Main program: chap4\_2sim.mdl**



**(2) S function of plant: chap4\_2plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end

```

```
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[0.2 0];
str=[];
ts=[-1 0];
function sys=mdlDerivatives(t,x,u)
sys(1)=x(2);
sys(2)=-10*x(2)-x(1)+u(1);
function sys=mdlOutputs(t,x,u)
th=x(1);w=x(2);

sys(1)=th;
sys(2)=w;
(3) S function of observer: chap4_2obv.m
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
```

```

sys=simsizes(sizes);
x0=[0 0];
str=[];
ts=[-1 0];
function sys=mdlDerivatives(t,x,u)
tol=3;
th_tol=u(1);
yp=th_tol;
ut=u(2);
z_tol=[u(3);u(4)];
thp=x(1);wp=x(2);
%%%%%%%%%%%%%
A=[0 1; -1 -10];
C=[1 0];
H=[0;1];
k1=0.1;k2=0.1; %Verify by design_K.m
z=[thp wp]';
%%%%%%%%%%%%%
K=[k1 k2]';
dz=A*z+H*ut+K*(yp-C*z_tol);
for i=1:2
 sys(i)=dz(i);
end
function sys=mdlOutputs(t,x,u)
thp=x(1);wp=x(2);
sys(1)=thp;
sys(2)=wp;
(4) Plot program: chap4_2plot.m
close all;
figure(1);
subplot(211);
plot(t,p(:,1),'k',t,p(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x1 and its estimate');
legend('ideal signal','estimation signal');
subplot(212);
plot(t,p(:,2),'k',t,p(:,4),'r','linewidth',2);

```

```

xlabel('time(s)'); ylabel('x2 and its estimate');
legend('ideal signal','estimation signal');

figure(2);
subplot(211);
plot(t,p(:,1)-p(:,3),'r','linewidth',2);
xlabel('time(s)'); ylabel('error of x1 and its
estimate');

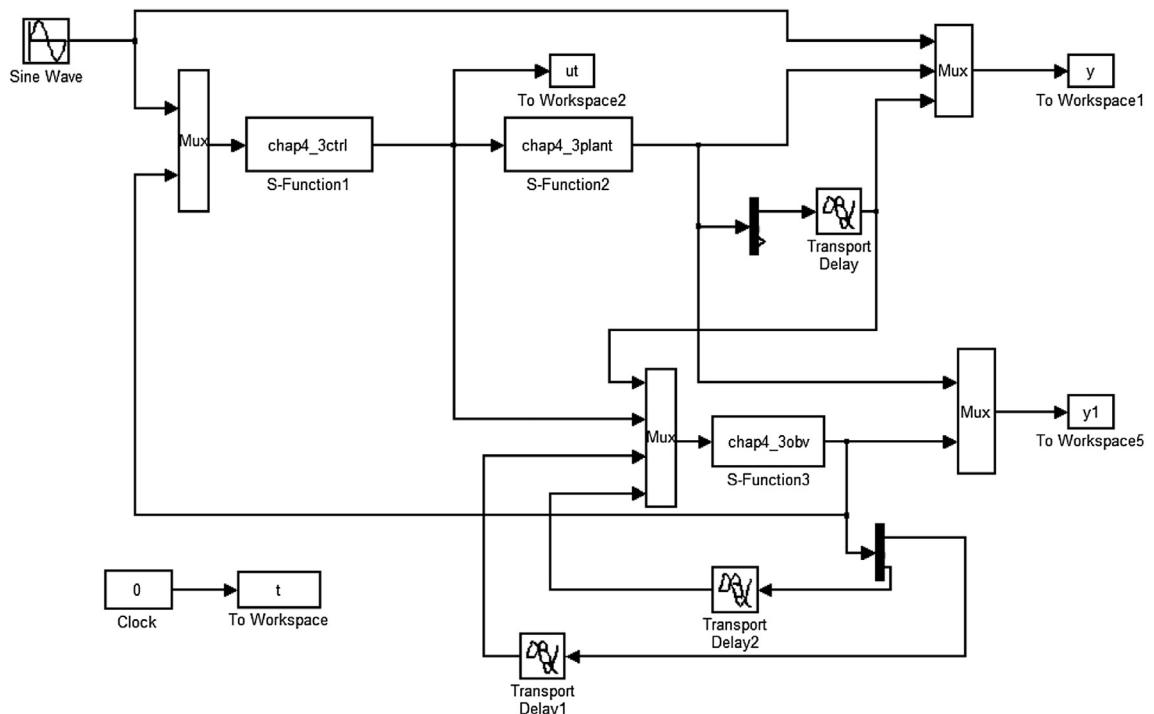
subplot(212);
plot(t,p(:,2)-p(:,4),'r','linewidth',2);
xlabel('time(s)'); ylabel('error of x2 and its
estimate');

figure(3);
plot(t,p1(:,1),'k',t,p1(:,2),'r','linewidth',2);
xlabel('time(s)'); ylabel('x1 and its delayed value');
legend('ideal signal','delayed signal');

```

### 3. Programs for sliding mode control system

#### (1) Main program: chap4\_3sim.mdl



(2) S function of controller: chap4\_3ctrl.m

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {1,2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[-1 0];
function sys=mdlOutputs(t,x,u)
tol=3;
thd=sin(t);
wd=cos(t);
ddthd=-sin(t);

thp=u(2);
wp=u(3);

e1p=thd-thp;
e2p=wd-wp;

k=1;a=10;b=1;
c=10;
xite=15;
sp=c*e1p+e2p;
ut=1/k*(ddthd+a*wp+b*thp+xite*sp+c*e2p);

sys(1)=ut;
```

**(3) S function of plant: chap4\_3plant.m**

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[0.2 0];
str=[];
ts=[-1 0];
function sys=mdlDerivatives(t,x,u)
sys(1)=x(2);
sys(2)=-10*x(2)-x(1)+u(1);
function sys=mdlOutputs(t,x,u)
th=x(1);w=x(2);

 sys(1)=th;
 sys(2)=w;
```

**(4) S function of observer: chap4\_3obv.m**

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
```

```

case 3,
 sys = mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[0 0];
str=[];
ts=[-1 0];
function sys=mdlDerivatives(t,x,u)
tol=3;
th_tol=u(1);
yp=th_tol;
ut=u(2);
z_tol=[u(3);u(4)];
thp=x(1);wp=x(2);
%%%%%
A=[0 1; -1 -10];
C=[1 0];
H=[0;1];
k1=0.1;k2=0.1;%Verify by design_K.m
z=[thp wp]';
%%%%%
K=[k1 k2]';
dz=A*z+H*ut+K*(yp-C*z_tol);
for i=1:2
 sys(i)=dz(i);
end

```

```
function sys = mdlOutputs(t,x,u)
thp = x(1);wp = x(2);
sys(1) = thp;
sys(2) = wp;
(5) plot program: chap4_3plot.m
close all;
figure(1);
plot(t,y(:,1),'k',t,y(:,3),'r:','linewidth',2);
xlabel('time(s)');ylabel('thd and y');
legend('ideal position signal','delayed position
signal');

figure(2);
subplot(211);
plot(t,y1(:,1),'k',t,y1(:,3),'r:','linewidth',2);
xlabel('time(s)');ylabel('x1 and its estimate');
legend('ideal signal x1','estimation signal x1');
subplot(212);
plot(t,y1(:,2),'k',t,y1(:,4),'r:','linewidth',2);
xlabel('time(s)');ylabel('x2 and its estimate');
legend('ideal signal x2','estimation signal x2');

figure(3);
subplot(211);
plot(t,y(:,1),'k',t,y(:,2),'r:','linewidth',2);
xlabel('time(s)');ylabel('thd and y');
legend('ideal position signal','position tracking
signal');
subplot(212);
plot(t,cos(t),'k',t,y(:,3),'r:','linewidth',2);
xlabel('time(s)');ylabel('dthd and dy');
legend('ideal speed signal','speed tracking signal');

figure(4);
subplot(211);
plot(t,y(:,1)-y(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('error between thd and y');
legend('position tracking error');
subplot(212);
plot(t,cos(t)-y(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('error between dthd and dy');
legend('speed tracking error');
```

```

figure(5);
plot(t,ut(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('Control input');

```

### 4.3 SLIDING MODE CONTROL BASED ON A TIME VARYING DELAYED OUTPUT OBSERVER

#### 4.3.1 System description

Consider a nonlinear system as

$$\dot{x}(t) = Ax(t) + M(x, t, u), \quad (4.20)$$

where  $x = [x_1 \ x_2]^T$ ,  $x_1$  is position signal,  $u(t)$  is control input,  $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ ,  $M(x, t, u) = B\chi(x, t, u)$ , and  $B = [0 \ 1]^T$ .

Assumption 1:  $M(x, t, u)$  is Lipschitz;

Assumption 2:  $\delta(t)$  is time varying value,  $\delta(t) \in [0, \Delta]$ .

The delayed measurement signal is

$$\bar{y}(t) = x_1(t - \delta) = Cx(t - \delta(t)), \quad (4.21)$$

where  $\delta(t)$  is delayed time and  $C = [1 \ 0]$ .

The observer goals are  $\hat{x}_1(t) \rightarrow x_1(t)$  and  $\hat{x}_2(t) \rightarrow x_2(t)$  as  $t \rightarrow \infty$ .

#### 4.3.2 Delayed output observer design

**Theorem 4.2:** [5]: for system (4.20),  $x_1(t - \delta)$  is measurement signal with delay, delayed output observer can be designed as

$$\dot{\hat{x}}(t) = A\hat{x}(t) + M(\hat{x}, t, u) + K(\bar{y}(t) - C\hat{x}(t - \delta)).$$

For a second nonlinear system, the observer can be expressed as

$$\dot{\hat{x}}_1 = \hat{x}_2 + k_1(x_1(t - \delta) - \hat{x}_1(t - \delta)) \quad \dot{\hat{x}}_2 = \chi(\hat{x}, t, u) + k_2(x_1(t - \delta) - \hat{x}_1(t - \delta)) \quad (4.22)$$

where  $\hat{x}(t - \delta)$  is the delayed signal of  $\hat{x}(t)$ ,  $K = [k_1 \ k_2]^T$ ,  $K$  must be designed to satisfy  $A - KC$  is Hurwitz.

Then the estimation error will converge to zero asymptotically as  $t \rightarrow \infty$ ,  $\hat{x}_1(t) \rightarrow x_1(t)$ ,  $\hat{x}_2(t) \rightarrow x_2(t)$ .

The proof and analysis of the are given in paper [5].

### 4.3.3 $K$ design based on Hurwitz

Since

$$\bar{A} = A - KC = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} [10] = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} k_1 & 0 \\ k_2 & 0 \end{bmatrix} = \begin{bmatrix} -k_1 & 1 \\ -k_2 & 0 \end{bmatrix},$$

the characteristic equation of  $\bar{A}$  is

$$|\lambda I - \bar{A}| = \begin{vmatrix} \lambda + k_1 & 1 \\ -k_2 & \lambda \end{vmatrix} = \lambda^2 + k_1\lambda + k_2 = 0.$$

From  $(\lambda+k)^2 = 0$ , we have  $\lambda^2 + 2k\lambda + k^2 = 0$ . If we design  $k > 0$ ,  $\bar{A}$  can be Hurwitz, then we can design

$$k_1 = 2k, k_2 = k^2. \quad (4.23)$$

### 4.3.4 Simulation example

Consider dynamic equation of one link manipulator as

$$\ddot{x} = -\frac{1}{I}(d\dot{x} + mgl \cos x) + \frac{1}{I}u,$$

i.e.,

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{d}{I}x_2 - \frac{mgl}{I} \cos x_1 + \frac{1}{I}u, \end{aligned}$$

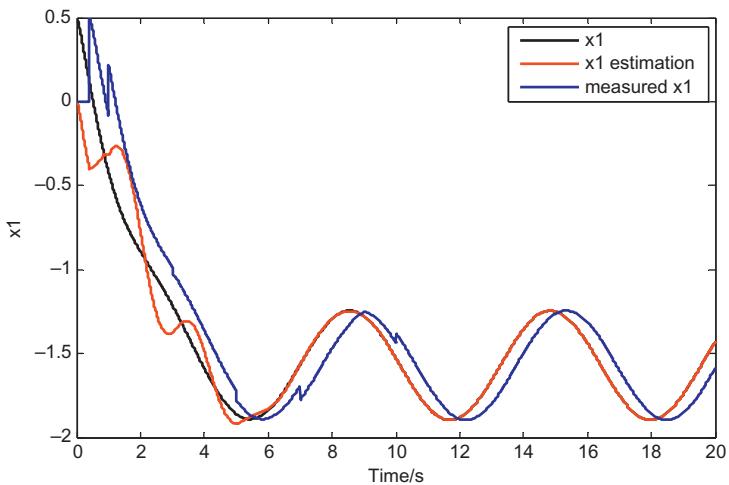
where  $x_1$  and  $x_2$  are angle signal and speed signal,  $u$  is control input,  $g = 9.8 \text{ m/s}^2$ ,  $m = 1$ ,  $l = 0.25$ ,  $d = 2.0$ .

The above equation can be expressed as

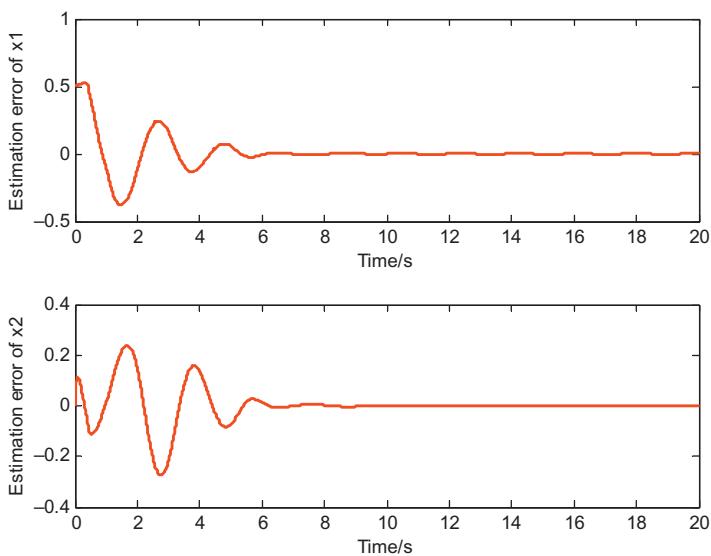
$$\dot{x}(t) = Ax(t) + M(x, t, u),$$

where  $x = [x_1 \ x_2]^T$ ,  $x_1$  is position signal,  $u(t)$  is control input,  $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ ,  $M(x, t, u) = B\chi(x, t, u)$ ,  $B = [0 \ 1]^T$  and  $\chi(x, t, u) = \left[0 \ -\frac{d}{I}x_2 - \frac{mgl}{I} \cos x_1 + \frac{1}{I}u\right]^T$ .

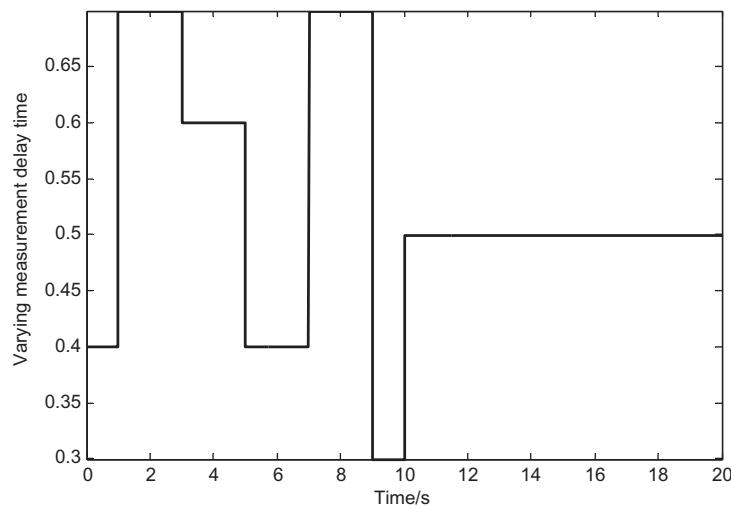
Obviously,  $M(x, t, u)$  is Lipschitz. Delayed time is set as  $\Delta = 1.0$  with  $\delta(t) \in [0, 1.0]$ . In observer (4.22), choose  $k = 1$ ,  $K = [2 \ 1]^T$ , and set  $\hat{x}(t - \delta) = [0 \ 0]^T$ . Use input as  $u(t) = \sin t$ , the initial states of the plant are set as  $[0.500]^T$ . The simulation results are shown in Figs. 4.8–4.10.



■ FIGURE 4.8 Ideal signal, estimation signal, and delayed signal.



■ FIGURE 4.9 Position and speed estimation error.



■ FIGURE 4.10 Time varying measurement delay.

Simulation programs:

(1) S function for delayed signal: chap4\_4delta.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 0;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0 = [];

```

```

str=[];
ts=[-1 0];
function sys=mdlOutputs(t,x,u)
sys(1)=delta(t);

```

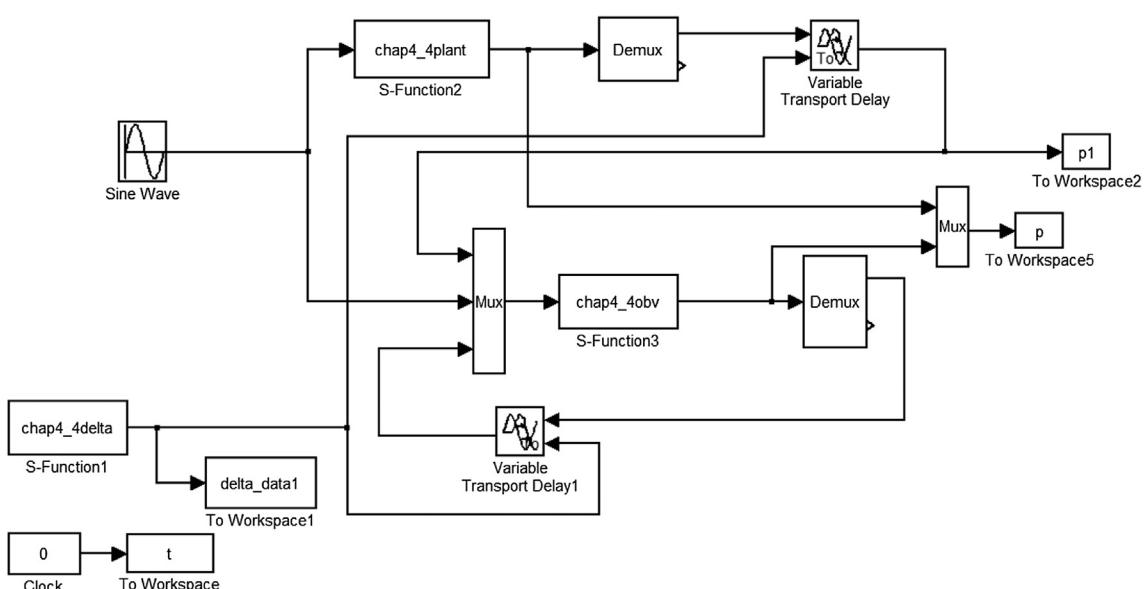
**(2) Delayed signal generation function: delta.m**

```

function [tout]=tolt(tin)
if tin<=1.0
 tout=0.40;
elseif tin<=3.0
 tout=0.70;
elseif tin<=5.0
 tout=0.60;
elseif tin<=7.0
 tout=0.40;
elseif tin<=9.0
 tout=0.70;
elseif tin<=10.0
 tout=0.30;
else
 tout=0.50;
end
tout=tout;

```

**(3) Main program: chap4\_4sim.mdl**



**(4) S function of plant: chap4\_4plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.5 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
g=9.8;m=1;l=0.25;d=2.0;
I=4/3*m*l^2;
sys(1)=x(2);
sys(2)=1/I*(-d*x(2)-m*g*l*cos(x(1)))+1/I*u;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);

```

**(5) S function of observer: chap4\_4obv.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;

```

```
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys=simsizes(sizes);
x0=[0 0];
str=[];
ts=[-1 0];
function sys=mdlDerivatives(t,x,u)
x1p=x(1);
x2p=x(2);

y=u(1);
ut=u(2);
yp=u(3);

k1=2;k2=1;
K=[k1 k2]';

g=9.8;m=1;l=0.25;d=2.0;
I=4/3*m*l^2;

X=1/I*(-d*x2p-m*g*l*cos(x1p))+1/I*ut;

sys(1)=x2p+k1*(y-yp);
sys(2)=X+k2*(y-yp);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
```

(6) Plot program: chap4\_4plot.m

```

close all;
figure(1);
plot(t,p(:,1),'k',t,p(:,3),'r',t,p1
(:,1),'b','linewidth',2);
xlabel('Time/s');ylabel('x1');
legend('x1','x1 estimation','measured x1');

figure(2);
subplot(211);
plot(t,p(:,1)-p(:,3),'r','linewidth',2);
xlabel('Time/s');ylabel('Estimation error of x1');
subplot(212);
plot(t,p(:,2)-p(:,4),'r','linewidth',2);
xlabel('Time/s');ylabel('Estimation error of x2');

figure(3);
plot(t,delta_data1,'k','linewidth',2);
xlabel('Time/s');ylabel('Varying measurement delay
time');

```

### 4.3.5 Sliding mode control based on a delayed output observer

Consider dynamic equation of one link manipulator as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{d}{I}x_2 - \frac{mgl}{I}\cos x_1 + \frac{1}{I}u\end{aligned}\quad (4.24)$$

The control goal is  $x_1 \rightarrow x_{1d}$  and  $x_2 \rightarrow \dot{x}_{1d}$  as  $t \rightarrow \infty$ .

Design the sliding mode function as

$$s = ce + \dot{e},$$

where  $c > 0$ ,  $e = x_1 - x_{1d}$ .

Design the Lyapunov function as

$$V = \frac{1}{2}s^2.$$

Since

$$\ddot{e} = \ddot{x}_1 - \ddot{x}_{1d} = -\frac{d}{I}x_2 - \frac{mgl}{I}\cos x_1 + \frac{1}{I}u - \ddot{x}_{1d},$$

then

$$\dot{s} = c\dot{e} + \ddot{e} = c\dot{e} - \frac{d}{I}x_2 - \frac{mgl}{I}\cos x_1 + \frac{1}{I}u - \ddot{x}_{1d}$$

Design the controller as

$$u(t) = I\left(\frac{d}{I}\hat{x}_2 + \frac{mgl}{I}\cos\hat{x}_1 + \ddot{x}_{1d} - c\hat{e} - \eta\hat{s}\right), \quad (4.25)$$

where  $\eta > 0$ ,  $\hat{e} = \hat{x}_1 - x_{1d}$ ,  $\hat{s} = c\hat{e} + \hat{\dot{e}}$ .

Define  $\tilde{s} = \hat{s} - s$ , then

$$\begin{aligned} \dot{s} &= c\dot{e} + \ddot{e} = c\dot{e} - \frac{d}{I}x_2 - \frac{mgl}{I}\cos x_1 + \frac{1}{I}u - \ddot{x}_{1d} \\ &= c\dot{e} - \frac{d}{I}x_2 - \frac{mgl}{I}\cos x_1 + \frac{d}{I}\hat{x}_2 + \frac{mgl}{I}\cos\hat{x}_1 - c\hat{e} - \eta\hat{s} \\ &= -\eta s - \eta\tilde{s} - c\tilde{e} + \frac{d}{I}\tilde{x}_2 + \frac{mgl}{I}(\cos\hat{x}_1 - \cos x_1) \\ &= -\eta s - \eta(c\tilde{x}_1 + \tilde{x}_2) - c\tilde{x}_2 + \frac{d}{I}\tilde{x}_2 + \frac{mgl}{I}(\cos\hat{x}_1 - \cos x_1) \end{aligned}$$

where  $\tilde{e} = e - \hat{e} = x_1 - \hat{x}_1 = \tilde{x}_1$ ,  $\tilde{\dot{e}} = \dot{e} - \hat{\dot{e}} = x_2 - \hat{x}_2 = \tilde{x}_2$ ,  $\tilde{s} = \hat{s} - s = c\tilde{e} + \tilde{\dot{e}} = c\tilde{x}_1 + \tilde{x}_2$ .

Then

$$\begin{aligned} \dot{V} &= -\eta s^2 + s\left(-\eta(c\tilde{x}_1 + \tilde{x}_2) - c\tilde{x}_2 + \frac{d}{I}\tilde{x}_2 + \frac{mgl}{I}(\cos\hat{x}_1 - \cos x_1)\right) \\ &\leq -\eta s^2 + s\left(-\eta(c\tilde{x}_1 + \tilde{x}_2) - c\tilde{x}_2 + \frac{d}{I}\tilde{x}_2 + \frac{2mgl}{I}\right) \\ &= -\eta s^2 + s\left(O(\tilde{x}_1, \tilde{x}_2) + \frac{2mgl}{I}\right) \end{aligned}$$

where  $O(\tilde{x}_1, \tilde{x}_2) = -\eta(c\tilde{x}_1 + \tilde{x}_2) - c\tilde{x}_2 + \frac{d}{I}\tilde{x}_2$ .

Consider the asymptotical convergence of the observer; we have  $O(\tilde{x}_1, \tilde{x}_2) + \frac{2mgl}{I} \leq O_{\max}$ , then

$$\dot{V} \leq -\eta s^2 + 0.5(s^2 + O_{\max}^2) = -(\eta - 0.5)s^2 + 0.5O_{\max}^2 = -(2\eta - 1)V + 0.5O_{\max}^2$$

Let  $\eta_1 = 2\eta - 1 > 0$ , the solution of  $\dot{V} \leq -\eta_1 V + 0.5O_{\max}^2$  is

$$V(t) \leq e^{-\eta_1 t}V(t_0) + 0.5O_{\max}^2 \int_0^t e^{-\eta_1(t-\tau)} d\tau$$

Since  $\int_0^t e^{-\eta_1(t-\tau)} d\tau = \frac{1}{\eta_1} e^{-\eta_1 t} \int_0^t e^{\eta_1 \tau} d\eta_1 \tau = \frac{1}{\eta_1} e^{-\eta_1 t} e^{\eta_1 t} = \frac{1}{\eta_1}$ , then

$$V(t) \leq e^{-\eta_1 t} V(t_0) + \frac{1}{2\eta_1} O_{\max}^2.$$

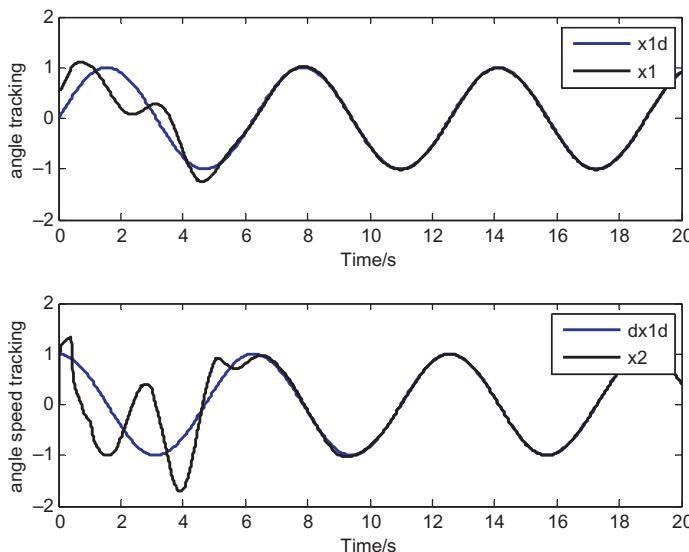
As  $t \rightarrow \infty$ ,  $V(t) \rightarrow \frac{1}{2\eta_1} O_{\max}^2$ , the convergence precision is decided by  $\eta$  and  $O_{\max}$ .

#### 4.3.6 Simulation example

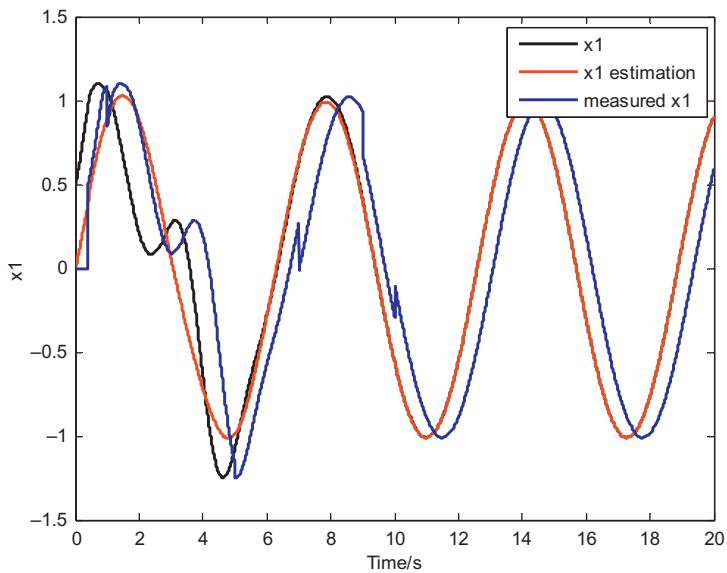
Consider the same plant as Section 4.3.4, the initial states are chosen as  $[0.50 \ 0]^T$ , set  $\Delta = 1.0$  with  $\delta(t) \in [0, 1.0]$ , use controller (4.25) with observer (4.22), set the initial states of the observer as  $\hat{x}(t - \delta) = [0 \ 0]^T$ , and choose  $c = 50$ ,  $\eta = 30$ ,  $k = 1$  and  $K = [2 \ 1]^T$ . The simulation results are given in Figs. 4.11–4.13.

Simulation Programs:

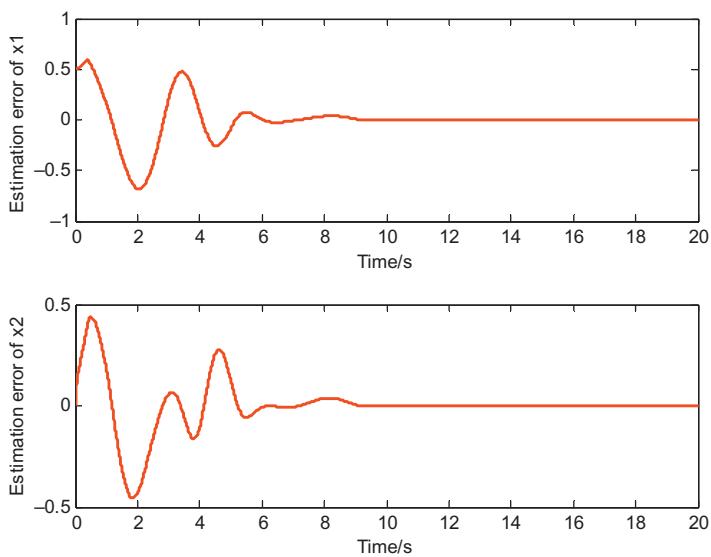
- (1) S function for delayed signal: chap4\_4delta.m
- (2) Delayed signal generation function: delta.m



■ FIGURE 4.11 Position and speed tracking.

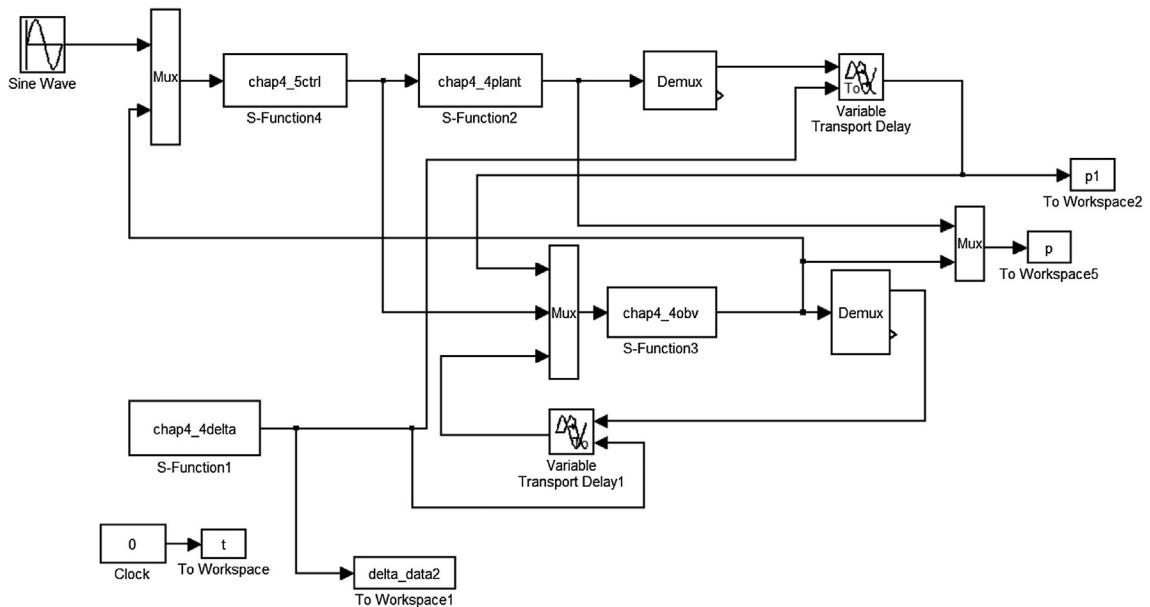


■ FIGURE 4.12 Ideal signal, observed signal, and delayed signal.



■ FIGURE 4.13 Estimation error of position and speed signal.

## (3) Main program: chap4\_5sim.mdl



## (4) S function of controller: chap4\_5ctrl.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {1,2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

```

```

sys=simsizes(sizes);
x0=[];
str=[];
ts=[-1 0];
function sys=mdlOutputs(t,x,u)
x1d=sin(t);
dx1d=cos(t);
ddx1d=-sin(t);

x1p=u(2);
x2p=u(3);

e1p=x1p-x1d;
e2p=x2p-dx1d;

c=50;
xite=30;
sp=c*e1p+e2p;
g=9.8;m=1;l=0.25;d=2.0;I=4/3*m*l^2;

ut=I*(d/I*x2p+m*g*l*cos(x1p)+ddx1d-c*e2p-xite*sp);

sys(1)=ut;
(5) S function of plant: chap4_4plant.m
(6) S function of observer: chap4_4obv.m
(7) Plot program: chap4_5plot.m
 close all;
 figure(1);
 subplot(211);
 plot(t,sin(t),t,p(:,1),'k','linewidth',2);
 xlabel('Time/s');ylabel('angle tracking');
 legend('x1d','x1');
 subplot(212);
 plot(t,cos(t),t,p(:,2),'k','linewidth',2);
 xlabel('Time/s');ylabel('angle speed tracking');
 legend('dx1d','x2');

 figure(2);
 plot(t,p(:,1),'k',t,p(:,3),'r',t,p1
 (:,1),'b','linewidth',2);
 xlabel('Time/s');ylabel('x1');
 legend('x1','x1 estimation','measured x1');

 figure(3);
 subplot(211);
 plot(t,p(:,1)-p(:,3),'r','linewidth',2);

```

```
xlabel('Time/s');ylabel('Estimation error of x1');
subplot(212);
plot(t,p(:,2)-p(:,4),'r','linewidth',2);
xlabel('Time/s');ylabel('Estimation error of x2');
figure(4);
plot(t,delta_data2,'k','linewidth',2);
xlabel('Time/s');ylabel('Varying measurement delay
time');
```

## REFERENCES

- [1] K. Atsuo, I. Hiroshi, S. Kiyoshi, Chattering reduction of disturbance observer based sliding mode control, *IEEE Trans. Ind. Appl.* 30 (2) (1994) 456–461.
- [2] W.H. Chen, D.J. Balance, P.J. Gawthrop, J.O. Reilly, A nonlinear disturbance observer for robotic manipulator, *IEEE Trans. Ind. Electronics* 47 (4) (2000) 932–938.
- [3] S. Leping, Stability criteria for delay differential equations, *J. Shanghai Teachers Univer.* 27 (3) (1998) 1–6.
- [4] C.A. Desoer, M. Vidyasagar, *Feedback System: Input-output Properties*, Academic Press, New York, 1977.
- [5] Q. He, J. Liu, An observer for a velocity-sensorless VTOL aircraft with time-varying measurement delay, *Int. J. Syst. Sci.* 47 (3) (2015) 652–661.

## FURTHER READING

- Q. He, J. Liu, Sliding mode observer for a class of globally Lipschitz nonlinear systems with time-varying delay and noise in its output, *IET Control Theory Appl.* 8 (14) (2014) 1328–1336.

# Sliding mode control based on LMI

In this chapter, six sections about sliding mode controller design on LMI technology are introduced as follows.

In [Section 5.1](#), LMI solution toolbox-YALMIP toolbox is introduced, and an example is given. In [Section 5.2](#), LMI based sliding mode controller design for linear system is introduced, the states response and position tracking exponentially are all realized, and two examples are given. In [Section 5.3](#), consider input disturbance, a LMI based sliding mode controller design for linear system is introduced, the states response and position tracking exponentially are all realized, and two examples are given. In [Section 5.4](#), LMI based sliding mode controller design for a Lipschitz nonlinear system is introduced, the states response exponentially is realized, and an example is given. In [Section 5.5](#), LMI based sliding mode controller design for a Lipschitz nonlinear system is introduced, and position tracking exponentially is realized, and an example is given. In [Section 5.6](#), LMI based sliding mode control for chaotic systems is introduced, dynamic compensation is designed, the states response exponentially is realized, and an example is given.

## 5.1 THE NEW LMI SOLUTION TOOLBOX—YALMIP TOOLBOX

YALMIP is an independent toolbox of Matlab. It is based on symbolic computation, which has a strong ability to optimize and solve the problem such as linear planning, nonlinear planning, mixed planning and LMI problems.

LMI solution can be achieved easily by using YALMIP to solve; you can write the LMI expression directly.

The key integrated commands of YALMIP toolbox are [\[1\]](#):

1. the command “`sdpvar`” is used to express decision variables in optimization problem;
2. the command “`set`” can be used to express all constraint conditions;
3. the command “`solvesdp`” can be used to solve optimization problems;
4. The solution matrix  $X$  can be extracted by  $X = \text{double}(x)$ .

YALMIP toolbox can be downloaded free from internet, the toolbox name is “yalmip.rar,” then you can decompress and set path “yalmip.rar” to Matlab toolbox path with “add with subfolders.”

A typical LMI is

$$\mathbf{A}^T \mathbf{P} + \mathbf{F}^T \mathbf{B}^T \mathbf{P} + \mathbf{P} \mathbf{A} + \mathbf{P} \mathbf{B} \mathbf{F} < 0 \quad (5.1)$$

If  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{P}$  are known, we can use LMI **YALMIP toolbox** to solve  $\mathbf{F}$ .

For example, choose  $\mathbf{A} = \begin{bmatrix} -2.548 & 9.1 & 0 \\ 1 & -1 & 0 \\ 0 & -14.2 & 0 \end{bmatrix}$ ,  $\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  and  
 $\mathbf{P} = \begin{bmatrix} 1,000,000 & 0 & 0 \\ 0 & 1,000,000 & 0 \\ 0 & 0 & 1,000,000 \end{bmatrix}$ , use LMI program chap5\_1.m,  
we can get  $\mathbf{F} = \begin{bmatrix} -492.4768 & -5.05 & 0 \\ -5.05 & -494.0248 & 6.6 \\ 0 & 6.6 & -495.0248 \end{bmatrix}$ .

```
Simulation program: chap5_1.m
clear all;
close all;

%First example on the paper by M.Rehan
A=[-2.548 9.1 0;1 -1 1;0 -14.2 0];
B=[1 0 0;0 1 0;0 0 1];
F=sdpvar(3,3);
M=sdpvar(3,3);
P=1000000*eye(3);

FAI=(A' + F'*B')*P + P*(A+B*F);

%LMI description
L=set(FAI<0);
solvesdp(L);
F=double(F)
```

## 5.2 SLIDING MODE CONTROLLER DESIGN FOR A LINEAR SYSTEM BASED ON LMI

### 5.2.1 System description

Consider the linear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad (5.2)$$

where  $\mathbf{x} \in \mathbb{R}^n$ ,  $\mathbf{u} \in \mathbb{R}^n$ ,  $\mathbf{A} \in \mathbb{R}^{n \times n}$ ,  $\mathbf{B} \in \mathbb{R}^{n \times n}$ .

The control goal is  $\mathbf{x} \rightarrow 0$  exponentially.

### 5.2.2 Linear system stabilization based on LMI

For system (5.1), the controller is designed as

$$\mathbf{u} = \mathbf{F}\mathbf{x}, \quad (5.3)$$

where  $\mathbf{F}$  is state feedback gain, which can be solved by LMI.

**Theorem 5.1:** If the following LMI is satisfied

$$\alpha\mathbf{P} + \mathbf{A}^T\mathbf{P} + \mathbf{M}^T + \mathbf{P}\mathbf{A} + \mathbf{M} < 0, \quad (5.4)$$

where  $\mathbf{F} = (\mathbf{P}\mathbf{B})^{-1}\mathbf{M}$ .

Then the closed system with plant (5.1) and controller (5.2) is exponentially stable.

**Proof:** Choose the Lyapunov function as

$$V = \mathbf{x}^T\mathbf{P}\mathbf{x},$$

where  $\mathbf{P} = \mathbf{P}^T > 0$ .

Then

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{F}\mathbf{x}$$

and

$$\begin{aligned} \alpha V + \dot{V} &= \alpha V + (\mathbf{x}^T\mathbf{P})'\mathbf{x} + \mathbf{x}^T\mathbf{P}\dot{\mathbf{x}} = \alpha V + \dot{\mathbf{x}}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{P}\dot{\mathbf{x}} \\ &= \alpha V + (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{F}\mathbf{x})^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{P}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{F}\mathbf{x}) \\ &= \alpha V + \mathbf{x}^T\mathbf{A}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{F}^T\mathbf{B}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\mathbf{P}\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{P}\mathbf{B}\mathbf{F}\mathbf{x}, \\ &= \alpha V + \mathbf{x}^T(\mathbf{A}^T\mathbf{P} + \mathbf{F}^T\mathbf{B}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{P}\mathbf{B}\mathbf{F})\mathbf{x} \\ &= \alpha\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{x}^T\Omega\mathbf{x} = \mathbf{x}^T(\alpha\mathbf{P} + \Omega)\mathbf{x} \end{aligned}$$

where  $\alpha > 0$ ,  $\Omega = \mathbf{A}^T\mathbf{P} + \mathbf{F}^T\mathbf{B}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{P}\mathbf{B}\mathbf{F}$ .

To guarantee  $\alpha V + \dot{V} \leq 0$ ,  $\alpha\mathbf{P} + \Omega < 0$  is needed, that is

$$\alpha\mathbf{P} + \mathbf{A}^T\mathbf{P} + \mathbf{F}^T\mathbf{B}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{P}\mathbf{B}\mathbf{F} < 0. \quad (5.5)$$

Since  $\mathbf{F}$  and  $\mathbf{P}$  are unknown, to solve Eq. (5.5), the left part must be linearized. Define  $\mathbf{M} = \mathbf{P}\mathbf{B}\mathbf{F}$ , then we have

$$\alpha\mathbf{P} + \mathbf{A}^T\mathbf{P} + \mathbf{M}^T + \mathbf{P}\mathbf{A} + \mathbf{M} < 0. \quad (5.6)$$

From  $\alpha V + \dot{V} \leq 0$ , using Lemma 1.3, we have  $V(t) \leq V(0)\exp(-\alpha t)$ , i.e., if  $t \rightarrow \infty$ ,  $V(t) \rightarrow 0$  and  $\mathbf{x} \rightarrow 0$ , the system states convergence to zero exponentially.

From the LMI YALMIP toolbox, we can get  $\mathbf{M}$  and  $\mathbf{P}$ , then  $\mathbf{F} = (\mathbf{P}\mathbf{B})^{-1}\mathbf{M}$ .

### 5.2.3 Tracking control for linear system based on LMI

The control goal is  $x \rightarrow x_r$ ,  $x_r$  is ideal command. Define the tracking error as  $z = x - x_r$ , then

$$\dot{z} = \dot{x} - \dot{x}_r = Ax + Bu - \dot{x}_r$$

Controller is designed as

$$u = Fx + u_r, \quad (5.7)$$

where  $F$  is state feedback gain, which can be solved by LMI,  $u_r = -Fx_r - B^{-1}Ax_r + B^{-1}\dot{x}_r$ , then

$$u = Fx - Fx_r - B^{-1}Ax_r + B^{-1}\dot{x}_r = Fz - B^{-1}Ax_r + B^{-1}\dot{x}_r$$

and

$$\dot{z} = Ax + B(Fz - B^{-1}Ax_r + B^{-1}\dot{x}_r) - \dot{x}_r = Ax + BFz - Ax_r + \dot{x}_r - \dot{x}_r = Az + BFz$$

**Theorem 5.2:** If the following LMI is satisfied

$$\alpha P + A^T P + M^T + PA + M < 0, \quad (5.8)$$

where  $F = (PB)^{-1}M$ .

Then the closed system with plant (5.2) and controller (5.7) is exponentially stable.

**Proof:** Design the Lyapunov function as

$$V = z^T P z,$$

where  $P = P^T > 0$ , then

$$\begin{aligned} \alpha V + \dot{V} &= \alpha V + (z^T P)' z + z^T P \dot{z} = \alpha V + \dot{z}^T P z + z^T P \dot{z} \\ &= \alpha V + (Az + BFz)^T P x + z^T P (Az + BFz) \\ &= \alpha V + z^T A^T P z + z^T F^T B^T P z + z^T P A z + z^T P B F z, \\ &= \alpha V + z^T (A^T P + F^T B^T P + PA + PBF) z \\ &= \alpha z^T P z + z^T \Omega z = z^T (\alpha P + \Omega) z \end{aligned}$$

where  $\alpha > 0$ ,  $\Omega = A^T P + F^T B^T P + PA + PBF$ .

To guarantee  $\alpha V + \dot{V} \leq 0$ , we need  $\alpha P + \Omega < 0$ , that is

$$\alpha P + A^T P + F^T B^T P + PA + PBF < 0. \quad (5.9)$$

Since  $F$  and  $P$  are unknown, to solve Eq. (5.9), the left part must be linearized, define  $M = PBF$ , then we have

$$\alpha P + \Omega A^T P + M^T + PA + M < 0. \quad (5.10)$$

From  $\alpha V + \dot{V} \leq 0$ , using Lemma 1.3, we have  $V(t) \leq V(0) \exp(-\alpha t)$ , i.e., if  $t \rightarrow \infty$ ,  $V(t) \rightarrow 0$  and  $z \rightarrow 0$ , the tracking error convergence to zero exponentially.

From the LMI YALMIP toolbox, we can get  $\mathbf{M}$  and  $\mathbf{P}$ , then we have  $\mathbf{F} = (\mathbf{PB})^{-1}\mathbf{M}$ .

#### 5.2.4 Simulation example

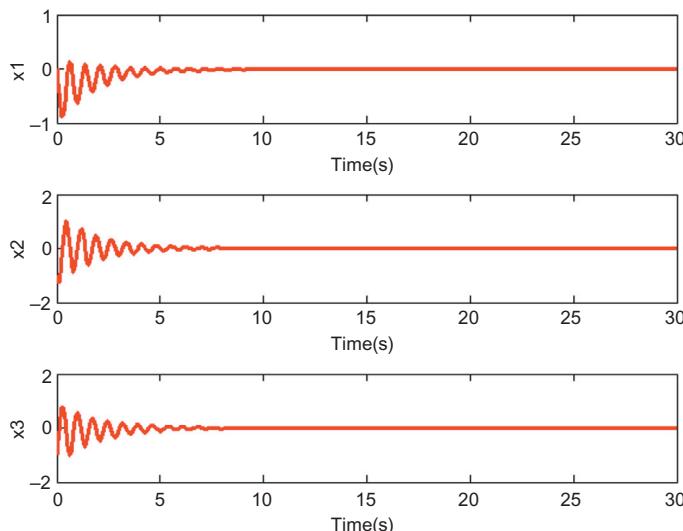
Consider plant (5.2), and choose

$$\mathbf{A} = \begin{bmatrix} -2.548 & 9.1 & 0 \\ 1 & -1 & 0 \\ 0 & -14.2 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

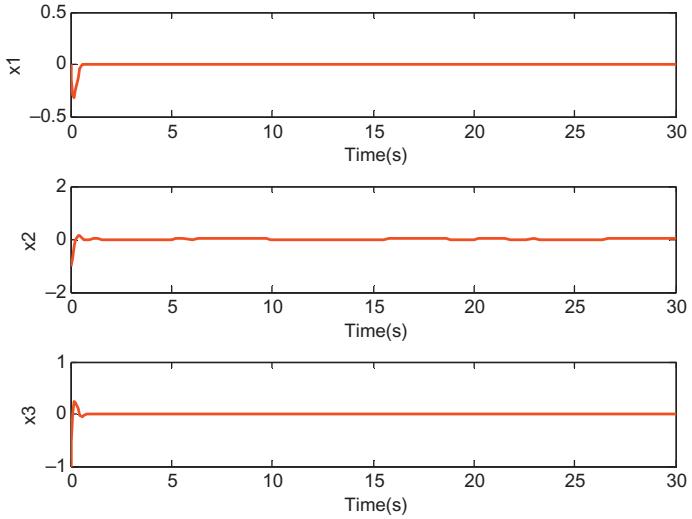
From LMI (5.4), let  $\alpha = 0$  and  $\alpha = 10$ , respectively, we can get

$$\mathbf{F} = \begin{bmatrix} 2.048 & -5.05 & 0 \\ -5.05 & 0.5 & 6.6 \\ 0 & 6.6 & -0.5 \end{bmatrix} \text{ and } \mathbf{F} = \begin{bmatrix} -2.952 & -5.05 & 0 \\ -5.05 & -4.5 & 6.6 \\ 0 & 6.6 & -5.5 \end{bmatrix}$$

respectively, use the controller (5.3), the simulation results are given in Figs. 5.1 and 5.2.



■ FIGURE 5.1 System states response with  $\alpha = 0$ .

■ FIGURE 5.2 System states response with  $\alpha = 10$ .

The ideal states are set as  $[\sin t \cos t \sin t]$ , from LMI (5.8), let  $\alpha = 0$  and  $\alpha = 10$  respectively, we can get  
 $\mathbf{F} = \begin{bmatrix} 2.048 & -5.05 & 0 \\ -5.05 & 0.5 & 6.6 \\ 0 & 6.6 & -0.5 \end{bmatrix}$  and  $\mathbf{F} = \begin{bmatrix} -2.952 & -5.05 & 0 \\ -5.05 & -4.5 & 6.6 \\ 0 & 6.6 & -5.5 \end{bmatrix}$   
respectively, use controller (5.7), the simulation results are given in Figs. 5.3 and 5.4.

Simulation programs:

First simulation: Stabilization programs

1. LMI design program: chap5\_21mi.m

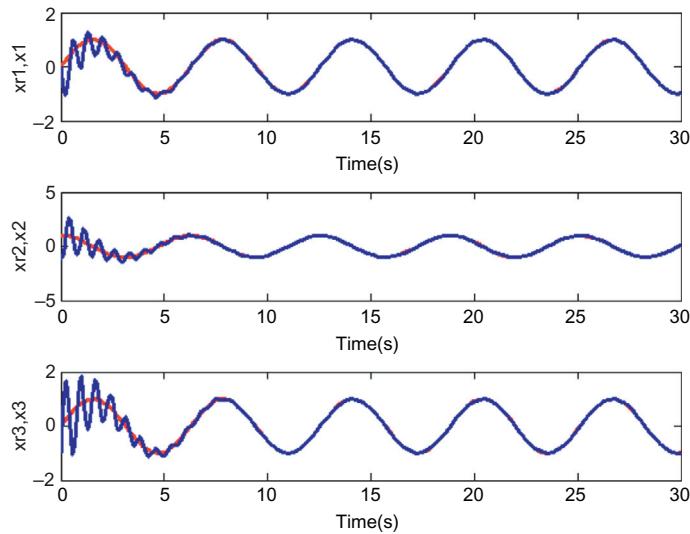
```
clear all;
close all;
```

```
%First example on the paper by M.Rehan
```

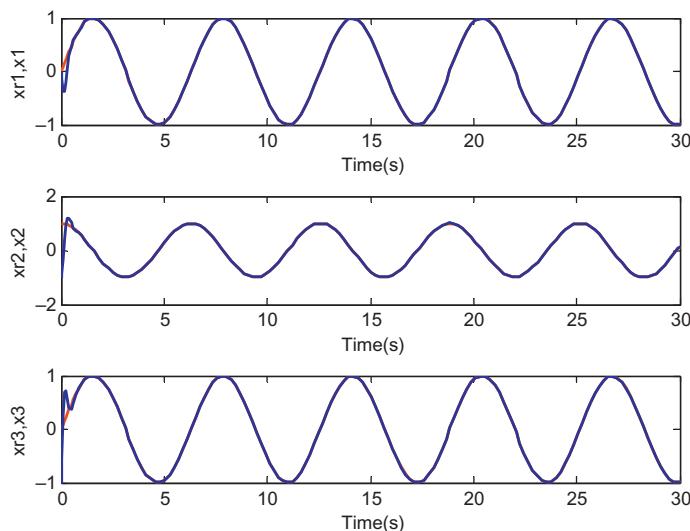
```
A=[-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
```

```
B=[1 0 0;
 0 1 0;
 0 0 1];
```

```
P=sdpvar(3,3);
F=sdpvar(3,3);
M=sdpvar(3,3);
```



■ FIGURE 5.3 System states tracking with  $\alpha = 0$ .



■ FIGURE 5.4 System states tracking with  $\alpha = 10$ .

```
%FAI = (A' + F'*B')*P + P*(A + B*F);
FAI = A'*P + M' + P*A + M; %M = PBF
```

```
%LMI description
```

```
L1=set(P>0);
```

```
alfa=0;
```

```
alfa=10;
```

```
L2=set((FAI + alfa*P)<0);
```

```
LL=L1+L2;
```

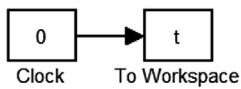
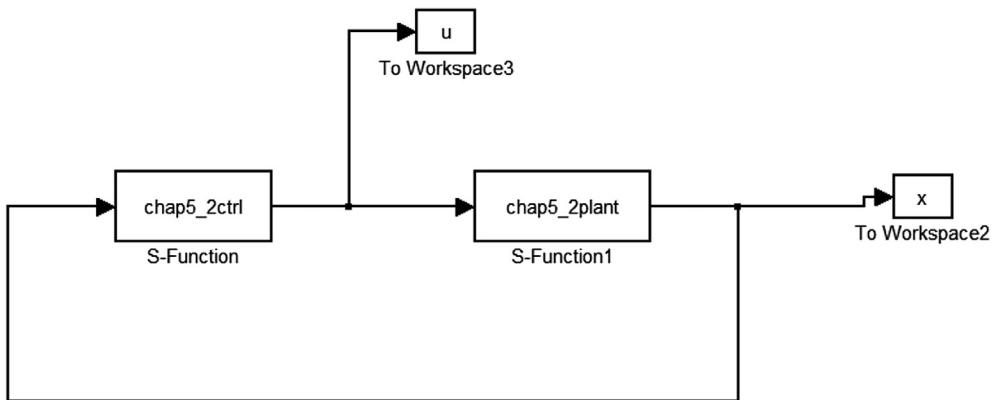
```
solvesdp(LL);
```

```
P=double(P);
```

```
M=double(M)
```

```
F=inv(P*B)*M
```

**2. Simulink main program: chap5\_2sim.mdl**



**3. Controller program: chap5\_2ctrl.m**

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
```

```

case 3,
 sys = mdlOutputs(t,x,u);
case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
x1=u(1);
x2=u(2);
x3=u(3);

alfa=10;
if alfa==0
F=[2.0480 -5.0500 0.0000;
 -5.0500 0.5000 6.6000;
 -0.0000 6.6000 -0.5000];
elseif alfa==10
F=[-2.9520 -5.0500 0.0000;
 -5.0500 -4.5000 6.6000;
 -0.0000 6.6000 -5.5000];
end
ut=F*[x1;x2;x3];

sys(1:3)=ut;

```

**4. Plant program: chap5\_2plant.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
```

```

case 1,
 sys = mdlDerivatives(t,x,u);
case 3,
 sys = mdlOutputs(t,x,u);
case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0, -1, -1];
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
A=[-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
B=[1 0 0;
 0 1 0;
 0 0 1];
ut=[u(1) u(2) u(3)]';
dx=A*x+B*ut;
sys(1)=dx(1);
sys(2)=dx(2);
sys(3)=dx(3);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
5. Plot program: chap5_2plot.m
close all;
figure(1);

```

```

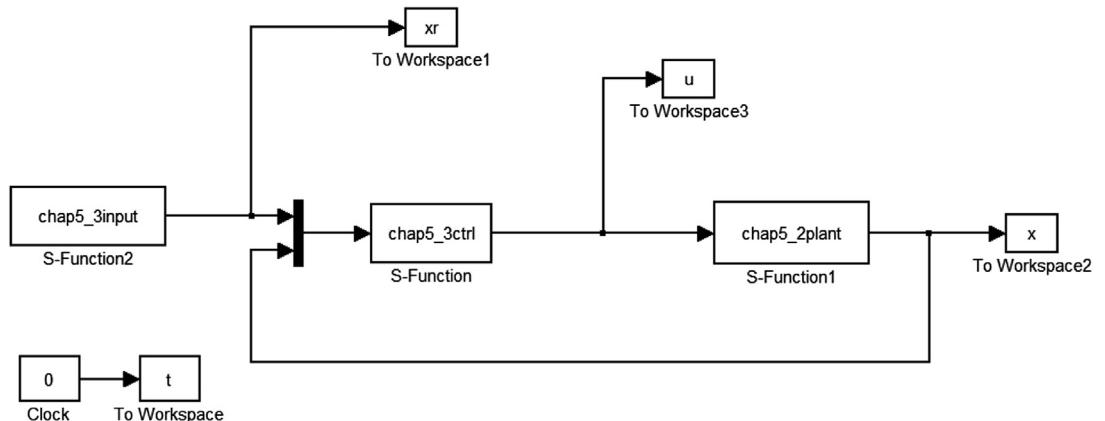
subplot(311);
plot(t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('x1');
subplot(312);
plot(t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('x2');
subplot(313);
plot(t,x(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x3');

figure(2);
subplot(311);
plot(t,u(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('u1');
subplot(312);
plot(t,u(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('u2');
subplot(313);
plot(t,u(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('u3');

```

**Second simulation: Tracking programs**

1. LMI design program: chap5\_2lmi.m
2. Simulink main program: chap5\_3sim.mdl



3. Ideal signal program: chap5\_3input.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
```

```

switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
S=2;
if S==1
 xr=[10 20 30]';
elseif S==2
 xr=[sin(t) cos(t) sin(t)];
end
sys(1:3)=xr(1:3);

```

**4. Controller program: chap5\_3ctrl.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];

```

```
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 6;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
x1=u(4);
x2=u(5);
x3=u(6);

x=[x1 x2 x3]';
xr=[u(1) u(2) u(3)]';

S=2;
if S==1
 dxr=[0 0 0]';
elseif S==2
 dxr=[cos(t) -sin(t) cos(t)]';
end

z=x-xr;
A=[-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
B=[1 0 0;
 0 1 0;
 0 0 1];

alfa=10;
if alfa==0
F=[2.0480 -5.0500 0.0000;
 -5.0500 0.5000 6.6000;
 -0.0000 6.6000 -0.5000];
elseif alfa==10
```

```

F=[-2.9520 -5.0500 0.0000;
 -5.0500 -4.5000 6.6000;
 -0.0000 6.6000 -5.5000];
end

```

```
ur = -F*xr - inv(B)*A*xr + inv(B)*dxr;
```

```
ut = F*x + ur;
```

```
sys(1:3)=ut;
```

**5. Plant program:** chap5\_2plant.m

**6. Plot program:** chap5\_3plot.m

```
close all;
```

```
figure(1);
```

```
subplot(311);
```

```
plot(t,xr(:,1),'r',t,x(:,1),'b','linewidth',2);
```

```
xlabel('time(s)');ylabel('xr1,x1');
```

```
subplot(312);
```

```
plot(t,xr(:,2),'r',t,x(:,2),'b','linewidth',2);
```

```
xlabel('time(s)');ylabel('xr2,x2');
```

```
subplot(313);
```

```
plot(t,xr(:,3),'r',t,x(:,3),'b','linewidth',2);
```

```
xlabel('time(s)');ylabel('xr3,x3');
```

```
figure(2);
```

```
subplot(311);
```

```
plot(t,xr(:,1)-x(:,1),'b','linewidth',2);
```

```
xlabel('time(s)');ylabel('xr1 tracking error');
```

```
subplot(312);
```

```
plot(t,xr(:,2)-x(:,2),'b','linewidth',2);
```

```
xlabel('time(s)');ylabel('xr2 tracking error');
```

```
subplot(313);
```

```
plot(t,xr(:,3)-x(:,3),'b','linewidth',2);
```

```
xlabel('time(s)');ylabel('xr3 tracking error');
```

```
figure(3);
```

```
subplot(311);
```

```
plot(t,u(:,1),'r','linewidth',2);
```

```
xlabel('time(s)');ylabel('u1');
```

```
subplot(312);
```

```
plot(t,u(:,2),'r','linewidth',2);
```

```
xlabel('time(s)');ylabel('u2');
```

```
subplot(313);
```

```
plot(t,u(:,3),'r','linewidth',2);
```

```
xlabel('time(s)');ylabel('u3');
```

## 5.3 SLIDING MODE CONTROLLER DESIGN FOR A LINEAR SYSTEM BASED ON LMI

### 5.3.1 System description

Consider a linear system as

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} + \mathbf{d}, \quad (5.11)$$

where  $\mathbf{x} \in R^n$ ,  $\mathbf{u} \in R^n$ ,  $\mathbf{A} \in R^{n \times n}$ ,  $\mathbf{B} \in R^{n \times n}$ ,  $\mathbf{d} \in R^{n \times 1}$  is disturbance.

The control goal is  $\mathbf{x} \rightarrow \mathbf{x}_r$  exponentially,  $\mathbf{x}_r$  is ideal signal.

### 5.3.2 Controller design

Define the tracking error as  $\mathbf{z} = \mathbf{x} - \mathbf{x}_r$ , then

$$\dot{\mathbf{z}} = \dot{\mathbf{x}} - \dot{\mathbf{x}}_r = \mathbf{Ax} + \mathbf{Bu} + \mathbf{d} - \dot{\mathbf{x}}_r$$

Design the controller as

$$\mathbf{u} = \mathbf{Fx} + \mathbf{u}_r + \mathbf{u}_s, \quad (5.12)$$

where  $\mathbf{F}$  is state feedback gain, which can be solved by LMI,  $\mathbf{u}_r = -\mathbf{Fx}_r - \mathbf{B}^{-1}\mathbf{Ax}_r + \mathbf{B}^{-1}\dot{\mathbf{x}}_r$ ,  $\mathbf{u}_s = -\mathbf{B}^{-1}(\eta \operatorname{sgn}(z))$ ,  $\eta \in R^{n \times 1}$ ,  $\eta_i > \bar{d}_i$ ,  $\eta \operatorname{sgn}(z) = [\eta_1 \operatorname{sgn} z_1 \dots \eta_n \operatorname{sgn} z_n]^T$ .

then

$$\begin{aligned} \mathbf{u} &= \mathbf{Fx} - \mathbf{Fx}_r - \mathbf{B}^{-1}\mathbf{Ax}_r + \mathbf{B}^{-1}\dot{\mathbf{x}}_r - \mathbf{B}^{-1}(\eta \operatorname{sgn}(z)) \\ &= \mathbf{Fz} - \mathbf{B}^{-1}\mathbf{Ax}_r + \mathbf{B}^{-1}\dot{\mathbf{x}}_r - \mathbf{B}^{-1}(\eta \operatorname{sgn}(z)) \end{aligned}$$

and

$$\begin{aligned} \dot{\mathbf{z}} &= \mathbf{Ax} + \mathbf{B}(\mathbf{Fz} - \mathbf{B}^{-1}\mathbf{Ax}_r + \mathbf{B}^{-1}\dot{\mathbf{x}}_r - \mathbf{B}^{-1}(\eta \operatorname{sgn}(z))) + \mathbf{d} - \dot{\mathbf{x}}_r \\ &= \mathbf{Ax} + \mathbf{BFz} - \mathbf{Ax}_r + \dot{\mathbf{x}}_r - \eta \operatorname{sgn}(z) + \mathbf{d} - \dot{\mathbf{x}}_r \\ &= \mathbf{Az} + \mathbf{BFz} - \eta \operatorname{sgn}(z) + \mathbf{d}, \end{aligned}$$

Referring to [2], we design the following theorem.

**Theorem 5.3:** If the following LMI is satisfied

$$\alpha \mathbf{P} + \mathbf{A}^T \mathbf{P} + \mathbf{M}^T + \mathbf{P} \mathbf{A} + \mathbf{M} < 0, \quad (5.13)$$

where  $\alpha > 0$ ,  $\mathbf{F} = (\mathbf{PB})^{-1}\mathbf{M}$ , then the closed system with plant (5.11) and controller (5.12) is exponentially stable.

**Proof:** Design the Lyapunov function as

$$V = \mathbf{z}^T \mathbf{P} \mathbf{z},$$

where  $\mathbf{P} = \operatorname{diag}\{p_i\}$  is a diagonal matrix and  $p_i > 0$ .

then

$$\begin{aligned}
\alpha V + \dot{V} &= \alpha V + (z^T P)' z + z^T P \dot{z} \\
&= \alpha V + \dot{z}^T P z + z^T P \dot{z} \\
&= \alpha V + (Az + BFz - \eta \operatorname{sgn}(z) + d)^T P z + z^T P (Az + BFz - \eta \operatorname{sgn}(z) + d) \\
&= \alpha V + z^T A^T P z + z^T F^T B^T P z + (-\eta \operatorname{sgn}(z) + d)^T P z \\
&\quad + z^T PAz + z^T PBFz + z^T P(-\eta \operatorname{sgn}(z) + d) \\
&\leq \alpha V + z^T (A^T P + F^T B^T P + PA + PBF) z \\
&= \alpha z^T P z + z^T \Omega z = z^T (\alpha P + \Omega) z
\end{aligned},$$

where  $\Omega = A^T P + F^T B^T P + PA + PBF$ ,

$$(-\eta \operatorname{sgn}(z) + d)^T P z = \sum_{i=1}^n (-\eta_i + d_i) p_i |z_i| < 0,$$

$$z^T P (-\eta \operatorname{sgn}(z) + d) = \sum_{i=1}^n (-\eta_i + d_i) p_i |z_i| < 0.$$

To guarantee  $\alpha V + \dot{V} \leq 0$ ,  $\alpha P + \Omega < 0$  is needed, that is

$$\alpha P + A^T P + F^T B^T P + PA + PBF < 0. \quad (5.14)$$

Since  $F$  and  $P$  are unknown, to solve Eq. (5.14), the left part must be linearized, define  $M = PBF$ , then we have

$$\alpha P + A^T P + M^T + PA + M < 0.$$

From  $\alpha V + \dot{V} \leq 0$ , using Lemma 1.3, we have  $V(t) \leq V(0)\exp(-\alpha t)$ , i.e., if  $t \rightarrow \infty$ ,  $V(t) \rightarrow 0$  and  $z \rightarrow 0$ , the tracking error converges to zero exponentially.

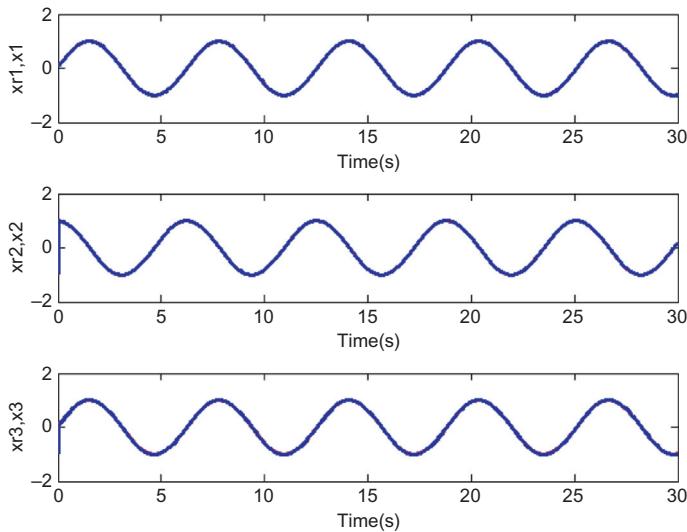
From the LMI YALMIP toolbox, we can get  $M$  and  $P$ , then  $F = (PB)^{-1}M$ .

### 5.3.3 Simulation example

Consider plant (5.11), and choose

$$A = \begin{bmatrix} -2.548 & 9.1 & 0 \\ 1 & -1 & 0 \\ 0 & -14.2 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ideal states are set as  $[\sin t \cos t \sin t]$ , disturbances are set as  $[50 \sin t \quad 50 \sin t \quad 50 \sin t]^T$ , from LMI (5.10), let



■ FIGURE 5.5 System states tracking with  $\alpha = 0$ .

$P = \begin{bmatrix} 1,000,000 & 0 & 0 \\ 0 & 1,000,000 & 0 \\ 0 & 0 & 1,000,000 \end{bmatrix}$ , and let  $\alpha = 0$  and  $\alpha = 10$ ,

respectively, we can get

$$F = \begin{bmatrix} -372.1481 & -5.05 & 0 \\ -5.05 & -373.6961 & 6.6 \\ 0 & 6.6 & -374.6961 \end{bmatrix} \text{ and}$$

$$F = \begin{bmatrix} -375.001 & -5.05 & 0 \\ -5.05 & -376.549 & 6.6 \\ 0 & 6.6 & -377.549 \end{bmatrix}, \text{ respectively.}$$

Use controller (5.12) and choose  $\eta = [50 \ 50 \ 50]^T$ . The simulation results are given in Figs. 5.5 and 5.6. In the simulation, we use the saturation function instead of the switch function, and set layer  $\Delta = 0.05$ .

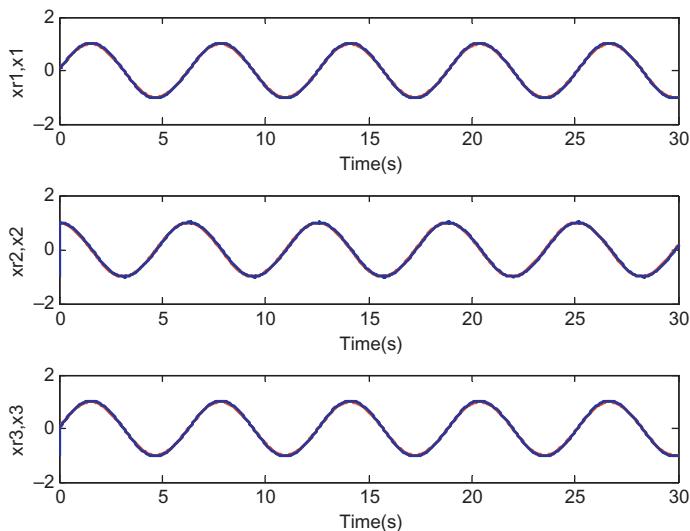
Simulation programs:

1. LMI design program: chap5\_4lmi.m

```
clear all;
close all;
```

```
%First example on the paper by M.Rehan
```

```
A = [-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
```



■ FIGURE 5.6 System states tracking with  $\alpha = 10$ .

```
B = [1 0 0;
 0 1 0;
 0 0 1];
%P = sdpvar(3,3);
F = sdpvar(3,3);
M = sdpvar(3,3);

P = 1000000*eye(3);

%FAI = (A' + F'*B')*P + P*(A + B*F);

alfa = 0;
alfa = 10;
FAI = alfa*P + A'*P + M' + P*A + M; %M = PBF

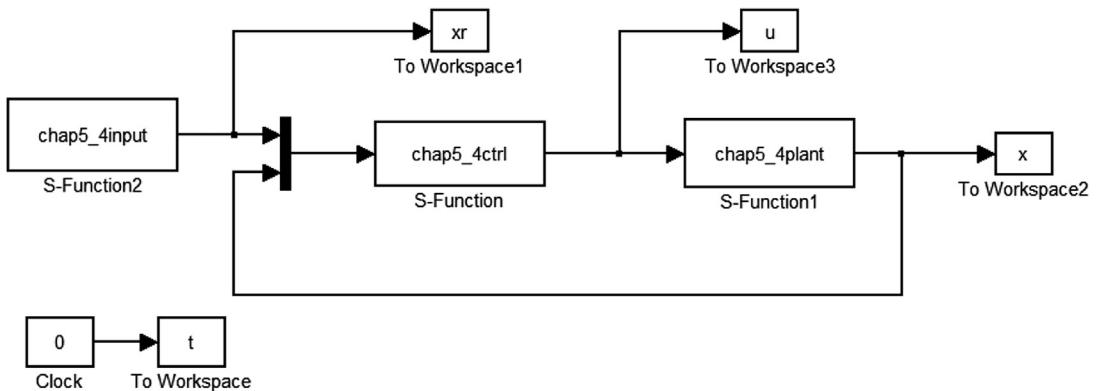
%LMI description
%L1 = set(P > 0);
L2 = set(FAI < 0);
%LL = L1 + L2;

solvesdp(L2);

%P = double(P);
M = double(M)

F = inv(P*B)*M
```

**2. Simulink main program: chap5\_4sim.mdl**



**3. Ideal signal program: chap5\_4input.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
S=2;

```

```

if S==1
 xr=[10 20 30]';
elseif S==2
 xr=[sin(t) cos(t) sin(t)];
end

```

```
sys(1:3)=xr(1:3);
```

**4. Controller program: chap5\_4ctrl.m**

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
```

```
switch flag,
```

```
case 0,
```

```
[sys,x0,str,ts]=mdlInitializeSizes;
```

```
case 3,
```

```
sys=mdlOutputs(t,x,u);
```

```
case {2,4,9}
```

```
sys=[];
```

```
otherwise
```

```
error(['Unhandled flag = ',num2str(flag)]);
```

```
end
```

```
function [sys,x0,str,ts]=mdlInitializeSizes
```

```
sizes=simsizes;
```

```
sizes.NumContStates=0;
```

```
sizes.NumDiscStates=0;
```

```
sizes.NumOutputs=3;
```

```
sizes.NumInputs=6;
```

```
sizes.DirFeedthrough=1;
```

```
sizes.NumSampleTimes=1;
```

```
sys=simsizes(sizes);
```

```
x0=[];
```

```
str=[];
```

```
ts=[0 0];
```

```
function sys=mdlOutputs(t,x,u)
```

```
x1=u(4);
```

```
x2=u(5);
```

```
x3=u(6);
```

```
x=[x1 x2 x3]';
```

```
xr=[u(1) u(2) u(3)]';
```

```
S=2;
```

```
if S==1
```

```
dxr=[0 0 0]';
```

```
elseif S==2
```

```
dxr=[cos(t) -sin(t) cos(t)]';
```

```
end
```

```

z = x - xr;

A = [-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
B = [1 0 0;
 0 1 0;
 0 0 1];

alfa = 10;
if alfa == 0
F = [-372.1481 -5.0500 0;
 -5.0500 -373.6961 6.6000;
 0 6.6000 -374.6961];
elseif alfa == 10
F = [-375.0010 -5.0500 0;
 -5.0500 -376.5490 6.6000;
 0 6.6000 -377.5490];
end

delta = 0.05;
kk = 1/delta;
for i = 1:1:3
 if z(i) > delta
 sats(i) = 1;
 elseif abs(z(i)) <= delta
 sats(i) = kk*z(i);
 elseif z(i) < -delta
 sats(i) = -1;
 end
end
xite = [50; 50; 50];

ur = -F*xr - inv(B)*A*xr + inv(B)*dxr;

%us = -inv(B)*[xite(1)*sign(z(1)) xite(2)*sign(z(2))
% xite(3)*sats]';
us = -inv(B)*[xite(1)*sats(1) xite(2)*sats(2) xite(3)
 *sats(3)]';
%us = 0;
ut = F*x + ur + us;
sys(1:3) = ut;

```

**5. Plant program: chap5\_4plant.m**

```

function [sys, x0, str, ts] = spacemodel(t, x, u, flag)
switch flag,

```

```

case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=3;
sizes.NumDiscStates=0;
sizes.NumOutputs=3;
sizes.NumInputs=3;
sizes.DirFeedthrough=0;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[0,-1,-1];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
A=[-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
B=[1 0 0;
 0 1 0;
 0 0 1];
ut=[u(1) u(2) u(3)]';
dt=[50*sin(t) 50*sin(t) 50*sin(t)]';
dx=A*x+B*ut+dt;

sys(1)=dx(1);
sys(2)=dx(2);
sys(3)=dx(3);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);

```

**6. Plot program: chap5\_4plot.m**

```

close all;

figure(1);
subplot(311);
plot(t,xr(:,1),'r',t,x(:,1),'b','linewidth',2);
xlabel('time(s)');ylabel('xr1,x1');
subplot(312);
plot(t,xr(:,2),'r',t,x(:,2),'b','linewidth',2);
xlabel('time(s)');ylabel('xr2,x2');
subplot(313);
plot(t,xr(:,3),'r',t,x(:,3),'b','linewidth',2);
xlabel('time(s)');ylabel('xr3,x3');

figure(2);
subplot(311);
plot(t,u(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('u1');
subplot(312);
plot(t,u(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('u2');
subplot(313);
plot(t,u(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('u3');

```

## 5.4 NONLINEAR SYSTEM STABILIZATION BASED ON LMI

### 5.4.1 System description

Consider a nonlinear system with Lipschitz conditions as

$$\dot{x} = f(x) + Ax + Bu, \quad (5.15)$$

where  $x \in R^n$ ,  $u \in R^n$ ,  $A \in R^{n \times n}$  and  $B \in R^{n \times n}$ , nonlinear function  $f(x)$  meet Lipschitz condition, that is

$$||f(x) - f(\bar{x})|| \leq ||L(x - \bar{x})||,$$

where  $L$  is the Lipschitz constant matrix.

The control goal is  $x \rightarrow 0$  exponentially.

### 5.4.2 Controller design

Design the controller as

$$u = Fx - B^{-1}f(0), \quad (5.16)$$

where  $F$  is state feedback gain, which can be solved by LMI.

Referring to [2], we design the following theorem.

**Theorem 5.4:** If the following LMI is satisfied

$$\begin{bmatrix} \alpha\mathbf{P} + \mathbf{A}^T\mathbf{P} + \mathbf{M}^T + \mathbf{P}\mathbf{A} + \mathbf{M} + \mathbf{L}^T\mathbf{L} & \mathbf{P} \\ \mathbf{P} & -\mathbf{I} \end{bmatrix} < 0, \quad (5.17)$$

where  $\mathbf{F} = (\mathbf{P}\mathbf{B})^{-1}\mathbf{M}$ ,  $\alpha > 0$ .

Then the closed system with plant (5.15) and controller (5.16) is exponentially stable.

**Proof:** design the Lyapunov function as

$$V = \mathbf{x}^T \mathbf{P} \mathbf{x},$$

where  $\mathbf{P} = \mathbf{P}^T > 0$ .

Since  $\dot{\mathbf{x}} = f(\mathbf{x}) - f(0) + (\mathbf{A} + \mathbf{B}\mathbf{F})\mathbf{x}$ , then

$$\begin{aligned} \dot{V} &= (\mathbf{x}^T \mathbf{P})' \mathbf{x} + \mathbf{x}^T \mathbf{P} \dot{\mathbf{x}} = \dot{\mathbf{x}}^T \mathbf{P} \mathbf{x} + \mathbf{x}^T \mathbf{P} \dot{\mathbf{x}} \\ &= (f(\mathbf{x}) - f(0) + (\mathbf{A} + \mathbf{B}\mathbf{F})\mathbf{x})^T \mathbf{P} \mathbf{x} + \mathbf{x}^T \mathbf{P}(f(\mathbf{x}) - f(0) + (\mathbf{A} + \mathbf{B}\mathbf{F})\mathbf{x}) \\ &= (f(\mathbf{x}) - f(0))^T \mathbf{P} \mathbf{x} + \mathbf{x}^T (\mathbf{A} + \mathbf{B}\mathbf{F})^T \mathbf{P} \mathbf{x} + \mathbf{x}^T \mathbf{P}(f(\mathbf{x}) - f(0)) + \mathbf{x}^T \mathbf{P}(\mathbf{A} + \mathbf{B}\mathbf{F})\mathbf{x}. \end{aligned}$$

Since  $f(\mathbf{x})$  is Lipschitz, then

$$[f(\mathbf{x}) - f(0)]^T [f(\mathbf{x}) - f(0)] \leq (\mathbf{L}(\mathbf{x} - 0))^T \mathbf{L}(\mathbf{x} - 0) = \mathbf{x}^T \mathbf{L}^T \mathbf{L} \mathbf{x},$$

i.e.,

$$\mathbf{x}^T \mathbf{L}^T \mathbf{L} \mathbf{x} - (f(\mathbf{x}) - f(0))^T (f(\mathbf{x}) - f(0)) \geq 0.$$

Let  $\alpha > 0$ , then

$$\begin{aligned} \alpha V + \dot{V} &\leq \alpha \mathbf{x}^T \mathbf{P} \mathbf{x} + (f(\mathbf{x}) - f(0))^T \mathbf{P} \mathbf{x} + \mathbf{x}^T (\mathbf{A} + \mathbf{B}\mathbf{F})^T \mathbf{P} \mathbf{x} + \mathbf{x}^T \mathbf{P}(f(\mathbf{x}) - f(0)) \\ &\quad + \mathbf{x}^T \mathbf{P}(\mathbf{A} + \mathbf{B}\mathbf{F})\mathbf{x} + \mathbf{x}^T \mathbf{L}^T \mathbf{L} \mathbf{x} - (f(\mathbf{x}) - f(0))^T (f(\mathbf{x}) - f(0)). \end{aligned} \quad (5.18)$$

Let  $\mathbf{Y} = \begin{bmatrix} \mathbf{x} \\ f(\mathbf{x}) - f(0) \end{bmatrix}^T$ , then  $\mathbf{Y}^T = [\mathbf{x}^T \quad (f(\mathbf{x}) - f(0))^T]$ . Since

$$\begin{aligned} \alpha \mathbf{x}^T \mathbf{P} \mathbf{x} + \mathbf{x}^T (\mathbf{A} + \mathbf{B}\mathbf{F})^T \mathbf{P} \mathbf{x} + \mathbf{x}^T \mathbf{P}(\mathbf{A} + \mathbf{B}\mathbf{F})\mathbf{x} + \mathbf{x}^T \mathbf{L}^T \mathbf{L} \mathbf{x} \\ = \mathbf{x}^T (\alpha \mathbf{P} + (\mathbf{A} + \mathbf{B}\mathbf{F})^T \mathbf{P} \mathbf{x} + \mathbf{P}(\mathbf{A} + \mathbf{B}\mathbf{F}) + \mathbf{L}^T \mathbf{L}) \mathbf{x}, \end{aligned}$$

then Eq. (5.18) becomes

$$\dot{V} \leq \mathbf{Y}^T \Omega \mathbf{Y},$$

where  $\Omega = \begin{bmatrix} \alpha \mathbf{P} + (\mathbf{A}^T + \mathbf{F}^T \mathbf{B}^T) \mathbf{P} + \mathbf{P}(\mathbf{A} + \mathbf{B}\mathbf{F}) + \mathbf{L}^T \mathbf{L} & \mathbf{P} \\ \mathbf{P} & -\mathbf{I} \end{bmatrix}$ .

To guarantee  $\alpha V + \dot{V} \leq 0$ ,  $\mathbf{Y}^T \boldsymbol{\Omega} \mathbf{Y} < 0$  is needed, that is,  $\boldsymbol{\Omega} < 0$ , i.e.,

$$\begin{bmatrix} \alpha \mathbf{P} + (\mathbf{A}^T + \mathbf{F}^T \mathbf{B}^T) \mathbf{P} + \mathbf{P}(\mathbf{A} + \mathbf{F}\mathbf{B}) + \mathbf{L}^T \mathbf{L} & \mathbf{P} \\ \mathbf{P} & -\mathbf{I} \end{bmatrix} < 0. \quad (5.19)$$

Since  $\mathbf{F}$  and  $\mathbf{P}$  are unknown, to solve Eq. (5.19), the left part must be linearized. Define  $\mathbf{M} = \mathbf{P}\mathbf{B}\mathbf{F}$ , then we have

$$\begin{bmatrix} \alpha \mathbf{P} + \mathbf{A}^T \mathbf{P} + \mathbf{M}^T + \mathbf{P}\mathbf{A} + \mathbf{M} + \mathbf{L}^T \mathbf{L} & \mathbf{P} \\ \mathbf{P} & -\mathbf{I} \end{bmatrix} < 0.$$

From  $\alpha V + \dot{V} \leq 0$ , using Lemma 1.3, we have  $V(t) \leq V(0) \exp(-\alpha t)$ , i.e., if  $t \rightarrow \infty$ ,  $V(t) \rightarrow 0$  and  $\mathbf{x} \rightarrow 0$ , the system states convergence to zero exponentially.

From the LMI YALMIP toolbox, we can get  $\mathbf{M}$  and  $\mathbf{P}$ , then  $\mathbf{F} = (\mathbf{P}\mathbf{B})^{-1}\mathbf{M}$ .

### 5.4.3 Simulation example

Consider system (5.15), and choose  $\mathbf{A} = \begin{bmatrix} -2.548 & 9.1 & 0 \\ 1 & -1 & 1 \\ 0 & -14.2 & 0 \end{bmatrix}$ ,  $\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  and  $f(\mathbf{x}) = \frac{1}{2} \begin{bmatrix} |x_1 + a_1| - |x_1 - a_2| \\ 0 \\ 0 \end{bmatrix}$ .

From the  $f(\mathbf{x})$  expression, we can get  $\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ . From LMI

(5.17), let  $\alpha = 0$  and  $\alpha = 10$ , respectively, we can get  $\mathbf{F} = \begin{bmatrix} -1.5075 & -5.05 & 0 \\ -5.05 & -0.5 & 6.6 \\ 0 & 6.6 & -1.5 \end{bmatrix}$  and  $\mathbf{F} = \begin{bmatrix} -6.5075 & -5.05 & 0 \\ -5.05 & -5.5 & 6.6 \\ 0 & 6.6 & -6.5 \end{bmatrix}$

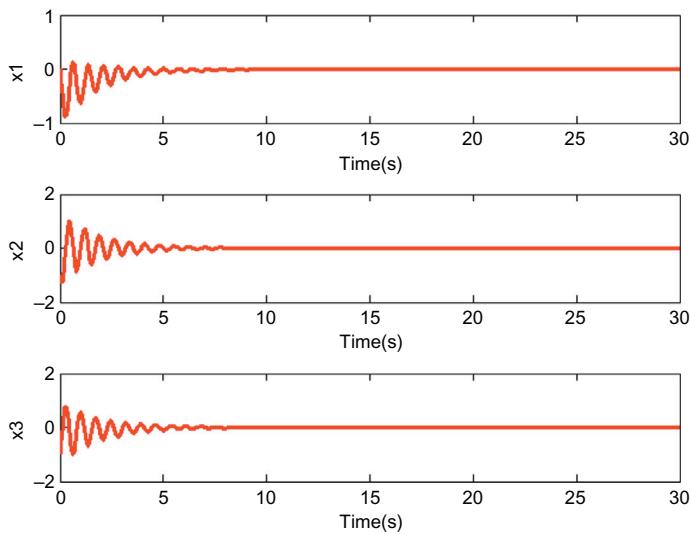
respectively, use controller (5.16), the simulation results are given in Figs. 5.7 and 5.8.

## APPENDIX: LIPSCHITZ CONSTANT MATRIX DESIGN

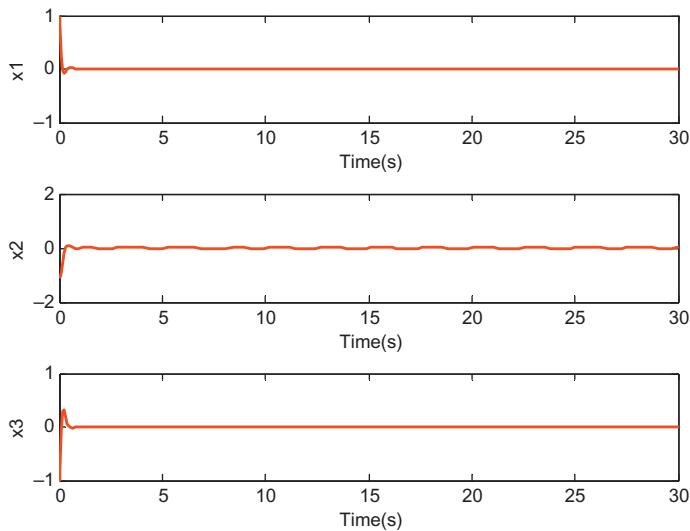
When  $\xi \in [a, b]$ ,  $f(\xi)$  is a smooth function, according to mean value theorem, we have  $f(a) - f(b) = f'(\xi)(a - b)$ , then

$$|f(a) - f(b)| = |f'(\xi)(a - b)| \leq \max(|f'(\xi)|) |a - b|,$$

where  $L = \max(|f'(\xi)|)$ .



■ FIGURE 5.7 System states response with  $\alpha = 0$ .



■ FIGURE 5.8 System states response with  $\alpha = 10$ .

Since  $\frac{\partial f(1)}{\partial x_1} = \frac{1}{2} \frac{\partial(|x_1 + a_1| - |x_1 - a_2|)}{\partial x_1}$  and

$$|x_1 + a_1| - |x_1 - a_2|$$

$$= \begin{cases} x_1 + a_1 - (x_1 - a_2) = a_2 - a_1 & x_1 + a_1 > 0 \& x_1 - a_2 > 0 \\ x_1 + a_1 + (x_1 - a_2) = 2x_1 - a_2 + a_1 & x_1 + a_1 > 0 \& x_1 - a_2 < 0 \\ -(x_1 + a_1) - (x_1 - a_2) = -2x_1 - a_1 + a_2 & x_1 + a_1 < 0 \& x_1 - a_2 > 0 \\ -(x_1 + a_1) + (x_1 - a_2) = -a_1 - a_2 & x_1 + a_1 < 0 \& x_1 - a_2 < 0 \end{cases}$$

according to  $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f(1)}{\partial x_1} & \frac{\partial f(1)}{\partial x_2} & \frac{\partial f(1)}{\partial x_3} \\ \frac{\partial f(2)}{\partial x_1} & \frac{\partial f(2)}{\partial x_2} & \frac{\partial f(2)}{\partial x_3} \\ \frac{\partial f(3)}{\partial x_1} & \frac{\partial f(3)}{\partial x_2} & \frac{\partial f(3)}{\partial x_3} \end{bmatrix}$ , then  $\frac{\partial f(1)}{\partial x_1}$  is equal to 0 or 1 or -1 or 0, then

$$\max\left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

According to  $\|f(\mathbf{x}) - f(\bar{\mathbf{x}})\| \leq \|L(\mathbf{x} - \bar{\mathbf{x}})\|$ , let  $L \geq \max(f'(\mathbf{x}))$ , then we

$$\text{have } L = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Simulation programs:

### 1. LMI design program: chap5\_51mi.m

```
clear all;
close all;

%First example on the paper by M. Rehan
A = [-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
B = [1 0 0;
 0 1 0;
 0 0 1];
L = [2 0 0;
 0 0 0;
 0 0 0];
P = sdpvar(3,3);
F = sdpvar(3,3);
M = sdpvar(3,3);
```

```

alfa=0;
alfa=10;
FAI=[alfa*P+A'*P+M'+P*A+M+L'*L;P;P- eye(3)]; %
M=PBF

%LMI description
L1=set(P>0);
L2=set(FAI<0);
LL=L1+L2;

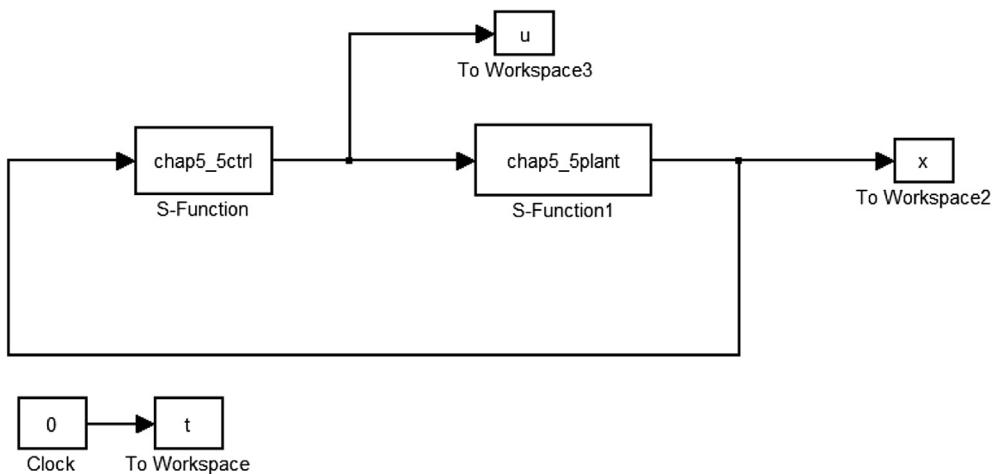
solvesdp(LL);

P=double(P);
M=double(M)

F=inv(P*B)*M

```

**2. Simulink main program: chap5\_5sim.mdl**



**3. Controller program: chap5\_5ctrl.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end

```

```

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
x1=u(1);
x2=u(2);
x3=u(3);

f0=[- 0.05 0 0]';

B=[1 0 0;
 0 1 0;
 0 0 1];

alfa=10;
if alfa==0
F=[-1.5075 -5.0500 0.0000;
 -5.0500 -0.5000 6.6000;
 0.0000 -6.6000 -1.5000];
elseif alfa==10
F=[-6.5075 -5.0500 0.0000
 -5.0500 -5.5000 6.6000
 -0.0000 6.6000 -6.5000];
end

ut=F*[x1;x2;x3]-inv(B)*f0;

sys(1:3)=ut;

```

**4. Plant program: chap5\_5plant.m**

```

function [sys,x0,str,ts]=spacemodel(t,x,u,flag)
switch flag,
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;

```

```

case 1,
 sys = mdlDerivatives(t,x,u);
case 3,
 sys = mdlOutputs(t,x,u);
case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [1, -1, -1];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
a1=1;a2=1.1;
fx=0.5*[abs(x(1)+a1)-abs(x(1)-a2);0;0];
A=[-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
B=[1 0 0;
 0 1 0;
 0 0 1];
ut=[u(1) u(2) u(3)]';
dx=fx+A*x+B*ut;

sys(1)=dx(1);
sys(2)=dx(2);
sys(3)=dx(3);
function sys = mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);

```

5. Plot program: chap5\_5plot.m

```

close all;

figure(1);
subplot(311);
plot(t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('x1');
subplot(312);
plot(t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('x2');
subplot(313);
plot(t,x(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x3');

figure(2);
subplot(311);
plot(t,u(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('u1');
subplot(312);
plot(t,u(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('u2');
subplot(313);
plot(t,u(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('u3');

```

## 5.5 NONLINEAR SYSTEM TRACKING CONTROL BASED ON LMI

### 5.5.1 System description

Consider a nonlinear system with Lipschitz conditions as

$$\dot{x} = f(x) + Ax + Bu + d, \quad (5.20)$$

where  $x \in R^n$ ,  $u \in R^n$ ,  $A \in R^{n \times n}$ ,  $B \in R^{n \times n}$  and  $d \in R^{n \times 1}$  are disturbance, nonlinear function  $f(x)$  meets the Lipschitz condition, that is

$$||f(x) - f(\bar{x})|| \leq ||L(x - \bar{x})||, \quad (5.21)$$

where  $L$  is the Lipschitz constant matrix.

The control goal is  $x \rightarrow x_r$  exponentially, and  $x_r$  is ideal command signal.

### 5.5.2 Controller design

Define the tracking error as  $z = x - x_r$ , then

$$\dot{z} = \dot{x} - \dot{x}_r = Ax + Bu + f(x) + d - \dot{x}_r.$$

Design the controller as

$$u = Fx + u_r + u_s, \quad (5.22)$$

where  $F$  is state feedback gain, which can be solved by LMI,  $u_r = -Fx_r - B^{-1}Ax_r - B^{-1}f(x_r) + B^{-1}\dot{x}_r$ ,  $u_s = -B^{-1}(\eta \operatorname{sgn}(z))$ ,  $\eta \in R^{n \times 1}$ ,  $\eta_i > \bar{d}_i$ ,  $\eta \operatorname{sgn}(z) = [\eta_1 \operatorname{sgn} z_1 \dots \eta_n \operatorname{sgn} z_n]^T$ .

then

$$\begin{aligned} u &= Fx - Fx_r - B^{-1}Ax_r - B^{-1}f(x_r) + B^{-1}\dot{x}_r - B^{-1}(\eta \operatorname{sgn}(z)) \\ &= Fz - B^{-1}Ax_r - B^{-1}f(x_r) + B^{-1}\dot{x}_r - B^{-1}(\eta \operatorname{sgn}(z)) \end{aligned}$$

and

$$\begin{aligned} \dot{z} &= Ax + B(Fz - B^{-1}Ax_r - B^{-1}f(x_r) + B^{-1}\dot{x}_r - B^{-1}(\eta \operatorname{sgn}(z))) + f(x) + d - \dot{x}_r \\ &= Ax + BFz - Ax_r - f(x_r) + \dot{x}_r + f(x) - \eta \operatorname{sgn}(z) + d - \dot{x}_r \\ &= Az + BFz + f(x) - f(x_r) - \eta \operatorname{sgn}(z) + d \end{aligned}$$

Referring to [2], we design the following theorem.

**Theorem 5.5:** If the following LMI is satisfied

$$\begin{bmatrix} \alpha P + A^T P + M^T + PA + M + L^T L & P \\ P & -I \end{bmatrix} < 0, \quad (5.23)$$

where  $F = (PB)^{-1}M$ .

Then the closed system with plant (5.20) and controller (5.22) is exponentially stable.

**Proof:** design the Lyapunov function as

$$V = z^T P z,$$

where  $P = P^T > 0$ .

Since

$$\dot{z} = Az + BFz + f(x) - f(x_r) - \eta \operatorname{sgn}(z) + d,$$

then

$$\begin{aligned}
\dot{V} &= (\mathbf{z}^T \mathbf{P})' \mathbf{z} + \mathbf{z}^T \mathbf{P} \dot{\mathbf{z}} = \dot{\mathbf{z}}^T \mathbf{P} \mathbf{z} + \mathbf{z}^T \mathbf{P} \dot{\mathbf{z}} \\
&= (\mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{F}\mathbf{z} + \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r) - \boldsymbol{\eta} \operatorname{sgn}(\mathbf{z}) + \mathbf{d})^T \mathbf{P} \mathbf{z} \\
&\quad + \mathbf{z}^T \mathbf{P} (\mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{F}\mathbf{z} + \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r) - \boldsymbol{\eta} \operatorname{sgn}(\mathbf{z}) + \mathbf{d}) \\
&= \mathbf{z}^T (\mathbf{A} + \mathbf{B}\mathbf{F})^T \mathbf{P} \mathbf{z} + (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r))^T \mathbf{P} \mathbf{z} + (-\boldsymbol{\eta} \operatorname{sgn}(\mathbf{z}) + \mathbf{d})^T \mathbf{P} \mathbf{z} \\
&\quad + \mathbf{z}^T \mathbf{P} (\mathbf{A} + \mathbf{B}\mathbf{F}) \mathbf{z} + \mathbf{z}^T \mathbf{P} (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r)) + \mathbf{z}^T \mathbf{P} (-\boldsymbol{\eta} \operatorname{sgn}(\mathbf{z}) + \mathbf{d})
\end{aligned}$$

Since

$$(-\boldsymbol{\eta} \operatorname{sgn}(\mathbf{z}) + \mathbf{d})^T \mathbf{P} \mathbf{z} = \sum_{i=1}^n (-\eta_i + d_i) p_i |z_i| < 0,$$

$$\mathbf{z}^T \mathbf{P} (-\boldsymbol{\eta} \operatorname{sgn}(\mathbf{z}) + \mathbf{d}) = \sum_{i=1}^n (-\eta_i + d_i) p_i |z_i| < 0$$

From Eq. (5.21), we have

$$\begin{aligned}
[\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r)]^T [\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r)] &\leq (\mathbf{L}(\mathbf{x} - \mathbf{x}_r))^T \mathbf{L}(\mathbf{x} - \mathbf{x}_r) \\
&= (\mathbf{x} - \mathbf{x}_r)^T \mathbf{L}^T \mathbf{L}(\mathbf{x} - \mathbf{x}_r) = \mathbf{z}^T \mathbf{L}^T \mathbf{L} \mathbf{z},
\end{aligned}$$

$$\text{i.e., } \mathbf{z}^T \mathbf{L}^T \mathbf{L} \mathbf{z} - [\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r)]^T [\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r)] \geq 0$$

Let  $\alpha > 0$ , then

$$\begin{aligned}
\alpha V + \dot{V} &\leq \alpha \mathbf{z}^T \mathbf{P} \mathbf{z} + \mathbf{z}^T (\mathbf{A} + \mathbf{B}\mathbf{F})^T \mathbf{P} \mathbf{z} + (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r))^T \mathbf{P} \mathbf{z} + \mathbf{z}^T \mathbf{P} (\mathbf{A} + \mathbf{B}\mathbf{F}) \mathbf{z} \\
&\quad + \mathbf{z}^T \mathbf{P} (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r)) + \mathbf{z}^T \mathbf{L}^T \mathbf{L} \mathbf{z} - [\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r)]^T [\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r)]
\end{aligned} \tag{5.24}$$

Define  $\mathbf{Y} = \begin{bmatrix} \mathbf{z} \\ \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r) \end{bmatrix}^T$ , then  $\mathbf{Y}^T = [\mathbf{z}^T \quad (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_r))^T]$ .

Since

$$\begin{aligned}
\alpha \mathbf{z}^T \mathbf{P} \mathbf{z} + \mathbf{z}^T (\mathbf{A} + \mathbf{B}\mathbf{F})^T \mathbf{P} \mathbf{z} + \mathbf{z}^T \mathbf{P} (\mathbf{A} + \mathbf{B}\mathbf{F}) \mathbf{z} + \mathbf{z}^T \mathbf{L}^T \mathbf{L} \mathbf{z} \\
= \mathbf{z}^T (\alpha \mathbf{P} + (\mathbf{A} + \mathbf{B}\mathbf{F})^T \mathbf{P} + \mathbf{P} (\mathbf{A} + \mathbf{B}\mathbf{F}) + \mathbf{L}^T \mathbf{L}) \mathbf{z},
\end{aligned}$$

then Eq. (5.24) becomes

$$\dot{V} \leq \mathbf{Y}^T \boldsymbol{\Omega} \mathbf{Y},$$

$$\text{where } \boldsymbol{\Omega} = \begin{bmatrix} \alpha \mathbf{P} + (\mathbf{A}^T + \mathbf{F}^T \mathbf{B}^T) \mathbf{P} + \mathbf{P} (\mathbf{A} + \mathbf{F}\mathbf{B}) + \mathbf{L}^T \mathbf{L} & \mathbf{P} \\ \mathbf{P} & -\mathbf{I} \end{bmatrix}.$$

To guarantee  $\alpha V + \dot{V} \leq 0$ , we need  $\mathbf{Y}^T \boldsymbol{\Omega} \mathbf{Y} < 0$ , i.e.,  $\boldsymbol{\Omega} < 0$ , then

$$\begin{bmatrix} \alpha \mathbf{P} + (\mathbf{A}^T + \mathbf{F}^T \mathbf{B}^T) \mathbf{P} + \mathbf{P} (\mathbf{A} + \mathbf{F}\mathbf{B}) + \mathbf{L}^T \mathbf{L} & \mathbf{P} \\ \mathbf{P} & -\mathbf{I} \end{bmatrix} < 0. \tag{5.25}$$

Since  $\mathbf{F}$  and  $\mathbf{P}$  are unknown, to solve Eq. (5.25), the left part must be linearized, define  $\mathbf{M} = \mathbf{PBF}$ , then we have

$$\begin{bmatrix} \alpha\mathbf{P} + \mathbf{A}^T\mathbf{P} + \mathbf{M}^T + \mathbf{PA} + \mathbf{M} + \mathbf{L}^T\mathbf{L} & \mathbf{P} \\ \mathbf{P} & -\mathbf{I} \end{bmatrix} < 0$$

From  $\alpha V + \dot{V} \leq 0$ , using Lemma 1.3, we have  $V(t) \leq V(0) \exp(-\alpha t)$ , i.e., if  $t \rightarrow \infty$ ,  $V(t) \rightarrow 0$  and  $z \rightarrow 0$ , the tracking error convergence to zero exponentially.

From the LMI YALMIP toolbox, we can get  $\mathbf{M}$  and  $\mathbf{P}$ , then  $\mathbf{F} = (\mathbf{PB})^{-1}\mathbf{M}$ .

### 5.5.3 Simulation example

Consider system (5.20), choose  $\mathbf{A} = \begin{bmatrix} -2.548 & 9.1 & 0 \\ 1 & -1 & 1 \\ 0 & -14.2 & 0 \end{bmatrix}$ ,  $\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  and  $f(\mathbf{x}) = \frac{1}{2} \begin{bmatrix} |x_1 + a_1| - |x_1 - a_2| \\ 0 \\ 0 \end{bmatrix}$ . From the expression of  $f(\mathbf{x})$ , we can obtain  $\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ .

Ideal states are set as  $[\sin t \cos t \sin t]$ , disturbances are set as  $[50 \sin t \quad 50 \sin t \quad 50 \sin t]^T$ .

From LMI (5.20), let  $\alpha = 0$  and  $\alpha = 10$ , respectively, we can get  $\mathbf{F} = \begin{bmatrix} -1.5075 & -5.05 & 0 \\ -5.05 & -0.5 & 6.6 \\ 0 & 6.6 & -1.5 \end{bmatrix}$  and  $\mathbf{F} = \begin{bmatrix} -6.5075 & -5.05 & 0 \\ -5.05 & -5.5 & 6.6 \\ 0 & 6.6 & -6.5 \end{bmatrix}$ , respectively, use the controller (5.22), choose  $\eta = [50 \quad 50 \quad 50]^T$ , the simulation results are given in Figs. 5.9 and 5.10. In the simulation, we use the saturation function instead of the switch function, and set layer  $\Delta = 0.05$ .

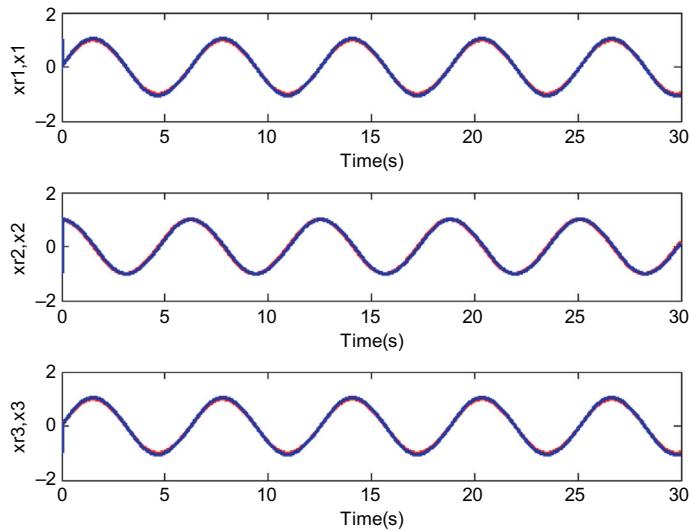
Simulation programs:

1. LMI design program: chap5\_6lmi.m

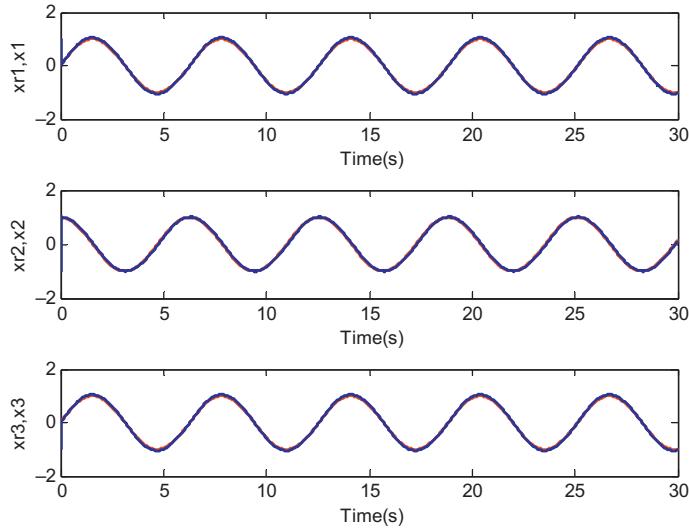
```
clear all;
close all;
```

%First example on the paper by M. Rehan

```
A = [-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
```



■ FIGURE 5.9 System states tracking with  $\alpha = 0$ .



■ FIGURE 5.10 System states tracking with  $\alpha = 10$ .

```
B = [1 0 0;
 0 1 0;
 0 0 1];
L = [2 0 0;
 0 0 0;
 0 0 0];
```

```

P = sdpvar(3,3);
F = sdpvar(3,3);
M = sdpvar(3,3);

alfa = 0;
alfa = 10;
FAI = [alfa*P + A'*P + M' + P*A + M + L'*L; P - eye(3)];
%M = PBF

%LMI description
L1 = set(P > 0);
L2 = set(FAI < 0);
LL = L1 + L2;

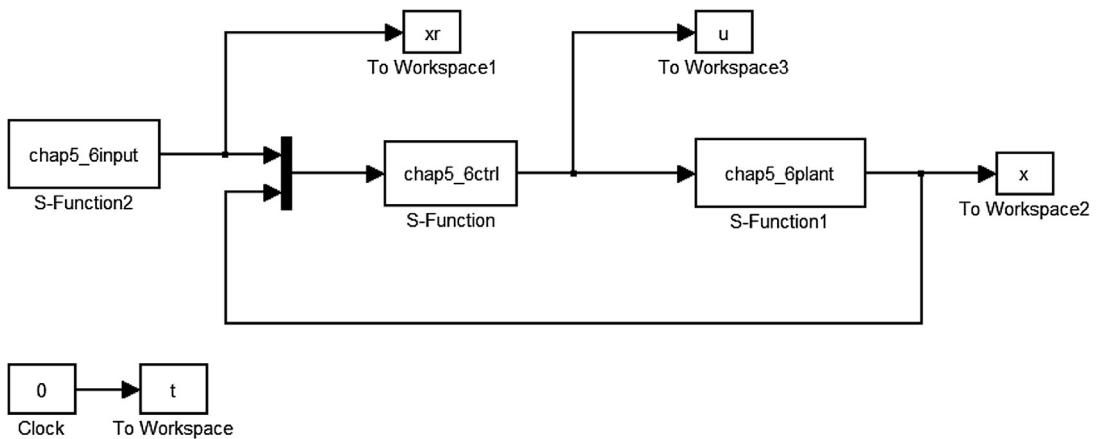
solvesdp(LL);

P = double(P);
M = double(M)

F = inv(P*B)*M

```

2. Simulink main program: chap5\_6sim.mdl



3. Ideal signal program: chap5\_6input.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
 sys = mdlOutputs(t,x,u);

```

```

case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
S=2;
if S==1
 xr=[10 20 30]';
elseif S==2
 xr=[sin(t) cos(t) sin(t)];
end
sys(1:3)=xr(1:3);

```

**4. Controller program: chap5\_6ctrl.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
```

```

sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 6;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys = mdlOutputs(t,x,u)

x1=u(4);
x2=u(5);
x3=u(6);

x=[x1 x2 x3]';
xr=[u(1) u(2) u(3)]';

S=2;
if S==1
 dxr=[0 0 0]';
elseif S==2
 dxr=[cos(t) -sin(t) cos(t)]';
end

z=x-xr;

A=[-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
B=[1 0 0;
 0 1 0;
 0 0 1];

alfa=10;
if alfa==0
F=[-1.5075 -5.0500 0.0000;
 -5.0500 -0.5000 6.6000;
 0.0000 6.6000 -1.5000];
elseif alfa==10
F=[-6.5075 -5.0500 0.0000;
 -5.0500 -5.5000 6.6000;
 0.0000 -6.6000 -6.5000];
end

```

```

a1=1;a2=1.1;
fxr=0.5*[abs(xr(1)+a1)-abs(xr(1)-a2);0;0];

delta=0.05;
kk=1/delta;
for i=1:1:3
 if z(i)>delta
 sats(i)=1;
 elseif abs(z(i))<=delta
 sats(i)=kk*z(i);
 elseif z(i)<-delta
 sats(i)=-1;
 end
end
xite=[50;50;50];

ur = - F*xr - inv(B)*A*xr + inv(B)*dxr;

%us = - inv(B)*[xite(1)*sign(z(1)) xite(2)*sign(z(2))
xite(3)*sats]';
us = - inv(B)*[xite(1)*sats(1) xite(2)*sats(2) xite(3)
*sats(3)]';
%us = 0;

ur = - F*xr - inv(B)*A*xr - inv(B)*fxr + inv(B)*dxr;
ut = F*x + ur + us;

sys(1:3)=ut;

```

**5. Plant program: chap5\_6plant.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;

```

```

sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [1, -1, -1];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
a1=1;a2=1.1;
fx=0.5*[abs(x(1)+a1)-abs(x(1)-a2);0;0];
A=[-2.548 9.1 0;
 1 -1 1;
 0 -14.2 0];
B=[1 0 0;
 0 1 0;
 0 0 1];
ut=[u(1) u(2) u(3)]';
dt=[50*sin(t) 50*sin(t) 50*sin(t)]';
dx = fx + A*x + B*ut + dt;

sys(1)=dx(1);
sys(2)=dx(2);
sys(3)=dx(3);
function sys = mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);

```

**6. Plot program: chap5\_6plot.m**

```

close all;

figure(1);
subplot(311);
plot(t,xr(:,1),'r',t,x(:,1),'b','linewidth',2);
xlabel('time(s)');ylabel('xr1,x1');
subplot(312);
plot(t,xr(:,2),'r',t,x(:,2),'b','linewidth',2);
xlabel('time(s)');ylabel('xr2,x2');
subplot(313);

```

```

plot(t,xr(:,3),'r',t,x(:,3),'b','linewidth',2);
xlabel('time(s)');ylabel('xr3,x3');

figure(2);
subplot(311);
plot(t,u(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('u1');
subplot(312);
plot(t,u(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('u2');
subplot(313);
plot(t,u(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('u3');

```

## 5.6 SLIDING MODE CONTROL FOR CHAOTIC SYSTEMS BASED ON LMI

### 5.6.1 System description

Consider a Lorenz system as

$$\begin{aligned}\dot{x}_1(t) &= a(x_2 - x_1) \\ \dot{x}_2(t) &= rx_1 - x_2 - x_1x_3 + u_1, \\ \dot{x}_3(t) &= -bx_3 + x_1x_2 + u_2\end{aligned}\quad (5.26)$$

where  $a = 10$ ,  $b = \frac{8}{3}$ ,  $r = 28$ ,  $u_1$  and  $u_2$  are control input.

Eq. (5.26) becomes

$$\dot{\mathbf{x}}(t) = \mathbf{Ax} + \mathbf{f}(\mathbf{x}) + \mathbf{Bu} = \mathbf{Ax} + \begin{pmatrix} 0 \\ f_2(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} 0 \\ \mathbf{I} \end{pmatrix} \mathbf{u}, \quad (5.27)$$

where  $\mathbf{x} \in \mathbf{R}^3$ ,  $\mathbf{u} = (u_1 \ u_2)^T \in \mathbf{R}^2$ ,  $\mathbf{A} = \begin{pmatrix} -a & a & 0 \\ r & -1 & 0 \\ 0 & 0 & -b \end{pmatrix}$ ,

$$f_2(\mathbf{x}) = \begin{pmatrix} -x_1x_3 \\ x_1x_2 \end{pmatrix} \text{ and } \mathbf{B} = \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

### 5.6.2 Traditional sliding mode control based on LMI

Define the sliding mode function as

$$s = \mathbf{Cx}, \quad (5.28)$$

where  $\mathbf{C} = [\mathbf{C}_1 \quad \mathbf{I}] = \begin{bmatrix} \mathbf{C}_1(1) & 1 & 0 \\ \mathbf{C}_1(2) & 0 & 1 \end{bmatrix}$ ,  $\mathbf{C}_1 \in \mathbb{R}^{2 \times 1}$ ,  $\mathbf{I}$  is a unit matrix with  $2 \times 2$ .

Choose the Lyapunov function as

$$V(t) = \frac{1}{2} \mathbf{s}^T \mathbf{s},$$

then

$$\dot{\mathbf{s}} = \mathbf{C}\dot{\mathbf{x}} = \mathbf{CAx} + \mathbf{Cf(x)} + \mathbf{Bu}$$

$$\dot{V}(t) = \mathbf{s}^T \dot{\mathbf{s}} = \mathbf{s}^T (\mathbf{CAx} + \mathbf{Cf(x)} + \mathbf{Bu}),$$

$$\text{where } \mathbf{CB} = [\mathbf{C}_1 \quad \mathbf{I}] \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix} = \mathbf{I}.$$

Design the controller as

$$\mathbf{u} = -\mathbf{CAx} - \mathbf{Cf(x)} - \eta \mathbf{s}, \quad (5.29)$$

$$\text{where } \boldsymbol{\eta} = \begin{bmatrix} \eta & 0 \\ 0 & \eta \end{bmatrix} \text{ and } \eta > 0.$$

then

$$\begin{aligned} \dot{V}(t) &= \mathbf{s}^T \dot{\mathbf{s}} = \mathbf{s}^T (\mathbf{CAx} + \mathbf{Cf(x)} + \mathbf{Bu}) \\ &= \mathbf{s}^T (\mathbf{CAx} + \mathbf{Cf(x)} + \mathbf{u}) = \mathbf{s}^T (\mathbf{CAx} + \mathbf{Cf(x)} - \mathbf{CAx} - \mathbf{Cf(x)} - \eta \mathbf{s}) \\ &\leq -\eta \|\mathbf{s}\| \leq 0 \end{aligned}$$

Since  $\mathbf{s}^T \dot{\mathbf{s}} \leq 0$ , there exists  $t > t_0$ , we have  $\mathbf{s} = 0$ , then we have  $\mathbf{u} = -\mathbf{CAx} - \mathbf{Cf(x)}$ .

Since

$$\begin{pmatrix} 0 \\ \mathbf{I} \end{pmatrix} [\mathbf{C}_1 \quad \mathbf{I}] f(\mathbf{x}) = \begin{pmatrix} 0 & \mathbf{0} \\ \mathbf{C}_1 & \mathbf{I} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ f_2(2) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \mathbf{C}_1(1) & 1 & 0 \\ \mathbf{C}_1(2) & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ f_2(2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ f_2(2) \end{pmatrix} = f(\mathbf{x}),$$

then

$$f(\mathbf{x}) - \mathbf{BCf(x)} = f(\mathbf{x}) - \begin{pmatrix} 0 \\ \mathbf{I} \end{pmatrix} [\mathbf{C}_1 \quad \mathbf{I}] f(\mathbf{x}) = 0$$

and

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{Ax} + f(\mathbf{x}) + \mathbf{Bu} = \mathbf{Ax} + f(\mathbf{x}) + \mathbf{B}(-\mathbf{CAx} - \mathbf{Cf(x)}) \\ &= \mathbf{Ax} + f(\mathbf{x}) - \mathbf{BCAx} - \mathbf{BCf(x)} = (\mathbf{A} - \mathbf{BCA})\mathbf{x} \end{aligned}$$

Define  $\mathbf{M} = \mathbf{A} - \mathbf{BCA}$ , then  $\dot{\mathbf{x}} = \mathbf{Mx}$ .

To guarantee  $\dot{\mathbf{x}} \rightarrow 0$  exponentially, design the Lyapunov function as

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{x}$$

then

$$\begin{aligned}\dot{V}(\mathbf{x}) &= (\mathbf{M}\mathbf{x})^T \mathbf{x} + \mathbf{x}^T \mathbf{M}\mathbf{x} = \mathbf{x}^T \mathbf{M}^T \mathbf{x} + \mathbf{x}^T \mathbf{M}\mathbf{x} \\ &= \mathbf{x}^T (\mathbf{M}^T + \mathbf{M})\mathbf{x}\end{aligned}$$

To guarantee  $\dot{V}(\mathbf{x}) \leq 0$ , choose

$$\mathbf{M}^T + \mathbf{M} < 0. \quad (5.30)$$

The matrix  $\mathbf{C}$  can be solved by LMI (5.30). However, in Eq. (5.30),  $\mathbf{M} = 0$  cannot be avoided, thus  $\mathbf{M}^T + \mathbf{M} < 0$  cannot be satisfied, sliding mode function based on dynamic compensation should be designed.

### 5.6.3 Sliding mode control based on dynamic compensation

Define the sliding mode function as

$$\mathbf{s} = \mathbf{Cx} + \mathbf{z}, \quad (5.31)$$

where  $\mathbf{C} = [\mathbf{C}_1 \quad \mathbf{I}] = \begin{bmatrix} \mathbf{C}_1(1) & 1 & 0 \\ \mathbf{C}_1(2) & 0 & 1 \end{bmatrix}$ ,  $\mathbf{C}_1 \in \mathbb{R}^{2 \times 1}$  and  $\mathbf{I}$  is a unit matrix with  $2 \times 2$ .

To realize  $\mathbf{x} \rightarrow 0$  exponentially, compensation is designed as follows [3]:

$$\dot{\mathbf{z}} = \mathbf{Kx} - \mathbf{z}, \quad (5.32)$$

where  $\mathbf{z} \in \mathbb{R}^2$  is the state of the compensator,  $\mathbf{K} \in \mathbb{R}^{2 \times 3}$ .

In Eqs. (5.31) and (5.32),  $\mathbf{C}$  and  $\mathbf{K}$  can be solved by LMI.

Choose the Lyapunov function as

$$V(t) = \frac{1}{2} \mathbf{s}^T \mathbf{s}$$

Since

$$\dot{\mathbf{s}} = \mathbf{C}\dot{\mathbf{x}} + \dot{\mathbf{z}} = \mathbf{C}(\mathbf{Ax} + \mathbf{f}(\mathbf{x}) + \mathbf{Bu}) + \mathbf{Kx} - \mathbf{z}$$

then

$$\dot{V}(t) = \mathbf{s}^T \dot{\mathbf{s}} = \mathbf{s}^T (\mathbf{CAx} + \mathbf{Cf(x)} + \mathbf{CBu} + \mathbf{Kx} - \mathbf{z}).$$

Design the controller as

$$\mathbf{u} = -\mathbf{CAx} - \mathbf{Cf(x)} - \mathbf{Kx} + \mathbf{z} - \eta \mathbf{s}, \quad (5.33)$$

where  $\eta > 0$ .

Consider  $\mathbf{C}\mathbf{B} = \mathbf{I}$ , then

$$\begin{aligned}\dot{V}(t) &= \mathbf{s}^T \dot{\mathbf{s}} = \mathbf{s}^T ((\mathbf{C}\mathbf{A}\mathbf{x} + \mathbf{C}f(\mathbf{x}) + \mathbf{C}\mathbf{B}\mathbf{u}) + \mathbf{K}\mathbf{x} - \mathbf{z}) \\ &= \mathbf{s}^T (\mathbf{C}\mathbf{A}\mathbf{x} + \mathbf{C}f(\mathbf{x}) + \mathbf{u} + \mathbf{K}\mathbf{x} - \mathbf{z}) = \mathbf{s}^T (-\eta \mathbf{s}) \\ &= -\eta \|\mathbf{s}\| \leq 0\end{aligned}$$

Since

$$\begin{aligned}\mathbf{f}(\mathbf{x}) - \mathbf{B}\mathbf{C}\mathbf{f}(\mathbf{x}) &= \mathbf{f}(\mathbf{x}) - \begin{pmatrix} 0 \\ \mathbf{I} \end{pmatrix} [\mathbf{C}_1 \quad \mathbf{I}] \mathbf{f}(\mathbf{x}), \\ \begin{pmatrix} 0 \\ \mathbf{I} \end{pmatrix} [\mathbf{C}_1 \quad \mathbf{I}] \mathbf{f}(\mathbf{x}) &= \begin{pmatrix} 0 & \mathbf{0} \\ \mathbf{C}_1 & \mathbf{I} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ f_2(2) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \mathbf{C}_1(1) & 1 & 0 \\ \mathbf{C}_1(2) & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ f_2(2) \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \\ f_2(2) \end{pmatrix} = \mathbf{f}(\mathbf{x}),\end{aligned}$$

then

$$\mathbf{f}(\mathbf{x}) - \mathbf{B}\mathbf{C}\mathbf{f}(\mathbf{x}) = 0$$

and

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u} = \mathbf{A}\mathbf{x} + \mathbf{f}(\mathbf{x}) + \mathbf{B}(-\mathbf{C}\mathbf{A}\mathbf{x} - \mathbf{C}f(\mathbf{x}) - \mathbf{K}\mathbf{x} + \mathbf{z}) \\ &= \mathbf{A}\mathbf{x} + (\mathbf{f}(\mathbf{x}) - \mathbf{B}\mathbf{C}\mathbf{f}(\mathbf{x})) - \mathbf{B}\mathbf{C}\mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}\mathbf{x} + \mathbf{B}\mathbf{z} \\ &= (\mathbf{A} - \mathbf{B}\mathbf{C}\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} + \mathbf{B}\mathbf{z}\end{aligned}.$$

Since  $\mathbf{s}^T \dot{\mathbf{s}} \leq 0$ , there exists  $t > t_0$ , we have  $\mathbf{s} = 0$ , i.e.,  $\mathbf{z} = -\mathbf{C}\mathbf{x}$ , then

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\mathbf{C}\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} + \mathbf{B}(-\mathbf{C}\mathbf{x}) = (\mathbf{A} - \mathbf{B}(\mathbf{K} + \mathbf{C} + \mathbf{CA}))\mathbf{x}. \quad (5.34)$$

Define  $\mathbf{M} = \mathbf{A} - \mathbf{B}(\mathbf{K} + \mathbf{C} + \mathbf{CA})$ , then  $\dot{\mathbf{x}} = \mathbf{M}\mathbf{x}$ .

To guarantee  $\mathbf{x} \rightarrow 0$  exponentially, choose the Lyapunov function as

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{x}$$

Then

$$\begin{aligned}\alpha V(\mathbf{x}) + \dot{V}(\mathbf{x}) &= \alpha \mathbf{x}^T \mathbf{x} + (\mathbf{M}\mathbf{x})^T \mathbf{x} + \mathbf{x}^T \mathbf{M}\mathbf{x} \\ &= \alpha \mathbf{x}^T \mathbf{x} + \mathbf{x}^T \mathbf{M}^T \mathbf{x} + \mathbf{x}^T \mathbf{M}\mathbf{x}, \\ &= \mathbf{x}^T (\alpha \mathbf{I} + \mathbf{M}^T + \mathbf{M}) \mathbf{x}\end{aligned}$$

where  $\alpha > 0$ .

To guarantee  $\dot{V}(x) \leq 0$ , choose

$$\alpha I + M^T + M < 0. \quad (5.35)$$

Matrixes  $C$  and  $K$  can be solved by Eq. (5.35).

From  $\alpha V + \dot{V} \leq 0$ , using Lemma 1.3, we have  $V(t) \leq V(0)\exp(-\alpha t)$ , i.e., if  $t \rightarrow \infty$ ,  $V(t) \rightarrow 0$  and  $x \rightarrow 0$ , the system states converge to zero exponentially.

#### 5.6.4 Simulation example

Firstly, we test the Lorenz model (5.26); choose  $u_1 = u_2 = 0$  and  $x(0) = [0 \ -1 \ 0]$ . Simulation results are given in Fig. 5.5.

Then considering Lorenz system (5.26), we use LMI program chap5\_8lmi.m. The characteristic value of  $M^T + M = A^T - A^T C^T B^T + A - BCA$  can be obtained as

$$\begin{bmatrix} -40 \\ 0 \\ 0 \end{bmatrix}.$$

It can be seen that  $M^T + M < 0$  cannot be guaranteed if we use Eq. (5.36) to design the sliding mode function.

If we use Eqs. (5.31) and (5.32) to design the sliding mode function, dynamic compensation  $\dot{z} = Kx - z$  is added in the sliding mode function design, use program chap5\_8dylmi.m, let  $M = A - B(K + C + CA)$ , characteristic value of  $M^T + M$  is

$$\begin{bmatrix} -37.2159 \\ -37.2159 \\ -20 \end{bmatrix}$$

It can be seen by using sliding mode function with dynamic compensation, the closed system Hurwitz conditions can be guaranteed.

Consider system (5.26),  $x(0) = [0 \ -1 \ 0]$ , let  $\alpha = 10$ . We can get  $C$  and  $K$  from Eq. (5.35) by using LMI program chap5\_8dylmi.m as follows:

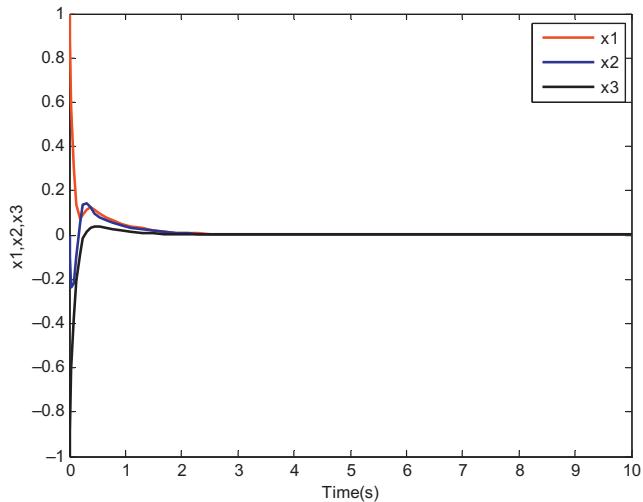
$$C = \begin{bmatrix} -9.2296 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K = \begin{bmatrix} -73.0663 & 109.9039 & 0 \\ 0 & 0 & 17.608 \end{bmatrix}.$$

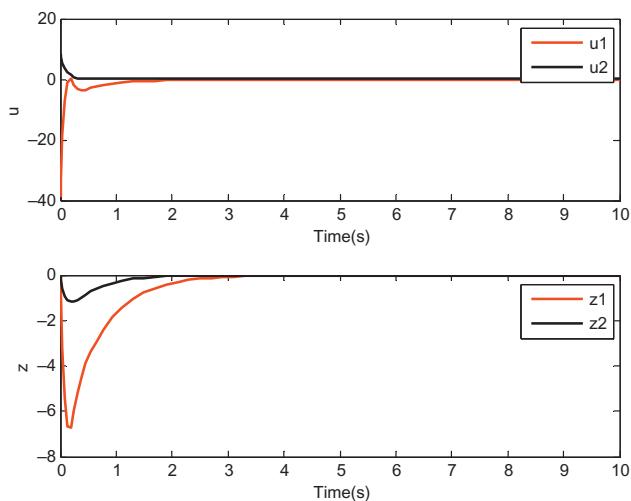
Then

$$s = \mathbf{C}\mathbf{x} + z = \begin{bmatrix} -9.2296x_1 + x_2 \\ x_3 \end{bmatrix} + z.$$

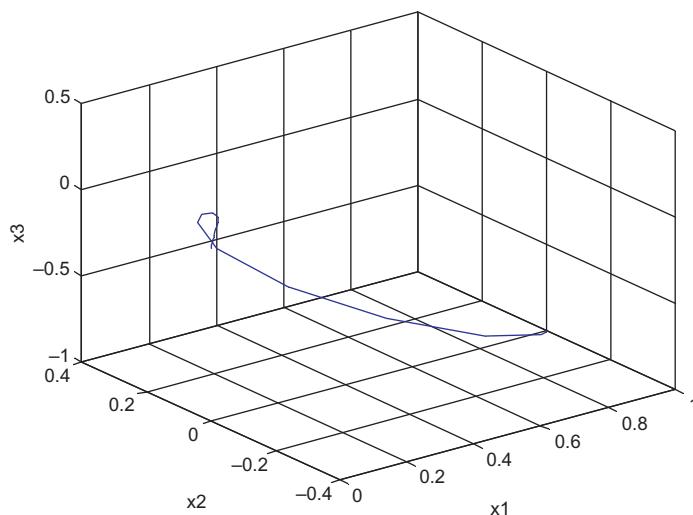
Use controller (5.33) and set  $\eta = 1.0$ . Simulation results are given in Figs. 5.11–5.13.



■ FIGURE 5.11 System states response.



■ FIGURE 5.12 Control input.

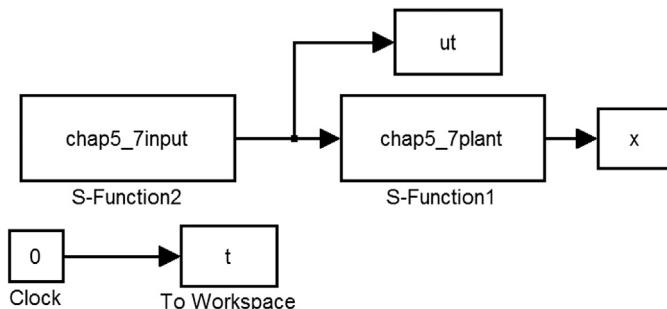


■ FIGURE 5.13 System states response.

Simulation programs:

First simulation: model test

1. Simulink main program: chap5\_7sim.mdl



2. Plant S function: chap5\_7plant.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);

```

```

case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[1,0,-1];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
u1=u(1);
u2=u(2);

a = 10; b = 8/3; r = 28;

sys(1)=a*(x(2)-x(1));
sys(2)=r*x(1)-x(2)-x(1)*x(3)+u1;
sys(3)=-b*x(3)+x(1)*x(2)+u2;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);

```

**3. Plot program: chap5\_7plot.m**

```

close all;

x1=x(:,1);
x2=x(:,2);
x3=x(:,3);

plot3(x3,x2,x1);
xlabel('x3'); ylabel('x2'); zlabel('x1');

```

**Second simulation:** Sliding mode control based on dynamic compensation

**1. LMI design program**

```
LMI design program: chap5_8LMI.m
clear all;
close all;

a = 10; b = 8/3; r = 28;

A = [-a a 0;
 r -1 0;
 0 0 -b];
B = [0 0;
 1 0;
 0 1];

M = sdpvar(3,3);
C1 = sdpvar(2,1);
C = [C1, eye(2)];
M = A - B*C*A;
F = set((M + M') < 0);

solvesdp(F);
M = double(M)

display('the eigenvalues of M is ');
eig(M)
display('the eigenvalues of M+MT is ');
eig(M + M')

C = double(C)
LMI with dynamic compensation program: chap5_8dyLMI.m
clear all;
close all;

a = 10; b = 8/3; r = 28;

A = [-a a 0;
 r -1 0;
 0 0 -b];
B = [0 0;
 1 0;
 0 1];
M = sdpvar(3,3);
```

```

C1 = sdpvar(2,1);
K = sdpvar(2,3);

C = [C1,eye(2)];

M = A - B*C*A - B*K - B*C;

F = set((M + M') < 0);

solvesdp(F);

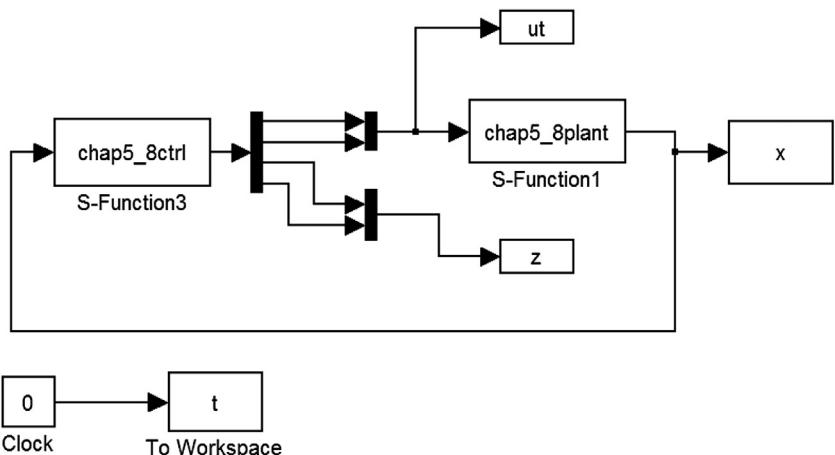
M = double(M)

display('the eigenvalues of M is ');
eig(M)
display('the eigenvalues of M+MT is ');
eig(M+M')

C = double(C)
K = double(K)

```

**2. Simulink main program: chap5\_8sim.mdl**



**3. Controller S function: chap5\_8ctrl.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);

```

```

case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
x1=u(1);
x2=u(2);
x3=u(3);
xp=[x1 x2 x3]';
z=[x(1) x(2)]';
C=[0.0497 1.0000 0;
 0 0 1.0000];
K=[10.4476 8.9989 0;
 0 0 9.4963];
dz=K*xp-z;
sys(1)=dz(1);
sys(2)=dz(2);
function sys=mdlOutputs(t,x,u)
z=[x(1) x(2)]';
x1=u(1);
x2=u(2);
x3=u(3);
xp=[x1 x2 x3]';
a=10;b=8/3;r=28;

```

```

f = [0 -x1*x3 x1*x2]';

A = [-a a 0;
 r -1 0;
 0 0 -b];

C = [0.0497 1.0000 0;
 0 0 1.0000];
K = [10.4476 8.9989 0;
 0 0 9.4963];

S = C*xp + z;

M = 1;
if M == 1
 xite = 1.5;
 ut = -C*A*xp - C*f - K*xp + z - xite*S;
elseif M == 2
 alfa1 = 0.50; niu = 1.0; xitel = 2.0;
 Xite = niu + xitel*(norm(S))^(alfa1-1);
 ut = -C*A*xp - C*f - K*xp + z - Xite*S;
end

```

```

sys(1) = ut(1);
sys(2) = ut(2);
sys(3) = z(1);
sys(4) = z(2);

```

**4. Plant S function: chap5\_8plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;

```

```

sizes.NumOutputs = 3;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[1,0,-1];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
u1=u(1);
u2=u(2);

a=10;b=8/3;r=28;

sys(1)=a*(x(2)-x(1));
sys(2)=r*x(1)-x(2)-x(1)*x(3)+u1;
sys(3)=-b*x(3)+x(1)*x(2)+u2;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);

```

**5. Plot program: chap5\_8plot.m**

```

close all;
figure(1);
plot(t,x(:,1),'r',t,x(:,2),'b',t,x
(:,3),'k','linewidth',2);
xlabel('time(s)');ylabel('x1,x2,x3');
legend('x1','x2','x3');

figure(2);
subplot(211);
plot(t,ut(:,1),'r',t,ut(:,2),'k','linewidth',2);
xlabel('time(s)');ylabel('u');
legend('u1','u2');
subplot(212);
plot(t,z(:,1),'r',t,z(:,2),'k','linewidth',2);
xlabel('time(s)');ylabel('z');
legend('z1','z2');

figure(3);
x1=x(:,1);
x2=x(:,2);
x3=x(:,3);
plot3(x1,x2,x3);
xlabel('x1');ylabel('x2');zlabel('x3');

```

```
grid on;

display('the last x is');
G=size(x,1);
x(G,:)
```

## REFERENCES

- [1] J. Löfberg, YALMIP: a free MATLAB Toolbox for rapid prototyping of optimization problems, <[http://www.mathworks.com/matlabcentral/newsreader/view\\_thread/59636](http://www.mathworks.com/matlabcentral/newsreader/view_thread/59636)>.
- [2] M. Rehan, K.-S. Hong, S.S. Ge, Stabilization and tracking control for a class of nonlinear systems, Nonlinear Anal. Real World Appl. 12 (2011) 1786–1796.
- [3] H. Wang, Z. Han, Q. Xie, W. Zhang, Sliding mode control for chaotic systems based on LMI, Commun. Nonlinear Sci. Numer. Simul. 14 (2009) 1410–1417.

# Sliding mode control based on the RBF neural network

Past research of the universal approximation theorem [1] has shown that any nonlinear function over a compact set with arbitrary accuracy can be approximated by the RBF neural network. There have been significant research efforts on the RBF neural control for nonlinear systems.

The unknown nonlinearities in the system can be approximated by the RBF neural network whose weight value parameters are adjusted online according to adaptive laws for the purpose of controlling the output of the nonlinear system to track a given trajectory. The Lyapunov function is used to develop adaptive control algorithm based on the RBF model. The chattering action is decreased.

In the books [2,3], we have designed a sliding mode controller based on RBF for several kinds of systems. In this section, we will design a sliding mode controller based on RBF for other kinds of systems.

In [Section 6.1](#), a simple adaptive sliding mode control based on RBF is introduced, unknown function  $f(x)$  is approximated by the RBF neural network.

In [Section 6.2](#), an adaptive sliding mode control based on RBF compensation is discussed, to improve control performance, we design the RBF neural network to compensate disturbance acted on the control input.

In [Section 6.3](#), sliding mode control based on the RBF neural network with minimum parameter learning method is introduced, minimum parameter learning can be used in the RBF neural network to reduce the computational burden and increase real-time performance [4]. Using the minimum parameter learning, in this section, a sliding mode controller based on the RBF neural network is designed.

In [Section 6.4](#), based on [Section 6.3](#), sliding mode controller based on RBF with MPL is designed for manipulators.

## 6.1 A SIMPLE ADAPTIVE SLIDING MODE CONTROL BASED ON RBF

### 6.1.1 System description

Consider a simple dynamic system as

$$\ddot{\theta} = f(\theta, \dot{\theta}) + u, \quad (6.1)$$

where  $\theta$  is angle,  $u$  is control input.

Eq. (6.1) can be written as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + u,\end{aligned} \quad (6.2)$$

where  $f(x)$  is an unknown function.

Let the desired output be  $x_d$ , and denote

$$e = x_1 - x_d, \quad \dot{e} = x_2 - \dot{x}_d.$$

Define the sliding mode function as

$$s = ce + \dot{e}, \quad c > 0, \quad (6.3)$$

then

$$\dot{s} = c\dot{e} + \ddot{e} = c\dot{e} + \dot{x}_2 - \ddot{x}_d = c\dot{e} + f(x) + u - \ddot{x}_d.$$

From Eq. (6.3), we can see that if  $s \rightarrow 0$ , then  $e \rightarrow 0$  and  $\dot{e} \rightarrow 0$ .

### 6.1.2 RBF neural network approximation

RBF networks are often used to approximate any unknown function [1]. The algorithms of RBF networks are:

$$h_j = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2b_j^2}\right), \quad (6.4)$$

$$f = \mathbf{W}^{*\top} \mathbf{h}(\mathbf{x}) + \varepsilon, \quad (6.5)$$

where  $\mathbf{x}$  is the input signal of the network,  $i$  is the input number of the network,  $j$  is the number of hidden layer nodes in the network,  $\mathbf{h} = [h_1, h_2, \dots, h_n]^T$  is the output of Gaussian function,  $\mathbf{W}^*$  is ideal neural network weight value,  $\varepsilon$  is approximation error of neural network, and  $\varepsilon \leq \varepsilon_N$ .

If we use the RBF network to approximate  $f(x)$ , the network input is selected as  $\mathbf{x} = [x_1 \ x_2]^T$ , and output of the RBF neural network is

$$\hat{f}(x) = \hat{\mathbf{W}}^T \mathbf{h}(x). \quad (6.6)$$

### 6.1.3 Sliding mode controller design

Define the Lyapunov function as

$$V = \frac{1}{2}s^2 + \frac{1}{2\gamma}\tilde{\mathbf{W}}^T\tilde{\mathbf{W}}, \quad (6.7)$$

where  $\gamma > 0$ ,  $\tilde{\mathbf{W}} = \hat{\mathbf{W}} - \mathbf{W}^*$ .

Since  $f(x) - \hat{f}(x) = \mathbf{W}^{*\top}\mathbf{h}(x) + \varepsilon - \hat{\mathbf{W}}^T\mathbf{h}(x) = -\tilde{\mathbf{W}}^T\mathbf{h}(x) + \varepsilon$ , then

$$\dot{V} = s\dot{s} + \frac{1}{\gamma}\tilde{\mathbf{W}}^T\dot{\tilde{\mathbf{W}}} = s(c\dot{e} + f(x) + u - \ddot{x}_d) + \frac{1}{\gamma}\tilde{\mathbf{W}}^T\dot{\tilde{\mathbf{W}}}$$

Design the sliding mode controller as

$$u = -c\dot{e} - \hat{f}(x) + \ddot{x}_d - \eta \text{sgn}(s), \quad (6.8)$$

then

$$\begin{aligned} \dot{V} &= s(f(x) - \hat{f}(x) - \eta \text{sgn}(s)) + \frac{1}{\gamma}\tilde{\mathbf{W}}^T\dot{\tilde{\mathbf{W}}} \\ &= s(-\tilde{\mathbf{W}}^T\mathbf{h}(x) + \varepsilon - \eta \text{sgn}(s)) + \frac{1}{\gamma}\tilde{\mathbf{W}}^T\dot{\tilde{\mathbf{W}}} \\ &= \varepsilon s - \eta|s| + \tilde{\mathbf{W}}^T\left(\frac{1}{\gamma}\dot{\tilde{\mathbf{W}}} - s\mathbf{h}(x)\right) \end{aligned}$$

Design the adaptive law as

$$\dot{\tilde{\mathbf{W}}} = \gamma s\mathbf{h}(x). \quad (6.9)$$

If we choose  $\eta > |\varepsilon|_{\max}$ , then  $\dot{V} = \varepsilon s - \eta|s| \leq 0$ .

From the above analysis, we can see that RBF approximation error can be overcome by the robust term  $\eta \text{sgn}(s)$ .

When  $\dot{V} \equiv 0$ , we have  $s \equiv 0$ , according to LaSalle invariance principle, the closed system is asymptotically stable,  $s \rightarrow 0$  as  $t \rightarrow \infty$ , and the convergence precision is related to  $\eta$ .

Since  $V \geq 0$ ,  $\dot{V} \leq 0$ ,  $V$  is limited as  $t \rightarrow \infty$ , thus  $\hat{\mathbf{W}}$  is limited. Since when  $\dot{V} \equiv 0$ , we cannot get  $\tilde{\mathbf{W}} \equiv 0$ ,  $\hat{\mathbf{W}}$  cannot converge to  $\mathbf{W}^*$ .

### 6.1.4 Simulation example

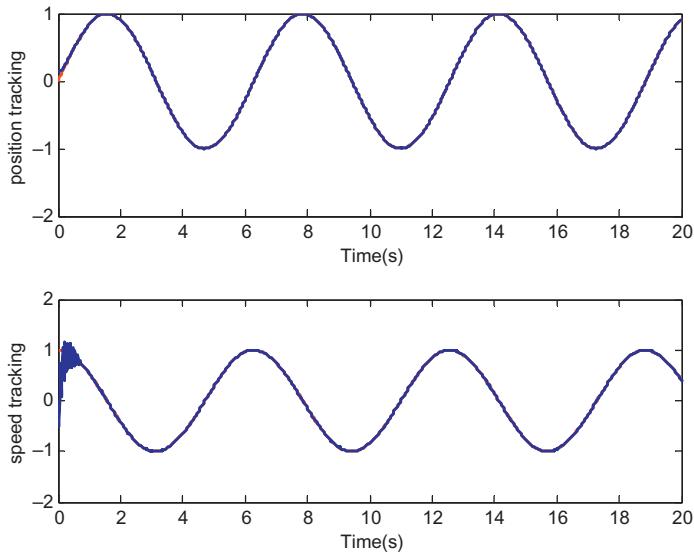
Consider a plant as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + u \end{aligned}$$

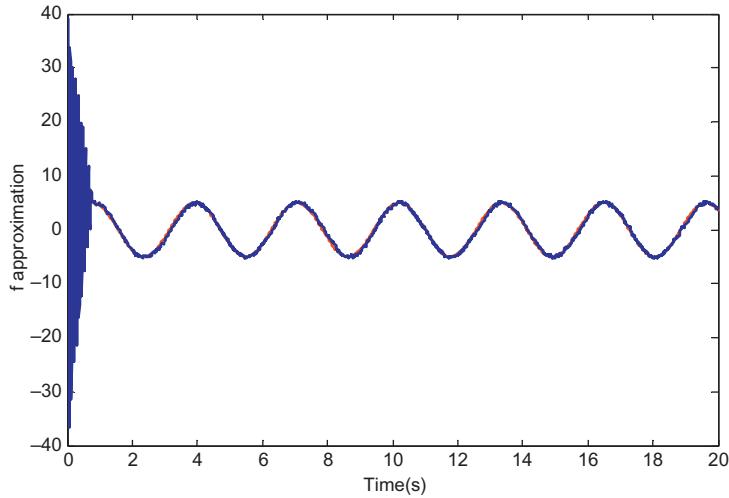
where  $f(x) = 10x_1x_2$ .

The desired trajectory is chosen as  $x_d = \sin t$ . The initial state of the plant is set as  $[0.15 \ 0]$ . We adapt control law as Eq. (6.8) and adaptive law as Eq. (6.9), choose  $c = 200$ ,  $\eta = 0.50$  and  $\gamma = 500$ .

The structure of RBF is chosen as 2-5-1. Consider the range of  $x_1$  and  $x_2$ , we choose  $c_i = [-2 \ -1 \ 0 \ 1 \ 2]$ ,  $b_j = 3.0$ , the initial value of each element of RBF weight matrix is set as 0.10. Simulation results are shown in Figs. 6.1 and 6.2.



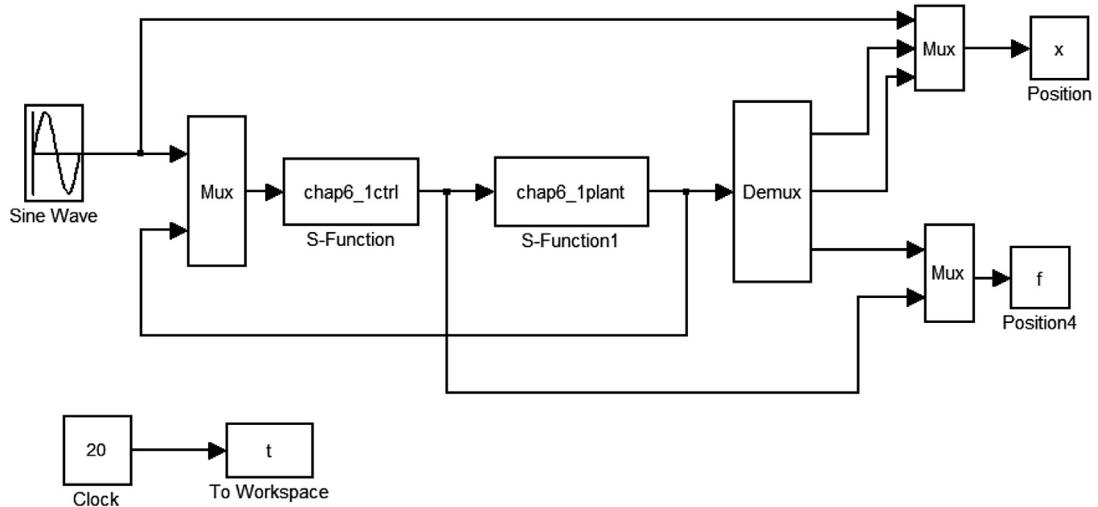
■ FIGURE 6.1 Position and speed tracking.



■ FIGURE 6.2  $f(x)$  and  $\hat{f}(x)$ .

Simulation Programs:

1. Main program: chap6\_1sim.mdl



2. S function of controller: chap6\_1ctrl.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
global b c lama
sizes=simsizes;
sizes.NumContStates = 5;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 4;

```

```
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = 0.1*ones(1,5);
str = [];
ts = [0 0];
c = 0.5*[-2 -1 0 1 2;
 -2 -1 0 1 2];
b = 3.0;
lama = 10;
function sys = mdlDerivatives(t,x,u)
global b c lama
xd = sin(t);
dxd = cos(t);

x1 = u(2);
x2 = u(3);
e = x1 - xd;
de = x2 - dxd;
s = lama*e + de;

W = [x(1) x(2) x(3) x(4) x(5)]';
xi = [x1;x2];

h = zeros(5,1);
for j = 1:1:5
 h(j) = exp(-norm(xi - c(:,j))^2/(2*b^2));
end

gama = 1500;
for i = 1:1:5
 sys(i) = gama*s*h(i);
end

function sys = mdlOutputs(t,x,u)
global b c lama
xd = sin(t);
dxd = cos(t);
ddxd = -sin(t);

x1 = u(2);
x2 = u(3);
e = x1 - xd;
```

```

de = x2 - dxd;
s = lama*e + de;

W=[x(1) x(2) x(3) x(4) x(5)];
xi=[x1;x2];

h=zeros(5,1);
for j=1:1:5
 h(j)=exp(-norm(xi-c(:,j))^2/(2*b^2));
end
fn=W*h;
xite=1.50;

%fn=10*x1+x2; %Precise f
ut=-lama*de+ddxd-fn-xite*sign(s);

sys(1)=ut;
sys(2)=fn;

```

**3. S function of plant: chap6\_1plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=2;
sizes.NumDiscStates=0;
sizes.NumOutputs=3;
sizes.NumInputs=2;
sizes.DirFeedthrough=0;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[0.15;0];

```

```

str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
f=10*x(1)*x(2);
sys(1)=x(2);
sys(2)=f+ut;
function sys=mdlOutputs(t,x,u)
f=10*x(1)*x(2);

sys(1)=x(1);
sys(2)=x(2);
sys(3)=f;
4. Plot program: chap6_1plot.m
close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r',t,x(:,2),'b','linewidth',2);
xlabel('time(s)');ylabel('position tracking');
subplot(212);
plot(t,cos(t),'r',t,x(:,3),'b','linewidth',2);
xlabel('time(s)');ylabel('speed tracking');

figure(2);
plot(t,f(:,1),'r',t,f(:,3),'b','linewidth',2);
xlabel('time(s)');ylabel('f approximation');

```

## 6.2 ADAPTIVE SLIDING MODE CONTROL BASED ON RBF COMPENSATION

### 6.2.1 System description

Consider a plant as

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x, t) + b(u - \delta(x_2)) + dt \end{cases} \quad (6.10)$$

where  $b > 0$ ,  $dt$  is disturbance,  $|dt| \leq D$ ,  $u$  is control input and  $\delta(x_2)$  is friction force.

The desired angle signal is set as  $x_d$ , and the control goals are  $x_1 \rightarrow x_d$  and  $x_2 \rightarrow \dot{x}_d$ . To improve control performance, we can design the RBF neural network to compensate  $\delta(x_2)$ .

### 6.2.2 Sliding mode control based on RBF compensation

The algorithm of the RBF networks is

$$\begin{aligned} h_j &= \exp\left(\frac{\|x - c_j\|^2}{2b_j^2}\right), \\ \delta &= W^{*T}h(x) + \varepsilon \end{aligned} \quad (6.11)$$

where  $x$  is the input signal of the network,  $i$  is the input number of the network,  $j$  is the number of hidden layer nodes in the network,  $h = [h_1, h_2, \dots, h_n]^T$  is the output of Gaussian function,  $W^*$  is ideal neural network weight value,  $\varepsilon$  is approximation error of neural network, and  $\varepsilon \leq \varepsilon_{\max}$ ,  $\hat{\delta}$  is output of RBF,

If we use the RBF network to approximate  $f(x)$ , the network input is selected as  $x = [x_1 \ x_2]^T$ , and output of the RBF neural network is

Consider  $x = x_2$  as input of RBF, we can design the RBF neural network to approximate  $\delta(x_2)$ , then the output of RBF is

$$\hat{\delta} = \hat{W}^T h. \quad (6.12)$$

Define  $\tilde{W} = \hat{W} - W^*$ , then

$$\delta - \hat{\delta} = W^{*T}h + \varepsilon - \hat{W}^T h = (W^{*T} - \hat{W}^T)h + \varepsilon = -\tilde{W}^T h + \varepsilon.$$

Define the tracking error as  $e = x_1 - x_d$ , then  $\dot{e} = \dot{x}_1 - \dot{x}_d$ , and define the sliding mode function as

$$s = ce + \dot{e}, \quad (6.13)$$

where  $c > 0$ .

Then

$$\begin{aligned} \dot{s} &= c\dot{e} + \ddot{e} = c\dot{e} + \ddot{x}_1 - \ddot{x}_d = c\dot{e} + f + b(u - \delta) + dt - \ddot{x}_d \\ &= c\dot{e} + f + b(u - \delta) + dt - \ddot{x}_d \end{aligned}$$

Design the sliding mode controller as

$$u = \frac{1}{b}(-c\dot{e} - f + \ddot{x}_d - \eta \text{sgn}(s)) + \hat{\delta}, \quad (6.14)$$

where  $\eta \geq D + b\varepsilon_{\max}$ .

Then

$$\dot{s} = -\eta \text{sgn}(s) + b(\hat{\delta} - \delta) + dt = -\eta \text{sgn}(s) + b(\tilde{W}^T h - \varepsilon) + dt.$$

Design the Lyapunov function as

$$L = \frac{1}{2}s^2 + \frac{1}{2}\gamma\tilde{\mathbf{W}}^T\tilde{\mathbf{W}}, \quad (6.15)$$

where  $\gamma > 0$ .

Then

$$\begin{aligned} \dot{V} &= s\dot{s} + \gamma\tilde{\mathbf{W}}^T\dot{\tilde{\mathbf{W}}} = -\eta|s| + s\cdot dt + sb(\tilde{\mathbf{W}}^T\mathbf{h} - \varepsilon) + \gamma\tilde{\mathbf{W}}^T\dot{\tilde{\mathbf{W}}} \\ &= -\eta|s| + s\cdot dt - sb\varepsilon + \tilde{\mathbf{W}}^T(sb\mathbf{h} + \gamma\dot{\tilde{\mathbf{W}}}) \end{aligned}$$

Design the adaptive law as

$$\dot{\tilde{\mathbf{W}}} = -\frac{1}{\gamma}sb\mathbf{h}. \quad (6.16)$$

Then

$$\dot{V} = -\eta|s| + (dt - b\varepsilon)s \leq 0$$

When  $\dot{V} \equiv 0$ , we have  $s \equiv 0$ , according to LaSalle invariance principle, the closed system is asymptotically stable,  $s \rightarrow 0$  as  $t \rightarrow \infty$ , and the convergence precision is related to  $\eta$ .

Since  $V \geq 0$ ,  $\dot{V} \leq 0$ ,  $V$  is limited as  $t \rightarrow \infty$ , thus  $\hat{\mathbf{W}}$  is limited. Since when  $\dot{V} \equiv 0$ , we cannot get  $\tilde{\mathbf{W}} \equiv 0$ ,  $\hat{\mathbf{W}}$  cannot converge to  $\mathbf{W}^*$ .

### 6.2.3 Simulation example

Consider a plant as

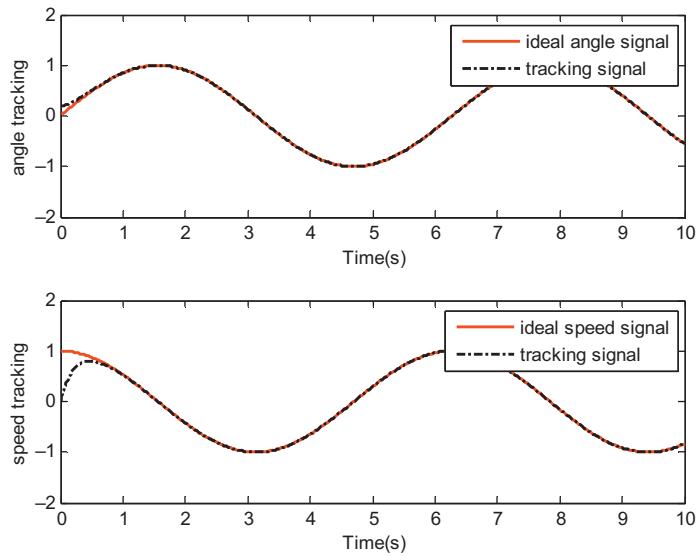
$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -25x_2 + 133(u - \delta(x_2)) + 10\sin t \end{cases}$$

where  $\delta(x_2)$  is considered as friction model and  $\delta(x_2) = x_2 + 0.01\text{sgn}(x_2)$ .

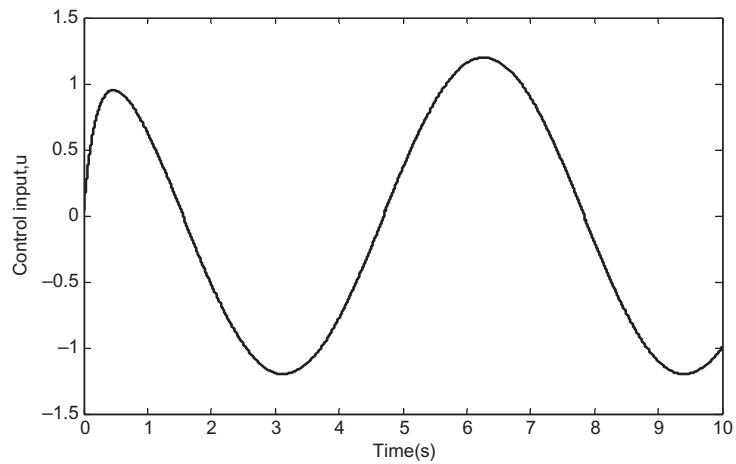
Consider Eq. (6.10), we have  $f(x, t) = -25x_2$ ,  $b = 133$ ,  $D = 10$ . The desired trajectory is  $x_d = \sin t$ . The initial states of the plant set as  $[1, 0]$ .

We adapt the control law as Eq. (6.14) and the adaptive law as Eq. (6.16), choose  $c = 5$ ,  $\eta = D + 0.5$  and  $\gamma = 0.10$ . In control law (6.14), the saturation function is used instead of the switch function, and  $\Delta = 0.02$ .

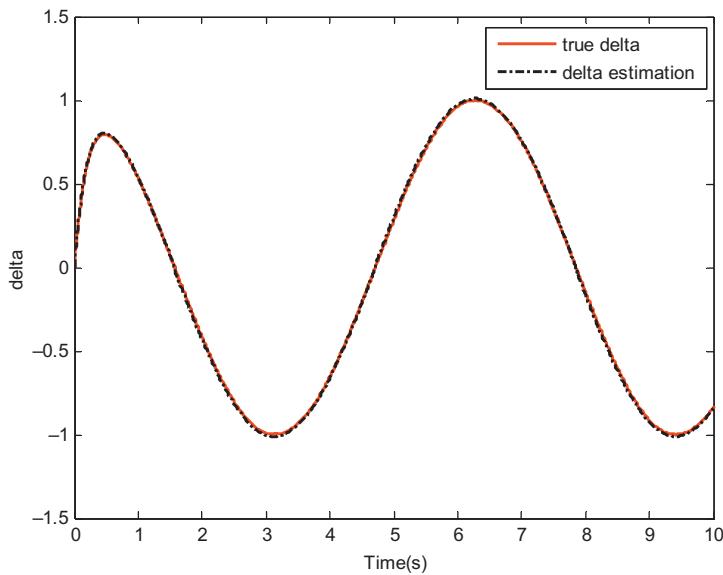
The structure of RBF is chosen as 1-5-1. Consider the range of  $x_2$ ; we choose  $\mathbf{c}_i = [-1.0 \quad -0.5 \quad 0 \quad 0.5 \quad 1.0]$ ,  $b_j = 1.0$ , the initial value of each element of RBF weight matrix is set as 0.0. Simulation results are shown in Figs. 6.3–6.5.



■ FIGURE 6.3 Angle and angle speed tracking.



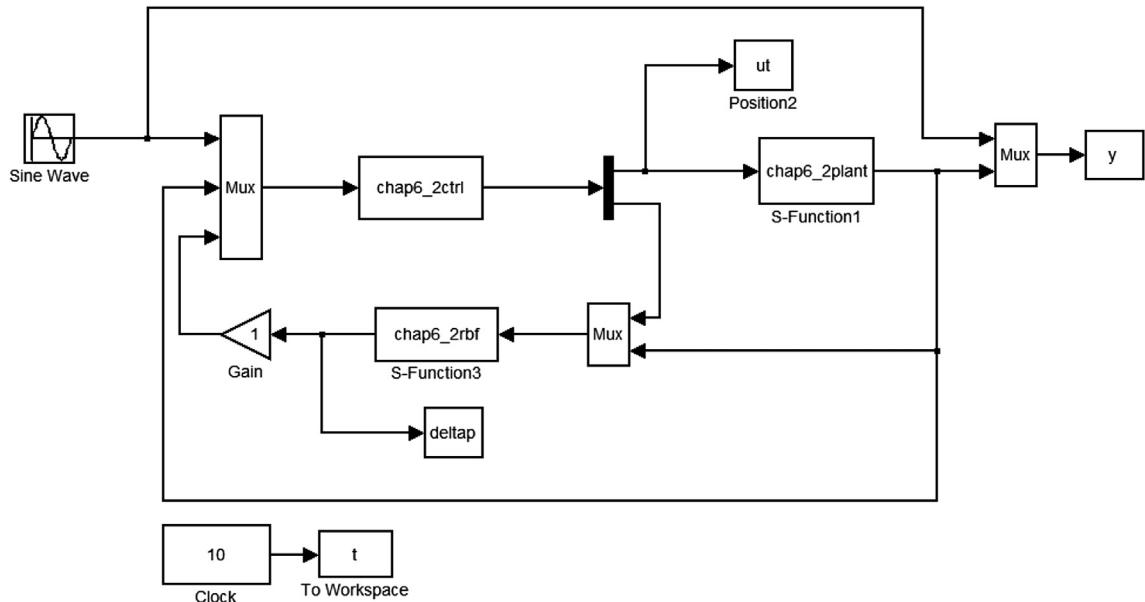
■ FIGURE 6.4 Control input.



■ FIGURE 6.5  $\delta(x_2)$  and  $\hat{\delta}(x_2)$ .

#### Simulation Programs:

##### 1. Main program: chap6\_2sim.mdl.m



**2. S function of controller: chap6\_2ctrl.m**

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];
function sys=mdlOutputs(t,x,u)
x1d=sin(t);
dx1d=cos(t);
ddx1d=-sin(t);
x1=u(2);
x2=u(3);
deltap=u(5);

e=x1-x1d;
de=x2-dx1d;
c=5;
s=c*e+de;

D=1.0;
xite=D+0.50;
fai=0.02;
```

```

 if abs(s)<=fai
 sat=s/fai;
 else
 sat=sign(s);
 end
 b=133;
 f=-25*x2;
 ut=1/b*(-c*de-f+ddx1d-xite*sat)+deltap;
 sys(1)=ut;
 sys(2)=s;
 3. RBF Approximation : chap6_2rbf.m
 function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
 switch flag,
 case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
 case 1,
 sys=mdlDerivatives(t,x,u);
 case 3,
 sys=mdlOutputs(t,x,u);
 case {2,4,9}
 sys=[];
 otherwise
 error(['Unhandled flag = ',num2str(flag)]);
 end
 function [sys,x0,str,ts]=mdlInitializeSizes
 global cij bj c
 sizes=simsizes;
 sizes.NumContStates=5;
 sizes.NumDiscStates=0;
 sizes.NumOutputs=1;
 sizes.NumInputs=4;
 sizes.DirFeedthrough=1;
 sizes.NumSampleTimes=0;
 sys=simsizes(sizes);
 x0=0*ones(1,5);
 str=[];
 ts=[];
 cij=1*[-1 -0.5 0 0.5 1];
 bj=1.0;
 function sys=mdlDerivatives(t,x,u)
 global cij bj

```

```

s = u(1);
x2 = u(3);

b = 133;

xi = x2;
h = zeros(5,1);
for j = 1:1:5
 h(j) = exp(-norm(xi - cij(:,j))^2/(2*bj^2));
end
gama = 0.10;
for i = 1:1:5
 sys(i) = -1/gama*s*b*h(i);
end
function sys = mdlOutputs(t,x,u)
global cij bj
s = u(1);
x2 = u(3);

xi = x2;

W = [x(1) x(2) x(3) x(4) x(5)]';
h = zeros(5,1);
for j = 1:1:5
 h(j) = exp(-norm(xi - cij(:,j))^2/(2*bj^2));
end
deltap = W'*h;

sys(1) = deltap;

```

**4. S function of plant: chap6\_2plant.m**

```

function [sys,x0,str,ts] = s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
 sys = mdlDerivatives(t,x,u);
case 3,
 sys = mdlOutputs(t,x,u);
case {2, 4, 9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end

```

```

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.2 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
delta=1*x(2)+0.01*sign(x(2));
sys(1)=x(2);
sys(2)=-25*x(2)+133*(ut-delta)+1.0*sin(t);
function sys=mdlOutputs(t,x,u)
delta=1*x(2)+0.01*sign(x(2));
sys(1)=x(1);
sys(2)=x(2);
sys(3)=delta;

```

**5. Plot program: chap6\_2plot.m**

```

close all;

figure(1);
subplot(211);
plot(t,y(:,1),'r',t,y(:,2),'-.k','linewidth',2);
xlabel('time(s)');ylabel('angle tracking');
legend('ideal angle signal','tracking signal');
subplot(212);
plot(t,cos(t),'r',t,y(:,3),'-.k','linewidth',2);
xlabel('time(s)');ylabel('speed tracking');
legend('ideal speed signal','tracking signal');

figure(2);
plot(t,ut(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('Control input,u');

figure(3);
plot(t,y(:,3),'r',t,deltap(:,1),'-.k','linewidth',2);
xlabel('time(s)');ylabel('delta');
legend('true delta','delta estimation');

```

### 6.3 SLIDING MODE CONTROL BASED ON RBF WITH MPL

The minimum parameter learning (MPL) method [4] can decrease the number of online adaptive parameters to only one, which can be used in the RBF neural network to reduce the computational burden and increase real-time performance. Using minimum parameter learning, in this section, a sliding mode control based on RBF is introduced.

#### 6.3.1 System description

Consider a second nonlinear system as

$$\ddot{\theta} = f(\theta, \dot{\theta}) + g(\theta, \dot{\theta})u + d(t), \quad (6.17)$$

where  $f(\theta, \dot{\theta})$  is unknown nonlinear function,  $g(\theta, \dot{\theta})$  is known nonlinear function,  $u \in R$  and  $y = \theta \in R$  are input and output,  $d(t)$  is disturbance,  $|d(t)| \leq D$ .

Denote  $x_1 = \theta$ ,  $x_2 = \dot{\theta}$ , we have

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x_1, x_2) + g(x_1, x_2)u + d(t).\end{aligned}$$

Define the ideal angle as  $x_{1d}$ , and angle tracking error as  $e = x_1 - x_{1d}$ , design the sliding mode function as

$$s = \dot{e} + ce, \quad (6.18)$$

where  $c > 0$ .

Then

$$\dot{s} = \ddot{e} + c\dot{e} = \ddot{x}_1 + c\dot{x}_1 - \ddot{x}_{1d} = f(x_1, x_2) + g(x_1, x_2)u + d - \ddot{x}_{1d} + c\dot{e}. \quad (6.19)$$

In this section, we design the RBF neural network with MPL to approximate  $f(x_1, x_2)$ .

#### 6.3.2 Sliding mode control based on RBF

The algorithm of the RBF network is

$$h_j = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2b_j^2}\right), \quad j = 1, \dots, m$$

$$f(x_1, x_2) = \mathbf{W}^{*T} \mathbf{h}(\mathbf{x}) + \varepsilon$$

where  $\mathbf{x}$  is the input signal of the network,  $j$  is the number of hidden layer nodes in the network,  $\mathbf{h} = [h_1, h_2, \dots, h_n]^T$  is the output of Gaussian function,

$\mathbf{W}^*$  is ideal neural network weight value,  $\varepsilon$  is approximation error of neural network, and  $\varepsilon \leq \varepsilon_N$ .

If we use the RBF network to approximate  $f(x_1, x_2)$ , the network input is selected as  $\mathbf{x} = [x_1 x_2]^T$ , and output of the RBF neural network can be expressed as

$$\hat{f}(\mathbf{x}) = \hat{\mathbf{W}}^T \mathbf{h}(\mathbf{x}). \quad (6.20)$$

Using minimum parameter learning [4], define  $\phi = \|\mathbf{W}\|^2$ , where  $\phi$  is a positive constant, let  $\hat{\phi}$  be an estimation of  $\phi$  and define  $\tilde{\phi} = \hat{\phi} - \phi$ .

Design the controller as

$$u = \frac{1}{g(\theta, \dot{\theta})} \left[ -\frac{1}{2} s \hat{\phi} \mathbf{h}^T \mathbf{h} + \ddot{\theta}_d - c \dot{e} - \eta \text{sgn}(s) - \mu s \right], \quad (6.21)$$

where  $\eta \geq \varepsilon_N + D$ ,  $\mu > 0$ .

Substituting Eq. (6.21) into Eq. (6.19), we then have

$$\dot{s} = \mathbf{W}^T \mathbf{h} + \varepsilon - \frac{1}{2} s \hat{\phi} \mathbf{h}^T \mathbf{h} - \eta \text{sgn}(s) + d - \mu s. \quad (6.22)$$

The Lyapunov function is designed as

$$L = \frac{1}{2} s^2 + \frac{1}{2\gamma} \tilde{\phi}^2,$$

where  $\gamma > 0$ .

Consider Eqs. (6.21) and (6.22); we have

$$\begin{aligned} \dot{L} &= ss + \frac{1}{\gamma} \tilde{\phi} \dot{\phi} = s \left( \mathbf{W}^T \mathbf{h} + \varepsilon - \frac{1}{2} s \hat{\phi} \mathbf{h}^T \mathbf{h} - \eta \text{sgn}(s) + d - \mu s \right) + \frac{1}{\gamma} \tilde{\phi} \dot{\phi} \\ &\leq \frac{1}{2} s^2 \phi \mathbf{h}^T \mathbf{h} + \frac{1}{2} - \frac{1}{2} s^2 \hat{\phi} \mathbf{h}^T \mathbf{h} + (\varepsilon + d)s - \eta + \frac{1}{\gamma} \tilde{\phi} \dot{\phi} - \mu s^2 \\ &= -\frac{1}{2} s^2 \tilde{\phi} \mathbf{h}^T \mathbf{h} + \frac{1}{2} + (\varepsilon + d)s - \eta |s| + \frac{1}{\gamma} \tilde{\phi} \dot{\phi} - \mu s^2 \\ &= \tilde{\phi} \left( -\frac{1}{2} s^2 \mathbf{h}^T \mathbf{h} + \frac{1}{\gamma} \dot{\phi} \right) + \frac{1}{2} + (\varepsilon + d)s - \eta |s| - \mu s^2 \\ &\leq \tilde{\phi} \left( -\frac{1}{2} s^2 \mathbf{h}^T \mathbf{h} + \frac{1}{\gamma} \dot{\phi} \right) + \frac{1}{2} - \mu s^2 \end{aligned}.$$

Design the adaptive law as

$$\dot{\tilde{\phi}} = \frac{\gamma}{2} s^2 \mathbf{h}^T \mathbf{h} - \kappa \gamma \hat{\phi}, \quad (6.23)$$

where  $\kappa > 0$ .

Then

$$\dot{L} \leq -\kappa\tilde{\phi}\hat{\phi} + \frac{1}{2} - \mu s^2 \leq -\frac{\kappa}{2}(\tilde{\phi}^2 - \phi^2) + \frac{1}{2} - \mu s^2 = -\frac{\kappa}{2}\tilde{\phi}^2 - \mu s^2 + \left(\frac{\kappa}{2}\phi^2 + \frac{1}{2}\right).$$

Define  $\kappa = \frac{2\mu}{\gamma}$ , then

$$\dot{L} \leq -\frac{\mu}{\gamma}\tilde{\phi}^2 - \mu s^2 + \left(\frac{\kappa}{2}\phi^2 + \frac{1}{2}\right) = -2\mu\left(\frac{1}{2\gamma}\tilde{\phi}^2 + \frac{1}{2}s^2\right) + \left(\frac{\kappa}{2}\phi^2 + \frac{1}{2}\right) = -2\mu L + Q,$$

where  $Q = \frac{\kappa}{2}\phi^2 + \frac{1}{2}$ .

Use Lemma 1.3; the solution of  $\dot{L} \leq -2\mu L + Q$  is

$$L \leq \frac{Q}{2\mu} + \left(L(0) - \frac{Q}{2\mu}\right)e^{-2\mu t},$$

then

$$\lim_{t \rightarrow \infty} L = \frac{Q}{2\mu} = \frac{\frac{\kappa}{2}\phi^2 + \frac{1}{2}}{2\mu} = \frac{\kappa\phi^2 + 1}{4\mu} = \frac{\frac{2\mu}{\gamma}\phi^2 + 1}{4\mu} = \frac{\phi^2}{2\gamma} + \frac{1}{4\mu}. \quad (6.24)$$

From above, we can see that the convergence precision depends on  $\gamma$  and  $\mu$ .

**Remark 1:**  $s^2\phi\mathbf{h}^T\mathbf{h} + 1 = s^2||\mathbf{W}||^2\mathbf{h}^T\mathbf{h} + 1 = s^2||\mathbf{W}||^2||\mathbf{h}||^2 + 1 = s^2||\mathbf{W}^T\mathbf{h}||^2 + 1 \geq 2s\mathbf{W}^T\mathbf{h}$ , i.e.,  $s\mathbf{W}^T\mathbf{h} \leq \frac{1}{2}s^2\phi\mathbf{h}^T\mathbf{h} + \frac{1}{2}$ ;

**Remark 2:** since  $(\tilde{\phi} + \phi)^2 \geq 0$ , then  $\tilde{\phi}^2 + 2\tilde{\phi}\phi + \phi^2 \geq 0$  and  $\tilde{\phi}^2 + 2\tilde{\phi}(\hat{\phi} - \tilde{\phi}) + \phi^2 \geq 0$ , i.e.,  $2\tilde{\phi}\hat{\phi} \geq \tilde{\phi}^2 - \phi^2$ .

However, the shortcoming of the minimum parameter learning method is that its approximation precision cannot be guaranteed.

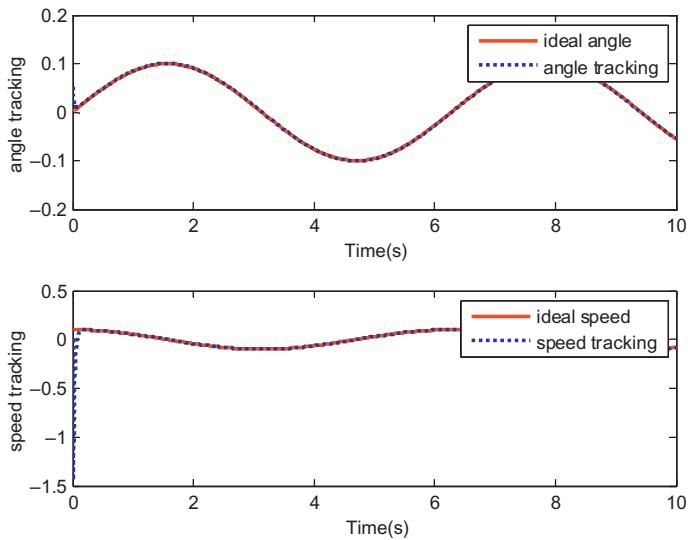
### 6.3.3 Simulation example

Consider an inverted pendulum as

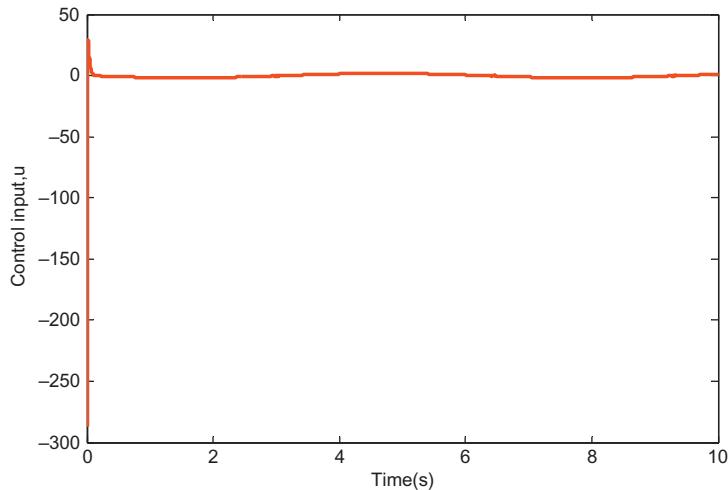
$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{g\sin x_1 - mlx_2^2\cos x_1 \sin x_1 / (m_c + m)}{l(4/3 - m\cos^2 x_1 / (m_c + m))} + \frac{\cos x_1 / (m_c + m)}{l(4/3 - m\cos^2 x_1 / (m_c + m))} u, \end{aligned}$$

where  $f(x_1, x_2) = \frac{g\sin x_1 - mlx_2^2\cos x_1 \sin x_1 / (m_c + m)}{l(4/3 - m\cos^2 x_1 / (m_c + m))}$ ,  $g(x_1, x_2) = \frac{\cos x_1 / (m_c + m)}{l(4/3 - m\cos^2 x_1 / (m_c + m))}$ ,  $x_1$  and  $x_2$  are angle and angle speed, respectively,  $g = 9.8$ ,  $m_c = 1$  is cart mass,  $m = 0.1$  is the mass of pendulum,  $l = 0.5$  is half the length of the pendulum and  $u$  is control input.

Let the ideal angle be  $x_{1d} = 0.1\sin(t)$ . The initial states of the plant is  $[\pi/60, 0]$ , we use control law as Eq. (6.21) and adaptive law as Eq. (6.23), choose  $\gamma = 0.05$ ,  $c = 15$  and  $\eta = 0.1$ . Simulation results are shown in Figs. 6.6 and 6.7.



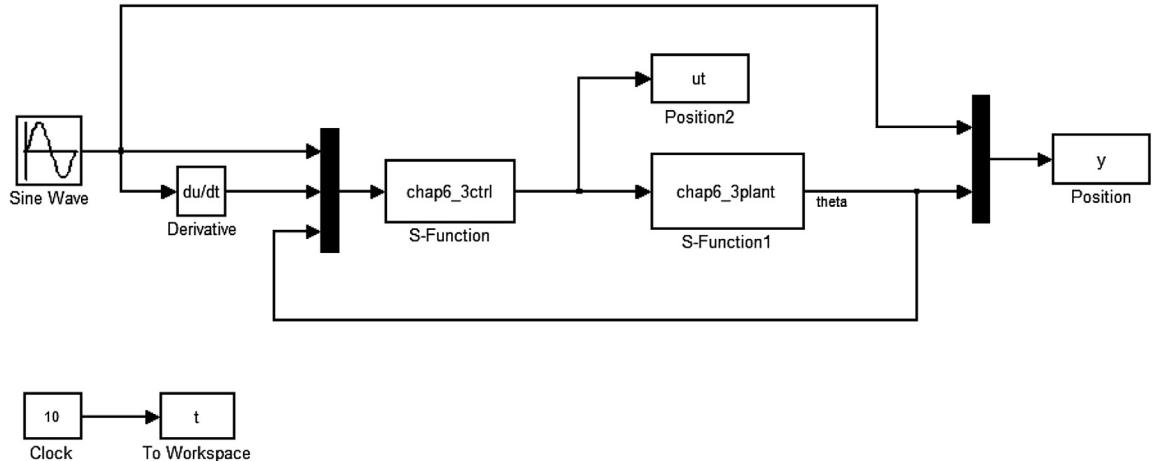
■ FIGURE 6.6 Angle and angle speed tracking.



■ FIGURE 6.7 Control input.

## Simulation programs

## 1. Main program: chap6\_3sim.mdl



## 2. S function of controller: chap6\_3ctrl.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
global cc bb c miu
sizes=simsizes;
sizes.NumContStates = 1;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;

```

```
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = 0;
str = [];
ts = [];
cc = [-2 -1 0 1 2;
 -2 -1 0 1 2];
bb = 1;
c = 200;
miu = 30;
function sys = mdlDerivatives(t,x,u)
global cc bb c miu
x1d = u(1);
dx1d = u(2);
x1 = u(3);
x2 = u(4);

e = x1 - x1d;
de = x2 - dx1d;
s = c*e + de;
xi = [x1;x2];
h = zeros(5,1);
for j = 1:1:5
 h(j) = exp(-norm(xi - cc(:,j))^2/(2*bb*bb));
end
gama = 150;
k = 2*miu/gama;
sys(1) = gama/2*s^2*h'*h - k*gama*x;

function sys = mdlOutputs(t,x,u)
global cc bb c miu
x1d = u(1);
dx1d = u(2);
x1 = u(3);
x2 = u(4);

e = x1 - x1d;
de = x2 - dx1d;
thd = 0.1*sin(t);
dthd = 0.1*cos(t);
ddthd = -0.1*sin(t);
s = c*e + de;
```

```

fi = x;
xi = [x1;x2];
h=zeros(5,1);
for j=1:1:5
 h(j)=exp(-norm(xi-cc(:,j))^2/(2*bb*bb));
end

g=9.8;mc=1.0;m=0.1;l=0.5;
S=l*(4/3-m*(cos(x1))^2/(mc+m));
gx=cos(x1)/(mc+m);
gx=gx/S;

xite=0.5;
miu=40;

ut=1/gx*(-0.5*s*fi*h'*h+ddthd-c*de-xite*sign(s)-
miu*s);
sys(1)=ut;

```

### 3. S function of plant: chap6\_3plant.m

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=2;
sizes.NumDiscStates=0;
sizes.NumOutputs=2;
sizes.NumInputs=1;
sizes.DirFeedthrough=0;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[pi/60 0];

```

```

str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
g=9.8;mc=1.0;m=0.1;l=0.5;
S=l*(4/3-m*(cos(x(1)))^2/(mc+m));
fx=g*sin(x(1))-m*l*x(2)^2*cos(x(1))*sin(x(1))/(mc+m);
fx=fx/S;
gx=cos(x(1))/(mc+m);
gx=gx/S;
%%%%%%%%%%%%%
dt=0.1*10*sin(t);
%%%%%%%%%%%%%

sys(1)=x(2);
sys(2)=fx+gx*u+dt;
function sys=mdlOutputs(t,x,u)
g=9.8;
mc=1.0;
m=0.1;
l=0.5;

S=l*(4/3-m*(cos(x(1)))^2/(mc+m));
fx=g*sin(x(1))-m*l*x(2)^2*cos(x(1))*sin(x(1))/(mc+m);
fx=fx/S;

sys(1)=x(1);
sys(2)=x(2);

```

**4. Plot program: chap6\_3plot.m**

```

close all;

figure(1);
subplot(211);
plot(t,y(:,1),'r',t,y(:,2),'b','linewidth',2);
xlabel('time(s)');ylabel('angle tracking');
legend('ideal angle','angle tracking');
subplot(212);
plot(t,0.1*cos(t),'r',t,y(:,3),'b','linewidth',2);
xlabel('time(s)');ylabel('speed tracking');
legend('ideal speed','speed tracking');

figure(2);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input,u');

```

## 6.4 SLIDING MODE CONTROL BASED ON RBF WITH MPL FOR MANIPULATORS

Based on [Section 6.3](#), in this section, we discuss about sliding mode controller design based on RBF with MPL for manipulators.

### 6.4.1 System description

Consider the dynamic equation of  $n$  link manipulators as

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\dot{\mathbf{q}}) + \boldsymbol{\tau}_d = \boldsymbol{\tau}, \quad (6.25)$$

where  $\mathbf{D}(\mathbf{q})$  is the inertia matrix,  $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$  is the coriolis/centripetal matrix,  $\mathbf{G}(\mathbf{q})$  is the gravity vector,  $\mathbf{F}(\dot{\mathbf{q}})$  is friction force,  $\boldsymbol{\tau}_d$  denotes unknown disturbance and  $\boldsymbol{\tau}$  is the control input.

Tracking errors are defined as

$$\mathbf{e}(t) = \mathbf{q}_d(t) - \mathbf{q}(t).$$

The control goals are  $\mathbf{e}(t) \rightarrow 0$  and  $\dot{\mathbf{e}}(t) \rightarrow 0$  as  $t \rightarrow \infty$ .

### 6.4.2 RBF neural network design

Define the sliding mode function as

$$\mathbf{r} = \dot{\mathbf{e}} + \Lambda \mathbf{e}, \quad (6.26)$$

where  $\Lambda = \Lambda^T > 0$ , then

$$\dot{\mathbf{q}} = -\mathbf{r} + \dot{\mathbf{q}}_d + \Lambda \mathbf{e}$$

$$\begin{aligned} \mathbf{D}\dot{\mathbf{r}} &= \mathbf{D}(\ddot{\mathbf{q}}_d - \ddot{\mathbf{q}} + \Lambda \ddot{\mathbf{e}}) = \mathbf{D}(\ddot{\mathbf{q}}_d + \Lambda \ddot{\mathbf{e}}) - \mathbf{D}\ddot{\mathbf{q}} \\ &= \mathbf{D}(\ddot{\mathbf{q}}_d + \Lambda \ddot{\mathbf{e}}) + \mathbf{C}\dot{\mathbf{q}} + \mathbf{G} + \mathbf{F} + \boldsymbol{\tau}_d - \boldsymbol{\tau} \\ &= \mathbf{D}(\ddot{\mathbf{q}}_d + \Lambda \ddot{\mathbf{e}}) - \mathbf{C}\mathbf{r} + \mathbf{C}(\dot{\mathbf{q}}_d + \Lambda \mathbf{e}) + \mathbf{G} + \mathbf{F} + \boldsymbol{\tau}_d - \boldsymbol{\tau}, \\ &= -\mathbf{C}\mathbf{r} - \boldsymbol{\tau} + \mathbf{f} + \boldsymbol{\tau}_d \end{aligned} \quad (6.27)$$

where  $\mathbf{f} = \mathbf{D}(\ddot{\mathbf{q}}_d + \Lambda \ddot{\mathbf{e}}) + \mathbf{C}(\dot{\mathbf{q}}_d + \Lambda \mathbf{e}) + \mathbf{G} + \mathbf{F}$ .

In practical engineering, modeling information in [Eq. \(6.25\)](#) is often unknown, in this section, we use the RBF neural network with MPL to approximate  $\mathbf{f}$  function [5], then we can design the controller without modelling.

For the  $i$ th joint, the algorithm of the RBF network is

$$h_{ij} = \exp(||\mathbf{x}_i - \mathbf{c}_{ij}||^2 / \sigma_{ij}^2), \quad j = 1, 2, \dots, m$$

$$f_i = \mathbf{w}_i^T \mathbf{h}_i + \varepsilon_i, \quad (6.28)$$

where  $\mathbf{x}_i = [\mathbf{e}_i \quad \dot{\mathbf{e}}_i \quad \mathbf{q}_{di} \quad \dot{\mathbf{q}}_{di} \quad \ddot{\mathbf{q}}_{di}]$  are input of RBF,  $\mathbf{h}_i = [h_{i1} h_{i2} \cdots h_{im}]^T$ ,  $\varepsilon_i$  is approximation error and  $\mathbf{w}_i$  is ideal weight value.

According to the  $f_i$  expression, the inputs of RBF are chosen as

$$\mathbf{X} = [\mathbf{e} \quad \dot{\mathbf{e}} \quad \mathbf{q}_d \quad \dot{\mathbf{q}}_d \quad \ddot{\mathbf{q}}_d],$$

then

$$\mathbf{f} = [f_1 \cdots f_i \cdots f_n]^T = \begin{bmatrix} \mathbf{w}_1^T \mathbf{h}_1 + \varepsilon_1 \\ \vdots \\ \mathbf{w}_i^T \mathbf{h}_i + \varepsilon_i \\ \vdots \\ \mathbf{w}_n^T \mathbf{h}_n + \varepsilon_n \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^T \mathbf{h}_1 \\ \vdots \\ \mathbf{w}_i^T \mathbf{h}_i \\ \vdots \\ \mathbf{w}_n^T \mathbf{h}_n \end{bmatrix} + \boldsymbol{\varepsilon},$$

where  $\boldsymbol{\varepsilon} = [\varepsilon_1 \cdots \varepsilon_i \cdots \varepsilon_n]^T$ ,  $\|\boldsymbol{\varepsilon}\| \leq \varepsilon_N$ .

#### 6.4.3 Controller design and analysis based on MPL

Define  $\hat{\mathbf{w}}_i$  as an estimation of  $\mathbf{w}_i$ , then define

$$\tilde{\mathbf{w}}_i = \mathbf{w}_i - \hat{\mathbf{w}}_i, \quad \|\mathbf{w}_i\|_F \leq w_{imax}.$$

Define the minimum parameter as  $\phi = \max_{1 \leq i \leq n} \{\|\mathbf{w}_i\|^2\}$  [4], where  $\phi$  is a positive constant, and  $\hat{\phi}$  is an estimation of  $\phi$ ,  $\tilde{\phi} = \hat{\phi} - \phi$ .

Define  $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{bmatrix}$ ,  $\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_n \end{bmatrix}$ ,  $\tilde{\mathbf{W}} = \mathbf{W} - \hat{\mathbf{W}}$ , according to the GL operator [6],

we have  $\mathbf{W} \circ \mathbf{H} = \begin{bmatrix} \mathbf{w}_1^T \mathbf{h}_1 \\ \vdots \\ \mathbf{w}_n^T \mathbf{h}_n \end{bmatrix}$ ,  $\mathbf{r} \circ \mathbf{r} = \begin{bmatrix} \mathbf{r}_1^T \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_n^T \mathbf{r}_n \end{bmatrix}$  and  $\mathbf{H} \circ \mathbf{H} = \begin{bmatrix} \mathbf{h}_1^T \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_n^T \mathbf{h}_n \end{bmatrix}$ , then  $\mathbf{f}$  can

be expressed as

$$\mathbf{f} = \mathbf{W} \circ \mathbf{H} + \boldsymbol{\varepsilon}.$$

Design the control law as

$$\tau = \frac{1}{2} \hat{\phi} \mathbf{r} \circ (\mathbf{H} \circ \mathbf{H}) + \mathbf{K}_v \mathbf{r} - \nu, \quad (6.29)$$

where  $\nu$  is the robust term to overcome approximation error  $\boldsymbol{\varepsilon}$ .

The robust term  $\nu$  is designed as

$$\nu = -(\varepsilon_N + b_d) \text{sgn}(\mathbf{r}), \quad (6.30)$$

where  $\|\tau_d\| \leq b_d$ .

Substituting Eq. (6.29) into Eq. (6.27), we have

$$\dot{\mathbf{D}}\mathbf{r} = -(\mathbf{K}_v + \mathbf{C})\mathbf{r} - \frac{1}{2}\hat{\phi}\mathbf{r} \circ (\mathbf{H} \circ \mathbf{H}) + (\mathbf{f} + \boldsymbol{\tau}_d) + \mathbf{v}. \quad (6.31)$$

Define the Lyapunov function as

$$L = \frac{1}{2}\mathbf{r}^T \mathbf{D}\mathbf{r} + \frac{1}{2\gamma}\tilde{\phi}^2,$$

where  $\gamma > 0$ .

Then

$$\begin{aligned} \dot{L} &= \mathbf{r}^T \mathbf{D}\mathbf{r} + \frac{1}{2}\mathbf{r}^T \dot{\mathbf{D}}\mathbf{r} + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} \\ &= \mathbf{r}^T \left[ -(\mathbf{K}_v + \mathbf{C})\mathbf{r} - \frac{1}{2}\hat{\phi}\mathbf{r} \circ (\mathbf{H} \circ \mathbf{H}) + (\mathbf{f} + \boldsymbol{\tau}_d) + \mathbf{v} \right] + \frac{1}{2}\mathbf{r}^T \dot{\mathbf{D}}\mathbf{r} + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} \\ &= \mathbf{r}^T \left[ -\mathbf{K}_v\mathbf{r} - \frac{1}{2}\hat{\phi}\mathbf{r} \circ (\mathbf{H} \circ \mathbf{H}) + \mathbf{W} \circ \mathbf{H} + (\boldsymbol{\varepsilon} + \boldsymbol{\tau}_d + \mathbf{v}) \right] - \frac{1}{2}\mathbf{r}^T (\dot{\mathbf{D}} - 2\mathbf{C})\mathbf{r} + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} \\ &= \mathbf{r}^T \left[ -\frac{1}{2}\hat{\phi}\mathbf{r} \circ (\mathbf{H} \circ \mathbf{H}) + \mathbf{W} \circ \mathbf{H} \right] - \mathbf{r}^T \mathbf{K}_v\mathbf{r} + \mathbf{r}^T (\boldsymbol{\varepsilon} + \boldsymbol{\tau}_d + \mathbf{v}) + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} \end{aligned}$$

Since

$$\mathbf{r}^T (\dot{\mathbf{D}} - 2\mathbf{C})\mathbf{r} = 0$$

$$\mathbf{r}^T (\boldsymbol{\varepsilon} + \boldsymbol{\tau}_d + \mathbf{v}) = \mathbf{r}^T (\boldsymbol{\varepsilon} + \boldsymbol{\tau}_d - (\boldsymbol{\varepsilon}_N + b_d)\text{sgn}(\mathbf{r})) \leq 0$$

$$\mathbf{r}^T [\mathbf{W} \circ \mathbf{H}] = \begin{bmatrix} r_1 & \cdots & r_n \end{bmatrix} \begin{bmatrix} \mathbf{w}_1^T \mathbf{h}_1 \\ \vdots \\ \mathbf{w}_n^T \mathbf{h}_n \end{bmatrix} = r_1 \mathbf{w}_1^T \mathbf{h}_1 + \cdots + r_n \mathbf{w}_n^T \mathbf{h}_n = \sum_{i=1}^n r_i \mathbf{w}_i^T \mathbf{h}_i$$

$$r_i^2 \phi \mathbf{h}_i^T \mathbf{h}_i + 1 \geq r_i^2 ||\mathbf{w}_i||^2 ||\mathbf{h}_i||^2 + 1 \geq 2r_i \mathbf{w}_i^T \mathbf{h}_i, \quad \text{which is}$$

$$r_i \mathbf{w}_i^T \mathbf{h}_i \leq \frac{1}{2} r_i^2 \phi \mathbf{h}_i^T \mathbf{h}_i + \frac{1}{2} = \frac{1}{2} r_i^2 \phi \mathbf{h}_i^T \mathbf{h}_i + \frac{1}{2}$$

$$\mathbf{r}^T [\mathbf{W} \circ \mathbf{H}] \leq \frac{1}{2} \phi \sum_{i=1}^n r_i^2 \mathbf{h}_i^T \mathbf{h}_i + \frac{n}{2}$$

$$\begin{aligned} \mathbf{r}^T \left[ -\frac{1}{2}\hat{\phi}\mathbf{r} \circ (\mathbf{H} \circ \mathbf{H}) \right] &= -\frac{1}{2}\hat{\phi} \begin{bmatrix} r_1 & \cdots & r_n \end{bmatrix} \left( \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix} \circ \begin{bmatrix} \mathbf{h}_1^T \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_n^T \mathbf{h}_n \end{bmatrix} \right) \\ &= -\frac{1}{2}\hat{\phi} \begin{bmatrix} r_1 & \cdots & r_n \end{bmatrix} \begin{bmatrix} r_1 \mathbf{h}_1^T \mathbf{h}_1 \\ \vdots \\ r_n \mathbf{h}_n^T \mathbf{h}_n \end{bmatrix}, \\ &= -\frac{1}{2}\hat{\phi}(r_1^2 ||\mathbf{h}_1||^2 + \cdots + r_n^2 ||\mathbf{h}_n||^2) = -\frac{1}{2}\hat{\phi} \sum_{i=1}^n r_i^2 ||\mathbf{h}_i||^2 \end{aligned}$$

where  $n$  denotes number of joints, for second link manipulators,  $n = 2$ .

Then we have

$$\begin{aligned}\dot{L} &\leq -\frac{1}{2}\hat{\phi}\sum_{i=1}^n r_i^2 \|\mathbf{h}_i\|^2 + \frac{1}{2}\phi\sum_{i=1}^n r_i^2 \mathbf{h}_i^T \mathbf{h}_i + \frac{n}{2} + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} - \mathbf{r}^T \mathbf{K}_v \mathbf{r} \\ &= -\frac{1}{2}\tilde{\phi}\sum_{i=1}^n r_i^2 \|\mathbf{h}_i\|^2 + \frac{n}{2} + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} - \mathbf{r}^T \mathbf{K}_v \mathbf{r} \\ &= \tilde{\phi}\left(-\frac{1}{2}\sum_{i=1}^n r_i^2 \|\mathbf{h}_i\|^2 + \frac{1}{\gamma}\dot{\phi}\right) + \frac{n}{2} - \mathbf{r}^T \mathbf{K}_v \mathbf{r}\end{aligned}.$$

Design the adaptive law as

$$\dot{\tilde{\phi}} = \frac{\gamma}{2} \sum_{i=1}^n r_i^2 \|\mathbf{h}_i\|^2. \quad (6.32)$$

Then

$$\dot{L} \leq \frac{n}{2} - \mathbf{r}^T \mathbf{K}_v \mathbf{r}.$$

To guarantee  $\dot{L} \leq 0$ , we must ensure  $\frac{n}{2} \leq \mathbf{r}^T \mathbf{K}_v \mathbf{r}$ , then we have

$$\|\mathbf{r}\| \leq \sqrt{\frac{n}{2K_v}}. \quad (6.33)$$

#### 6.4.4 Simulation example

Consider dynamic equation of two link manipulators as

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\dot{\mathbf{q}}) + \boldsymbol{\tau}_d = \boldsymbol{\tau},$$

where

$$\mathbf{D}(\mathbf{q}) = \begin{bmatrix} p_1 + p_2 + 2p_3 \cos q_2 & p_2 + p_3 \cos q_2 \\ p_2 + p_3 \cos q_2 & p_2 \end{bmatrix},$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -p_3 \dot{q}_2 \sin q_2 & -p_3 (\dot{q}_1 + \dot{q}_2) \sin q_2 \\ p_3 \dot{q}_1 \sin q_2 & 0 \end{bmatrix}$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} p_4 g \cos q_1 + p_5 g \cos(q_1 + q_2) \\ p_5 g \cos(q_1 + q_2) \end{bmatrix},$$

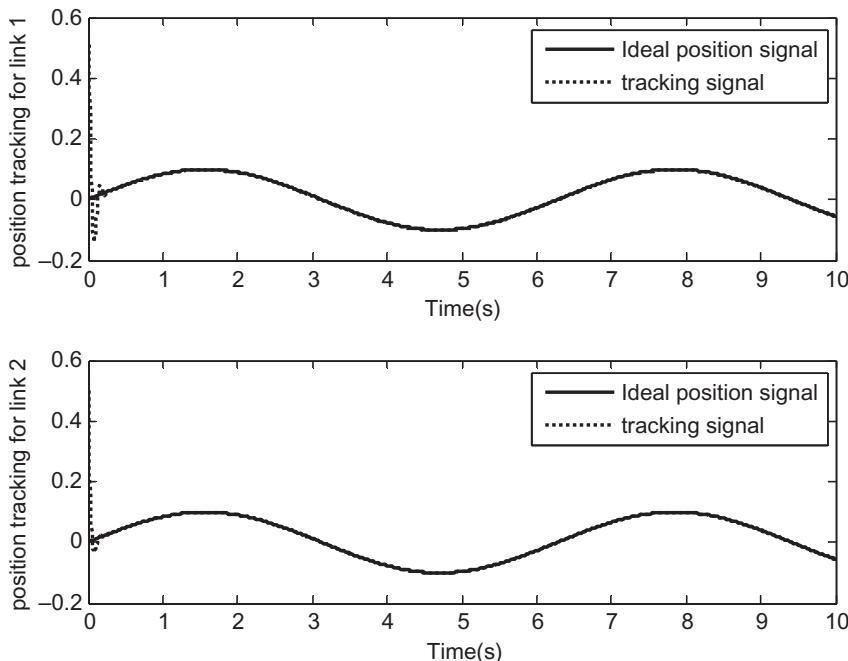
$$\mathbf{F}(\dot{\mathbf{q}}) = 0.2 \operatorname{sgn}(\dot{\mathbf{q}}), \quad \boldsymbol{\tau}_d = [0.2 \sin(t) \quad 0.2 \sin(t)]^T$$

The initial states of the plant are chosen as  $[0.50, 0, 0.50, 0]$ ,  $\mathbf{p} = [p_1, p_2, p_3, p_4, p_5] = [2.9, 0.76, 0.87, 3.04, 0.87]$ . The ideal angle signals of two joints are set as  $q_{1d} = 0.1\sin t$  and  $q_{2d} = 0.1\sin t$ . In RBF neural network design, according to  $f(\mathbf{x})$  expression, the inputs are chosen as  $\mathbf{z} = [e \quad \dot{e} \quad q_d \quad \dot{q}_d \quad \ddot{q}_d]$ , and according to the inputs range,

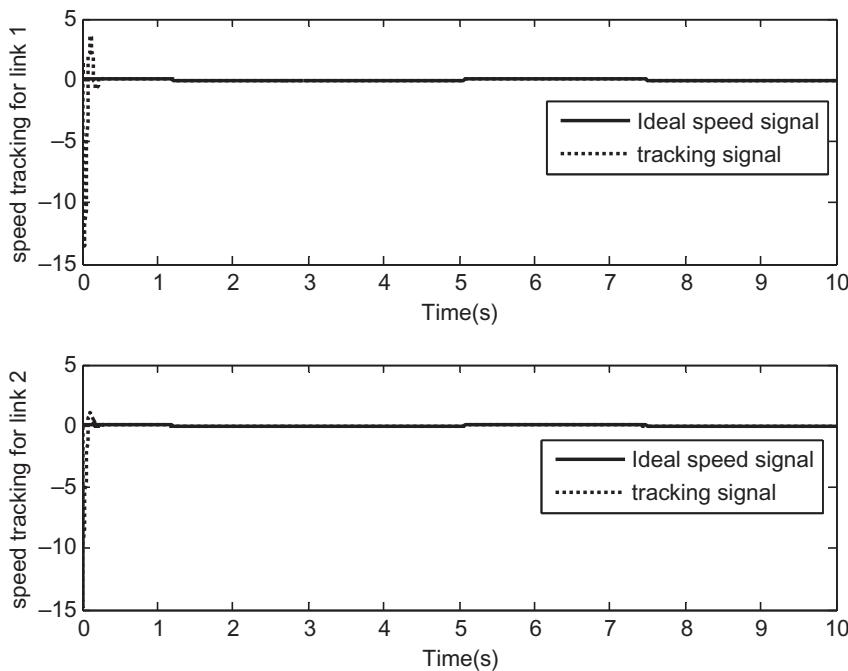
$$\text{we choose } c = 0.1 \times \begin{bmatrix} -1 & 0.5 & 0 & 0.5 & 1 \\ -1 & 0.5 & 0 & 0.5 & 1 \\ -1 & 0.5 & 0 & 0.5 & 1 \\ -1 & 0.5 & 0 & 0.5 & 1 \\ -1 & 0.5 & 0 & 0.5 & 1 \end{bmatrix}, \quad b_i = 0.50, \quad i = 1, \dots, 5,$$

and initial weight values are set as zero.

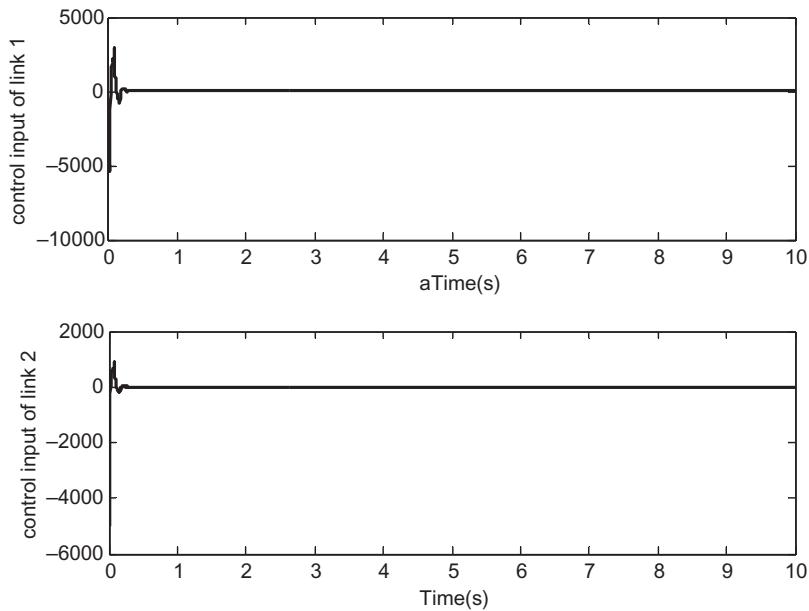
We adapt control law (6.29) and adaptive law (6.32), choose  $\mathbf{K}_v = \text{diag}\{200, 200\}$ ,  $\mathbf{\Lambda} = \text{diag}\{50, 50\}$ ,  $\varepsilon_N = 0.50$ ,  $b_d = 0.20$  and  $\gamma = 500$ , the saturation function is used instead of the switch function, and set  $\Delta = 0.01$ . The simulation results are shown in Figs. 6.8–6.10.



■ FIGURE 6.8 Angle tracking of first link and second link.



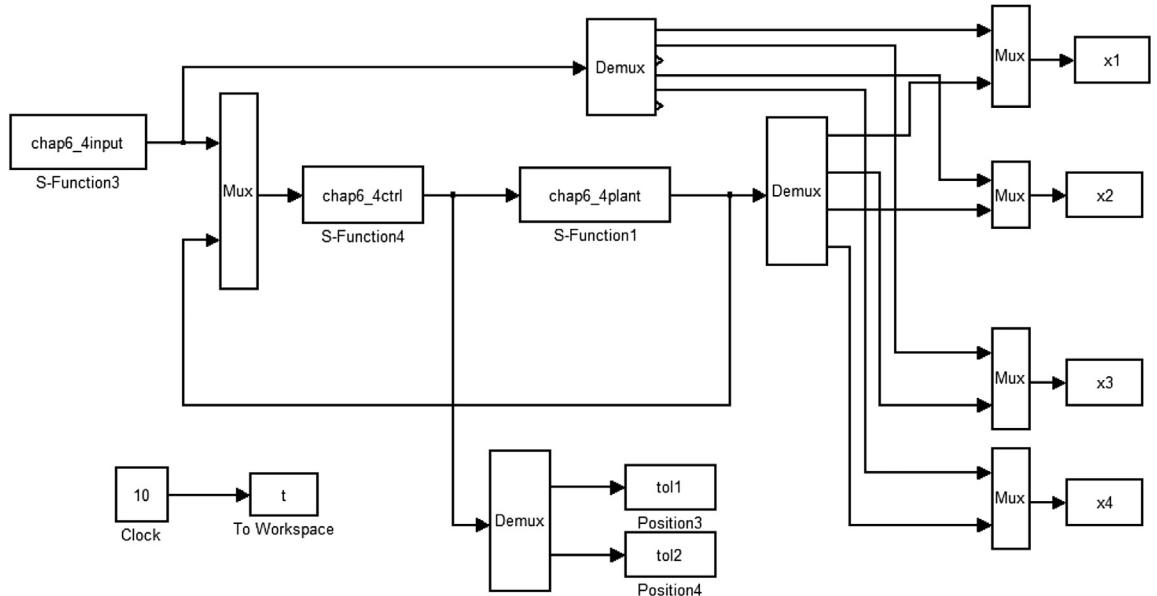
■ FIGURE 6.9 Angle speed tracking of first link and second link.



■ FIGURE 6.10 Control input of first link and second link.

## Simulation Programs:

## 1. Main simulink program: chap6\_4sim.mdl



## 2. S function of input: chap6\_4input.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 6;
sizes.NumInputs = 0;
sizes.DirFeedthrough = 0;

```

```

sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys = mdlOutputs(t,x,u)
qd1=0.1*sin(t);
d_qd1=0.1*cos(t);
dd_qd1=-0.1*sin(t);
qd2=0.1*sin(t);
d_qd2=0.1*cos(t);
dd_qd2=-0.1*sin(t);

sys(1)=qd1;
sys(2)=d_qd1;
sys(3)=dd_qd1;
sys(4)=qd2;
sys(5)=d_qd2;
sys(6)=dd_qd2;

```

**3. S function of controller: chap6\_4ctrl.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
global node c b Fai
node=5;
c=0.1*[-1 -0.5 0 0.5 1;
 -1 -0.5 0 0.5 1;
 -1 -0.5 0 0.5 1;
 -1 -0.5 0 0.5 1];
b=0.50;
Fai=50*eye(2);

```

```
sizes = simsizes;
sizes.NumContStates = 1;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 10;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0];
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
global node c b Fai
qd1=u(1);
d_qd1=u(2);
dd_qd1=u(3);
qd2=u(4);
d_qd2=u(5);
dd_qd2=u(6);

q1=u(7);
d_q1=u(8);
q2=u(9);
d_q2=u(10);

e1=qd1-q1;
e2=qd2-q2;
de1=d_qd1-d_q1;
de2=d_qd2-d_q2;
e=[e1;e2];
de=[de1;de2];
r=de+Fai*e;

qd=[qd1;qd2];
dqd=[d_qd1;d_qd2];
ddqd=[dd_qd1;dd_qd2];

z1=[e(1);de(1);qd(1);dqd(1);ddqd(1)];
z2=[e(2);de(2);qd(2);dqd(2);ddqd(2)];

h1=zeros(7,1);
h2=zeros(7,1);
for j=1:1:node
 h1(j)=exp(-norm(z1-c(:,j))^2/(b*b));
 h2(j)=exp(-norm(z2-c(:,j))^2/(b*b));
end
```

```

gama=500;
sumi=(r(1))^2*norm(h1)^2+(r(2))^2*norm(h2)^2;
sys(1)=gama/2*sumi;
function sys=mdlOutputs(t,x,u)
global node c b Fai
qd1=u(1);
d_qd1=u(2);
dd_qd1=u(3);
qd2=u(4);
d_qd2=u(5);
dd_qd2=u(6);

q1=u(7);
d_q1=u(8);
q2=u(9);
d_q2=u(10);

q=[q1;q2];
e1=qd1-q1;
e2=qd2-q2;
de1=d_qd1-d_q1;
de2=d_qd2-d_q2;
e=[e1;e2];
de=[de1;de2];
r=de+Fai*e;

qd=[qd1;qd2];
dqd=[d_qd1;d_qd2];
ddqd=[dd_qd1;dd_qd2];
faip=x(1);

z1=[e(1);de(1);qd(1);dqd(1);ddqd(1)];
z2=[e(2);de(2);qd(2);dqd(2);ddqd(2)];
h1=zeros(7,1);
h2=zeros(7,1);
for j=1:1:node
 h1(j)=exp(-norm(z1-c(:,j))^2/(b*b));
 h2(j)=exp(-norm(z2-c(:,j))^2/(b*b));
end

p=[2.9 0.76 0.87 3.04 0.87];
D=[p(1)+p(2)+2*p(3)*cos(q2) p(2)+p(3)*cos(q2);
 p(2)+p(3)*cos(q2) p(2)];
H=[h1;h2];

```

```

Kv = 200*eye(2);
epN = 0.50;
bd = 0.2;

M = 2;
if M == 1
 sat = sign(r);
elseif M == 2 %Saturated function
 faio = 0.01;
 if norm(r) <= faio
 sat = r/faio;
 else
 sat = sign(r);
 end
end
v = -(epN + bd)*sat;

```

```

Kw = Kv*r;
Dr = D*r;
%tol = (r.*faio)/2*(H.*H) + Kv*r - v;
tol(1) = faip/2*r(1)*norm(h1)^2 + Kw(1) - v(1);
tol(2) = faip/2*r(2)*norm(h2)^2 + Kw(2) - v(2);
sys(1) = tol(1);
sys(2) = tol(2);

```

**4. S function of plant: chap6\_4plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
global pg
sizes=simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;

```

```

sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.50 0 0.50 0];
str=[];
ts=[];
p=[2.9 0.76 0.87 3.04 0.87];
g=9.8;
function sys=mdlDerivatives(t,x,u)
global p g
D=[p(1)+p(2)+2*p(3)*cos(x(3)) p(2)+p(3)*cos(x(3));
 p(2)+p(3)*cos(x(3)) p(2)];
C=[-p(3)*x(4)*sin(x(3))-p(3)*(x(2)+x(4))*sin(x(3));
 p(3)*x(2)*sin(x(3)) 0];
G=[p(4)*g*cos(x(1))+p(5)*g*cos(x(1)+x(3));
 p(5)*g*cos(x(1)+x(3))];
dq=[x(2);x(4)];
F=0.2*sign(dq);
told=[0.2*sin(t);0.2*sin(t)];
tol=u(1:2);
S=inv(D)*(tol-C*dq-G-F-told);

```

```

sys(1)=x(2);
sys(2)=S(1);
sys(3)=x(4);
sys(4)=S(2);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);

```

**5. Plot program: chap6\_4plot.m**

```

close all;

figure(1);
subplot(211);
plot(t,x1(:,1),'k',t,x1(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('position tracking for link 1');
legend('Ideal position signal','tracking signal');
subplot(212);
plot(t,x2(:,1),'k',t,x2(:,2),'k:','linewidth',2);

```

```

xlabel('time(s)');ylabel('position tracking for link 2');
legend('Ideal position signal','tracking signal');

figure(2);
subplot(211);
plot(t,x3(:,1),'k',t,x3(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('speed tracking for link 1');
legend('Ideal speed signal','tracking signal');
subplot(212);
plot(t,x4(:,1),'k',t,x4(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('speed tracking for link 2');
legend('Ideal speed signal','tracking signal');

figure(3);
subplot(211);
plot(t,tol1(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('control input of link 1');

subplot(212);
plot(t,tol2(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('control input of link 2');

```

## REFERENCES

- [1] J. Park, I.W. Sandberg, Universal approximation using radial basis function networks, *Neural Comput.* 3 (2) (1991) 246–257.
- [2] J. Liu, X. Wang, *Advanced Sliding Mode Control for Mechanical Systems\_Design, Analysis and Matlab Simulation*, Tsinghua & Springer Press, Beijing, 2011.
- [3] J. Liu, *RBF Neural Network Control for Mechanical Systems\_Design, Analysis and Matlab Simulation*, Tsinghua & Springer Press, Beijing, 2013.
- [4] B. Chen, X. Liu, K. Liu, C. Lin, Direct adaptive fuzzy control of nonlinear strict-feedback systems, *Automatica* 45 (2009) 1530–1535.
- [5] F.L. Lewis, K. Liu, A. Yesildirek, Neural net robot controller with guaranteed tracking performance, *IEEE Trans. Neural Netw.* 6 (3) (1995) 703–715.
- [6] S.S. Ge, C.C. Hang, T.H. Lee, T. Zhang, *Stable Adaptive Neural Network Control*, Kluwer, Boston, MA, 2001.

## FURTHER READING

Hongjun Yang, Jinkun Liu, Minimum parameter learning method for an N-link manipulator with nonlinear disturbance observer, *Int. J. Robot. Autom.* (2015). Available from: <http://dx.doi.org/10.2316/Journal.206.2016.3.206-4442>.

# Sliding mode control based on a fuzzy system

Past research of universal approximation theorem [1] shown that any nonlinear function over a compact set with arbitrary accuracy can be approximated by a fuzzy system. There have been significant research efforts on adaptive fuzzy control for nonlinear systems [2].

In the book [3], several kinds of fuzzy sliding mode controller design methods have been introduced, refer to this book, in this chapter, two sections are introduced as follows:

In Section 7.1, sliding mode control based on a fuzzy system approximation is introduced, unknown function  $f(x)$  is approximated by fuzzy system, and robustness can be guaranteed by switch term.

In Section 7.2, sliding mode control based on a fuzzy system with minimum parameter learning method is introduced, minimum parameter learning can be used in fuzzy system to reduce the computational burden and increase real-time performance [4]. Using the minimum parameter learning, in this section, a sliding mode controller based on a fuzzy system is designed.

## 7.1 SLIDING MODE CONTROL BASED ON A FUZZY SYSTEM APPROXIMATION

### 7.1.1 Problem statement

Consider a second-order system as follows:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + u + d(t),\end{aligned}\tag{7.1}$$

where  $x_1$  and  $x_2$  are respectively angle and angle speed,  $u$  is the control input,  $d(t)$  is disturbance,  $|d(t)| \leq D$ .

Let the desired output be  $x_{1d}$ , and denote error as

$$e = x_{1d} - x_1.$$

Design a sliding mode function as

$$s = \dot{e} + ce, \quad (7.2)$$

where  $c > 0$ , then

$$\dot{s} = \ddot{e} + c\dot{e} = \ddot{x}_{1d} - \ddot{x}_1 + c\dot{e} = \ddot{x}_{1d} - f - u + c\dot{e} - d(t). \quad (7.3)$$

If  $f$  is known, we can design a control law as

$$u = -f + \ddot{x}_{1d} + c\dot{e} + \eta \operatorname{sgn}(s). \quad (7.4)$$

Then Eq. (7.3) becomes  $\dot{s} = -\eta \operatorname{sgn}(s)$ ; therefore, if we design  $\eta \geq |\varepsilon|_{\max}$ , we have

$$s\dot{s} = -\eta|s| - sd(t) \leq 0$$

If  $f(x)$  is unknown, we should approximate  $f(x)$  by some algorithms. In this section, fuzzy system is used to approximate unknown function  $f(x)$ .

### 7.1.2 Controller design based on a fuzzy system

#### 7.1.2.1 Uncertainty approximation using a fuzzy system

If  $f(x)$  is unknown, we can replace  $f(x)$  with the fuzzy system  $\hat{f}(x)$  to realize feedback control. Using the universal approximation theorem, three steps are designed as follows:

1. For  $x_1$  and  $x_2$ , define five fuzzy sets for  $A_1^{l_i}$  and  $A_2^{l_i}$  respectively,  $l_i = 1, 2, \dots, 5$ ;
2. Design  $\prod_{i=1}^n p_i = p_1 \times p_2 = 25$  fuzzy rules to construct fuzzy system  $\hat{f}(x|\theta_f)$ :

$$\begin{aligned} R^{(1)}: & \text{ If } x_1 \text{ is } A_1^1 \text{ and...and } x_2 \text{ is } A_2^1 \text{ then } \hat{f} \text{ is } B^1, \\ R^{(25)}: & \text{ If } x_1 \text{ is } A_1^5 \text{ and...and } x_2 \text{ is } A_2^5 \text{ then } \hat{f} \text{ is } B^{25}, \end{aligned} \quad (7.5)$$

where  $l_i = 1, 2, \dots, 5$  and  $i = 1, 2, p_1 = p_2 = 5$ .

3. Using fuzzy inference, the output of fuzzy system is

$$\hat{f}(x|\theta_f) = \frac{\sum_{l_1=1}^5 \sum_{l_2=1}^5 \bar{y}_f^{l_1 l_2} \left( \prod_{i=1}^2 \mu_{A_i^{l_i}}(x_i) \right)}{\sum_{l_1=1}^5 \sum_{l_2=1}^5 \left( \prod_{i=1}^2 \mu_{A_i^{l_i}}(x_i) \right)}, \quad (7.6)$$

where  $\mu_{A_i^{l_i}}(x_i)$  is the membership function of  $x_i$ .

Let  $\bar{y}_f^{l_1 l_2}$  be a free parameter and be put in the set  $\hat{\theta}_f \in R^{(25)}$ . Column vector  $\xi(x)$  is introduced and Eq. (7.6) can be written as:

$$\hat{f}(x|\theta_f) = \hat{\theta}_f^T \xi(x), \quad (7.7)$$

where  $x = [x_1 \ x_2]^T$ ,  $\xi(x)$  is  $\prod_{i=1}^n p_i = p_1 \times p_2 = 25$  dimensional column vector, and  $l_1, l_2$  elements are, respectively,

$$\xi_{l_1 l_2}(x) = \frac{\prod_{i=1}^2 \mu_{A_i^{l_i}}(x_i)}{\sum_{l_1=1}^{p_1} \sum_{l_2=1}^{p_2} \left( \prod_{i=1}^2 \mu_{A_i^{l_i}}(x_i) \right)}. \quad (7.8)$$

The membership functions are needed to be selected according to experiences. Moreover, all the states must be known.

### 7.1.2.2 Design of adaptive fuzzy sliding mode controller

Suppose the optimal parameter as

$$\theta_f^* = \arg \min_{\theta_f \in \Omega_f} \left[ \sup_{x \in R^n} |\hat{f}(x| \theta_f) - f(x)| \right]$$

where  $\Omega_f$  is the set of  $\theta_f$ , i.e.,  $\theta_f \in \Omega_f$ .

The term  $f$  can be expressed as

$$f = \theta_f^{*T} \xi(x) + \varepsilon, \quad (7.9)$$

where  $x$  is the input signal of the fuzzy system, where  $\xi(x)$  is the fuzzy vector,  $\varepsilon$  is approximation error of fuzzy system, and  $\varepsilon \leq \varepsilon_N$ .

The fuzzy system is used to approximate  $f$ . The fuzzy system input is selected as  $x = [x_1 \ x_2]^T$ , and the output of the fuzzy system is

$$\hat{f}(x|\theta_f) = \hat{\theta}_f^T \xi(x). \quad (7.10)$$

Control input item (7.4) is written as

$$u = -\hat{f} + \ddot{x}_{1d} + c\dot{e} + \eta \operatorname{sgn}(s). \quad (7.11)$$

Substituting Eq. (7.11) into Eq. (7.3), we have

$$\begin{aligned} \dot{s} &= \ddot{x}_{1d} - f - u + c\dot{e} = \ddot{\theta}_d - f - (-\hat{f} + \ddot{x}_{1d} + c\dot{e} + \eta \operatorname{sgn}(s)) + c\dot{e} \\ &= -f + \hat{f} - \eta \operatorname{sgn}(s) = -\tilde{f} - \eta \operatorname{sgn}(s), \end{aligned} \quad (7.12)$$

since

$$\tilde{f} = f - \hat{f} = f = \theta_f^{*T} \xi(x) + \varepsilon - \hat{\theta}_f^T \xi(x) = \tilde{\theta}_f^T \xi(x) + \varepsilon, \quad (7.13)$$

where  $\tilde{\theta}_f = \theta_f^* - \hat{\theta}_f$ .

Define the Lyapunov function as

$$V = \frac{1}{2}s^2 + \frac{1}{2}\gamma\tilde{\theta}_f^T\tilde{\theta}_f,$$

where  $\gamma > 0$ .

With a derivative  $V$ , and from Eqs. (7.4) and (7.12), we have

$$\begin{aligned}\dot{V} &= ss + \gamma\tilde{\theta}_f^T\dot{\tilde{\theta}}_f = s(-\tilde{f} - \eta \operatorname{sgn}(s)) - \gamma\tilde{\theta}_f^T\dot{\tilde{\theta}}_f \\ &= s\left(-\tilde{\theta}_f^T\xi(x) - \varepsilon - \eta \operatorname{sgn}(s)\right) - \gamma\tilde{\theta}_f^T\dot{\tilde{\theta}}_f \\ &= -\tilde{\theta}_f^T(s\xi(x) + \gamma\dot{\tilde{\theta}}_f) - s(\varepsilon + \eta \operatorname{sgn}(s)).\end{aligned}$$

Let the adaptive law be

$$\dot{\tilde{\theta}}_f = -\frac{1}{\gamma}s\xi(x), \quad (7.14)$$

then

$$\dot{V} = -s(\varepsilon + \eta \operatorname{sgn}(s)) = -s\varepsilon - \eta|s|.$$

Due to the approximation error  $\varepsilon$  is sufficiently small, if we design  $\eta \geq \varepsilon_N$ , we can obtain approximately  $\dot{V} \leq 0$ . Using LaSalle's invariance principle, if  $t \rightarrow \infty$ , then  $s \rightarrow 0$ , i.e.,  $e \rightarrow 0$  and  $\dot{e} \rightarrow 0$ .

### 7.1.3 Simulation example

Consider the following plant as

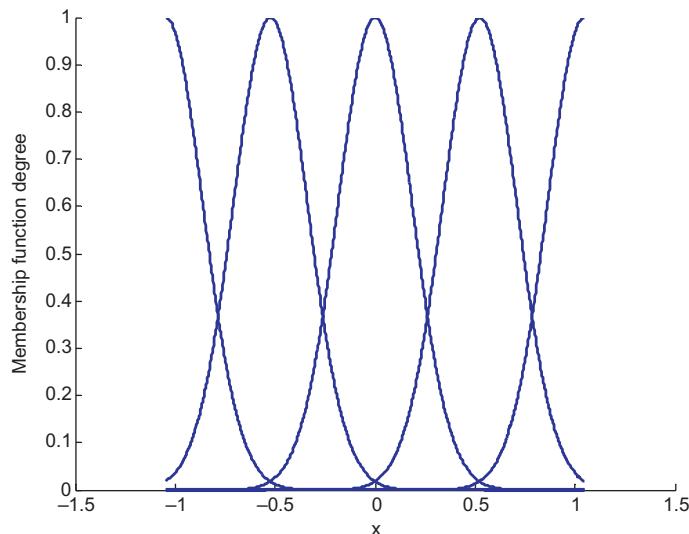
$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + u + d(t).\end{aligned}$$

To fuzzify  $x_1$  and  $x_2$ , consider the range value of  $x_1$  and  $x_2$ , we select five membership functions as:

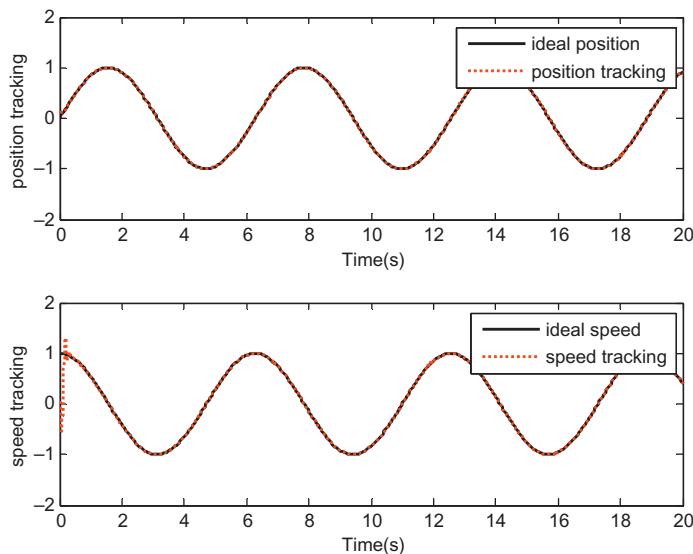
$$\begin{aligned}\mu_{NM}(x_i) &= \exp[-((x_i + \pi/3)/(\pi/12))^2], \quad \mu_{NS}(x_i) = \exp[-((x_i + \pi/6)/(\pi/12))^2], \\ \mu_Z(x_i) &= \exp[-(x_i/(\pi/12))^2], \quad \mu_{PS}(x_i) = \exp[-((x_i - \pi/6)/(\pi/12))^2], \\ \mu_{PM}(x_i) &= \exp[-((x_i - \pi/3)/(\pi/12))^2].\end{aligned}$$

The membership functions are given in Fig. 7.1.

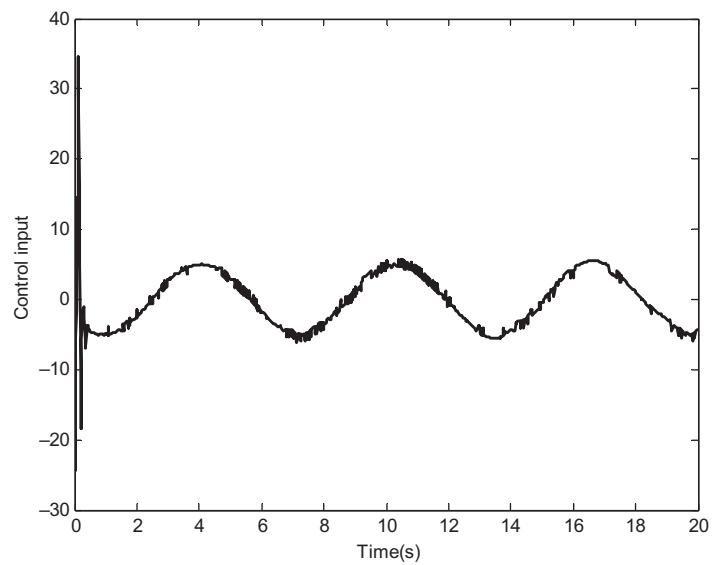
The desired trajectory is chosen as  $x_d(t) = \sin t$ , the initial states are  $[0.15, 0]$ , use control law (7.11) and adaptive law (7.14), choose  $c = 15$ ,  $\eta = 0.50$  and  $\gamma = 5000$ , the simulation results are shown in Figs. 7.2–7.4.



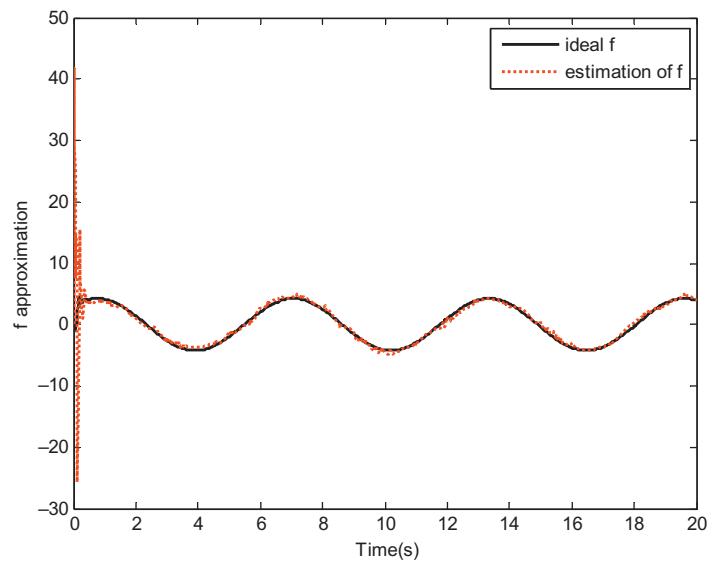
■ FIGURE 7.1 Membership function of  $\chi_i$ .



■ FIGURE 7.2 Position and speed tracking.



■ FIGURE 7.3 Control input.



■ FIGURE 7.4  $f(x)$  and  $\hat{f}(x)$ .

Simulation Programs:

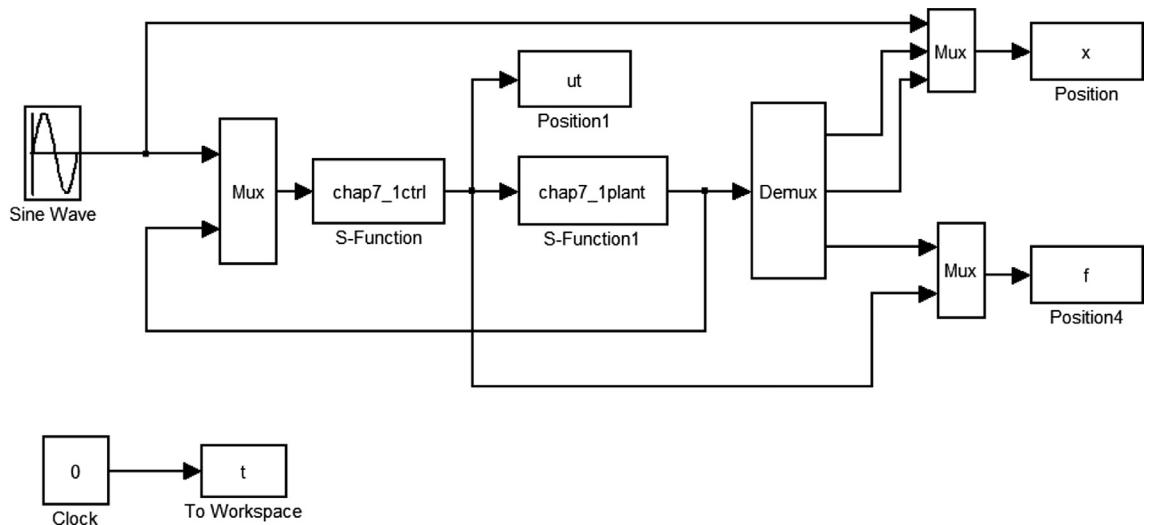
**1. Plot program: chap7\_1mf.m**

```
clear all;
close all;

L1 = -pi/3;
L2 = pi/3;
L = L2 - L1;

T = L*1/1000;
x = L1:T:L2;
figure(1);
for i = 1:1:5
 gs = -[(x + pi/3 - (i - 1)*pi/6)/(pi/12)].^2;
 u = exp(gs);
 hold on;
 plot(x,u,'linewidth',2);
end
xlabel('x'); ylabel('Membership function degree');
```

**2. Main Simulink program: chap7\_1sim.mdl**



**3. Control law program: chap7\_1ctrl.m**

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts] = mdlInitializeSizes;
```

```

case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates= 25;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0=[0.1*ones(25,1)];
str=[];
ts=[0 0];
function sys=mdlDerivatives(t,x,u)
xd=sin(t);
dxd=cos(t);

x1=u(2);
x2=u(3);
e=x1-xd;
de=x2-dxd;
c=15;
s=c*e+de;

xi=[x1;x2];

FS1=0;
for l1=1:1:5
 gs1=-[(x1+pi/3-(l1-1)*pi/6)/(pi/12)]^2;
 u1(l1)=exp(gs1);
end
for l2=1:1:5
 gs2=-[(x2+pi/3-(l2-1)*pi/6)/(pi/12)]^2;
 u2(l2)=exp(gs2);
end

```

```
for l1=1:1:5
 for l2=1:1:5
 FS2(5*(l1-1)+l2)=u1(l1)*u2(l2);
 FS1=FS1+u1(l1)*u2(l2);
 end
end
FS=FS2/(FS1+0.001);

for i=1:1:25
 thta(i,1)=x(i);
end
gama=5000;
S=gama*s*FS;

for i=1:1:25
 sys(i)=S(i);
end
function sys=mdlOutputs(t,x,u)
xd=sin(t);
dxd=cos(t);
ddxd=-sin(t);

x1=u(2);
x2=u(3);
e=x1-xd;
de=x2-dxd;
c=15;
s=c*e+de;

xi=[x1;x2];

FS1=0;
for l1=1:1:5
 gs1=-[(x1+pi/3-(l1-1)*pi/6)/(pi/12)]^2;
 u1(l1)=exp(gs1);
end
for l2=1:1:5
 gs2=-[(x2+pi/3-(l2-1)*pi/6)/(pi/12)]^2;
 u2(l2)=exp(gs2);
end
for l1=1:1:5
 for l2=1:1:5
 FS2(5*(l1-1)+l2)=u1(l1)*u2(l2);
 FS1=FS1+u1(l1)*u2(l2);
 end
end
```

```

 end
 FS=FS2/(FS1+0.001);
 for i=1:1:25
 thta(i,1)=x(i);
 end
 fxp=thta'*FS';
 xite=0.50;
 ut = - c*de + ddx - fxp - xite*sign(s);

 sys(1)=ut;
 sys(2)=fxp;

```

**4. Plant S function: chap7\_1plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates= 2;
sizes.NumDiscStates= 0;
sizes.NumOutputs= 3;
sizes.NumInputs= 2;
sizes.DirFeedthrough= 0;
sizes.NumSampleTimes= 0;
sys=simsizes(sizes);
x0=[0.15;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);

f=3*(x(1)+x(2));
sys(1)=x(2);
sys(2)=f+ut;
function sys=mdlOutputs(t,x,u)

```

```

f = 3*(x(1) + x(2));

sys(1)=x(1);
sys(2)=x(2);
sys(3)=f;
5. Plot program: chap7_1plot.m
close all;

figure(1);
subplot(211);
plot(t,x(:,1),'k',t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('position tracking');
legend('ideal position','position tracking');
subplot(212);
plot(t,cos(t),'k',t,x(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('speed tracking');
legend('ideal speed','speed tracking');

figure(2);
plot(t,ut(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,f(:,1),'k',t,f(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('f approximation');
legend('ideal f','estimation of f');

```

## 7.2 SLIDING MODE CONTROL BASED ON A FUZZY SYSTEM WITH MINIMUM PARAMETER LEARNING METHOD

The minimum parameter learning (MPL) method [4] can decrease the number of online adaptive parameters to only one, which can be used in fuzzy system to reduce the computational burden and increase real-time performance. Using the minimum parameter learning, in this section, a sliding mode control based on a fuzzy system is introduced.

### 7.2.1 Problem statement

Consider a second-order system as follows:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + u + d(t),\end{aligned}\tag{7.15}$$

where  $x_1$  and  $x_2$  are respectively angle and angle speed,  $u$  is the control input,  $d(t)$  is disturbance,  $|d(t)| \leq D$ .

Denote  $x_{1d}$  as an ideal angle and define the tracking error as

$$e = x_1 - x_{1d}.$$

Design a sliding mode function as

$$s = \dot{e} + ce, \quad (7.16)$$

where  $c > 0$ , then

$$\dot{s} = \ddot{e} + c\dot{e} = \ddot{x}_1 - \ddot{x}_{1d} + c\dot{e} = f + u + d - \ddot{x}_{1d} + c\dot{e}.$$

If  $f$  is known, we can design a control law as

$$u = -f + \ddot{x}_{1d} - c\dot{e} - \eta \operatorname{sgn}(s). \quad (7.17)$$

Then  $\dot{s} = -\eta \operatorname{sgn}(s) + d$ ; therefore, if we design  $\eta \geq D$ , we have

$$s\dot{s} = -\eta|s| + \eta d \leq 0.$$

If  $f(x)$  is unknown, we should approximate  $f(x)$  by some algorithms. In the following, we will simply recall fuzzy systems approximate unknown function  $f(x)$ .

### 7.2.2 Uncertainty approximation using a fuzzy system

If  $f(x)$  is unknown, we can replace  $f(x)$  with the fuzzy system  $\hat{f}(x)$  to realize feedback control. Using the universal approximation theorem, three steps are designed as follows:

1. For  $x_1$  and  $x_2$ , define the fuzzy sets  $A_1^{l_i}$  and  $A_2^{l_i}$ , respectively,  
 $l_i = 1, 2, \dots, 5$ ;
2. Design  $\prod_{i=1}^n p_i = p_1 \times p_2 = 25$  fuzzy rules to construct fuzzy system  
 $\hat{f}(x|\theta_f)$ :

$$\begin{aligned} R^{(1)}: & \text{ If } x_1 \text{ is } A_1^1 \text{ and...and } x_2 \text{ is } A_2^1 \text{ then } \hat{f} \text{ is } B_1^1, \\ R^{(25)}: & \text{ If } x_1 \text{ is } A_1^5 \text{ and...and } x_2 \text{ is } A_2^5 \text{ then } \hat{f} \text{ is } B_2^{25}, \end{aligned} \quad (7.18)$$

where  $l_i = 1, 2, \dots, 5$ ,  $i = 1, 2$ ,  $p_1 = p_2 = 5$ .

3. Using fuzzy inference, the output of a fuzzy system is

$$\hat{f}(x|\theta_f) = \frac{\sum_{l_1=1}^5 \sum_{l_2=1}^5 \bar{y}_f^{l_1 l_2} \left( \prod_{i=1}^2 \mu_{A_i^{l_i}}(x_i) \right)}{\sum_{l_1=1}^5 \sum_{l_2=1}^5 \left( \prod_{i=1}^2 \mu_{A_i^{l_i}}(x_i) \right)}, \quad (7.19)$$

where  $\mu_{A_i^{l_i}}(x_i)$  is the membership function of  $x_i$ .

Let  $\bar{y}_f^{l_1 l_2}$  be a free parameter and be put in the set  $\hat{\theta}_f \in R^{(25)}$ . Column vector  $\xi(x)$  is introduced and Eq. (7.19) can be written as:

$$\hat{f}(x|\theta_f) = \hat{\theta}_f^T \xi(x), \quad (7.20)$$

where  $x = [x_1 \ x_2]^T$ ,  $\xi(x)$  is the  $\prod_{i=1}^n p_i = p_1 \times p_2 = 25$  dimensional column vector, and  $l_1, l_2$  elements are respectively

$$\xi_{l_1 l_2}(x) = \frac{\prod_{i=1}^2 \mu_{A_i^{l_i}}(x_i)}{\sum_{l_1=1}^{p_1} \sum_{l_2=1}^{p_2} \left( \prod_{i=1}^2 \mu_{A_i^{l_i}}(x_i) \right)}. \quad (7.8)$$

The membership functions are needed to be selected according to experiences. Moreover, all the states must be known.

### 7.2.3 Design of an adaptive fuzzy sliding mode controller with MPL

Suppose the optimal parameter is

$$\theta_f^* = \arg \min_{\theta_f \in \Omega_f} \left[ \sup_{x \in R^n} |\hat{f}(x|\theta_f) - f(x)| \right],$$

where  $\Omega_f$  is the set of  $\theta_f$ , i.e.,  $\theta_f \in \Omega_f$ .

The term  $f$  can be expressed as

$$f = \theta_f^{*T} \xi(x) + \varepsilon, \quad (7.21)$$

where  $x$  is the input signal of the fuzzy system, where  $\xi(x)$  is the fuzzy vector,  $\varepsilon$  is approximation error of fuzzy system, and  $\varepsilon \leq \varepsilon_N$ .

The fuzzy system is used to approximate  $f$ . The fuzzy system input is selected as  $x = [x_1 \ x_2]^T$ , and the output of the fuzzy system is

$$\hat{f}(x|\theta_f) = \hat{\theta}_f^T \xi(x). \quad (7.22)$$

Using minimum parameter learning [4], define  $\phi = ||\theta_f^*||^2$ , where  $\phi$  is a positive constant, and let  $\hat{\phi}$  be an estimation of  $\phi$ .

Design the controller as

$$u = -\frac{1}{2} s \hat{\phi} \xi^T \xi + \ddot{x}_{1d} - c \dot{e} - \eta \operatorname{sgn}(s) - \mu s, \quad (7.23)$$

where  $\eta \geq \varepsilon_N + D$ ,  $\mu > 0$ .

Then we have

$$\begin{aligned}\dot{s} &= f + u + d - \ddot{x}_{1d} + c\dot{e} = f - \frac{1}{2}s\hat{\phi}\boldsymbol{\xi}^T\boldsymbol{\xi} + \ddot{x}_{1d} - c\dot{e} - \eta \operatorname{sgn}(s) - \mu s + d - \ddot{x}_{1d} + c\dot{e} \\ &= \boldsymbol{\theta}_f^{*\top}\boldsymbol{\xi}(x) + \varepsilon - \frac{1}{2}s\hat{\phi}\boldsymbol{\xi}^T\boldsymbol{\xi} - \eta \operatorname{sgn}(s) - \mu s + d.\end{aligned}$$

The Lyapunov function is designed as

$$V = \frac{1}{2}s^2 + \frac{1}{2\gamma}\tilde{\phi}^2,$$

where  $\gamma > 0$ ,  $\tilde{\phi} = \hat{\phi} - \phi$ .

Then

$$\begin{aligned}\dot{V} &= s\dot{s} + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} \\ &= s\left(\boldsymbol{\theta}_f^{*\top}\boldsymbol{\xi}(x) + \varepsilon - \frac{1}{2}s\hat{\phi}\boldsymbol{\xi}^T\boldsymbol{\xi} - \eta \operatorname{sgn}(s) - \mu s + d\right) + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} \\ &\leq \frac{1}{2}s^2\phi\boldsymbol{\xi}^T\boldsymbol{\xi} + \frac{1}{2} + \varepsilon s - \frac{1}{2}s^2\hat{\phi}\boldsymbol{\xi}^T\boldsymbol{\xi} - \eta|s| - \mu s^2 + ds + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} \\ &= -\frac{1}{2}s^2\tilde{\phi}\boldsymbol{\xi}^T\boldsymbol{\xi} + \frac{1}{2} + \varepsilon s - \eta|s| - \mu s^2 + ds + \frac{1}{\gamma}\tilde{\phi}\dot{\phi} \\ &= \tilde{\phi}\left(-\frac{1}{2}s^2\boldsymbol{\xi}^T\boldsymbol{\xi} + \frac{1}{\gamma}\dot{\phi}\right) + \frac{1}{2} - \mu s^2 + \varepsilon s - \eta|s| + ds \\ &\leq \tilde{\phi}\left(-\frac{1}{2}s^2\boldsymbol{\xi}^T\boldsymbol{\xi} + \frac{1}{\gamma}\dot{\phi}\right) + \frac{1}{2} - \mu s^2.\end{aligned}$$

Design the adaptive law as

$$\dot{\phi} = \frac{\gamma}{2}s^2\boldsymbol{\xi}^T\boldsymbol{\xi} - \kappa\gamma\hat{\phi}, \quad (7.24)$$

where  $\kappa > 0$ .

Then

$$\dot{V} \leq -\kappa\tilde{\phi}\hat{\phi} + \frac{1}{2} - \mu s^2 \leq -\frac{\kappa}{2}(\tilde{\phi}^2 - \phi^2) + \frac{1}{2} - \mu s^2 = -\frac{\kappa}{2}\tilde{\phi}^2 - \mu s^2 + \left(\frac{\kappa}{2}\phi^2 + \frac{1}{2}\right).$$

Define  $\kappa = \frac{2\mu}{\gamma}$ , then

$$\begin{aligned}\dot{V} &\leq -\frac{\mu}{\gamma}\tilde{\phi}^2 - \mu s^2 + \left(\frac{\kappa}{2}\phi^2 + \frac{1}{2}\right) = -2\mu\left(\frac{1}{2\gamma}\tilde{\phi}^2 + \frac{1}{2}s^2\right) + \left(\frac{\kappa}{2}\phi^2 + \frac{1}{2}\right) \\ &= -2\mu V + Q,\end{aligned}$$

where  $Q = \frac{\kappa}{2}\phi^2 + \frac{1}{2}$ .

Using Lemma 1.3, the solution of  $\dot{V} \leq -2\mu V + Q$  is

$$V \leq \frac{Q}{2\mu} + \left( V(0) - \frac{Q}{2\mu} \right) e^{-2\mu t},$$

then

$$\lim_{t \rightarrow \infty} V = \frac{Q}{2\mu} = \frac{\frac{\kappa}{2}\phi^2 + \frac{1}{2}}{2\mu} = \frac{\kappa\phi^2 + 1}{4\mu} = \frac{\frac{2\mu}{\gamma}\phi^2 + 1}{4\mu} = \frac{\phi^2}{2\gamma} + \frac{1}{4\mu}. \quad (7.25)$$

From above, we can see that the convergence precision depends on  $\gamma$  and  $\mu$ .

Regarding the above analysis, two remarks are given as follows.

**Remark 1:**  $s^2\phi\xi^T\xi + 1 = s^2||\theta_f^*||^2\xi^T\xi + 1 = s^2||\theta_f^*||^2||\xi||^2 + 1 = s^2||\theta_f^{*T}\xi||^2 + 1 \geq 2s\theta_f^{*T}\xi$ , i.e.,  $s\theta_f^{*T}\xi \leq \frac{1}{2}s^2\theta_f^*\xi^T\xi + \frac{1}{2}$ ;

**Remark 2:** since  $(\tilde{\phi} + \phi)^2 \geq 0$ , then  $\tilde{\phi}^2 + 2\tilde{\phi}\phi + \phi^2 \geq 0$  and  $\tilde{\phi}^2 + 2\tilde{\phi}(\hat{\phi} - \tilde{\phi}) + \phi^2 \geq 0$ , i.e.,  $2\tilde{\phi}\hat{\phi} \geq \tilde{\phi}^2 - \phi^2$ .

However, the shortcoming of minimum parameter learning method is that the approximation precision cannot be guaranteed.

#### 7.2.4 Simulation example

Consider the following plant:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(x) + u + d(t),\end{aligned}$$

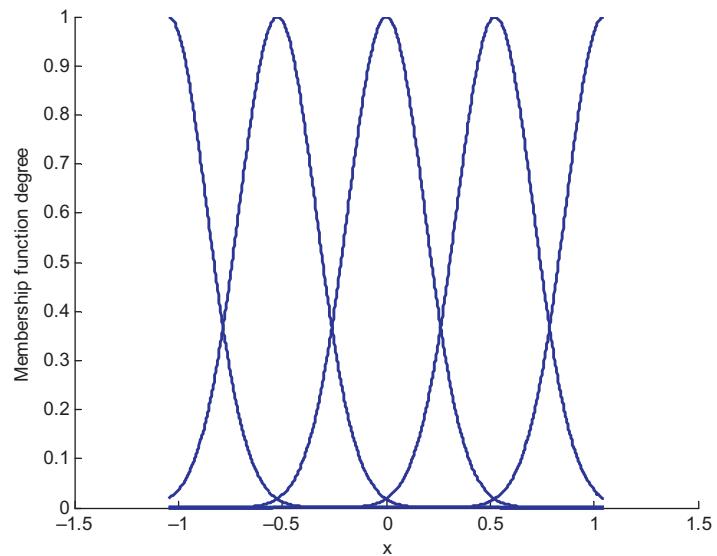
where  $f(x) = 3(x_1 + x_2)$ ,  $d(t) = \sin t$ .

To fuzzify  $x_1$  and  $x_2$ , consider the range value of  $x_1$  and  $x_2$ , we select five membership functions as:

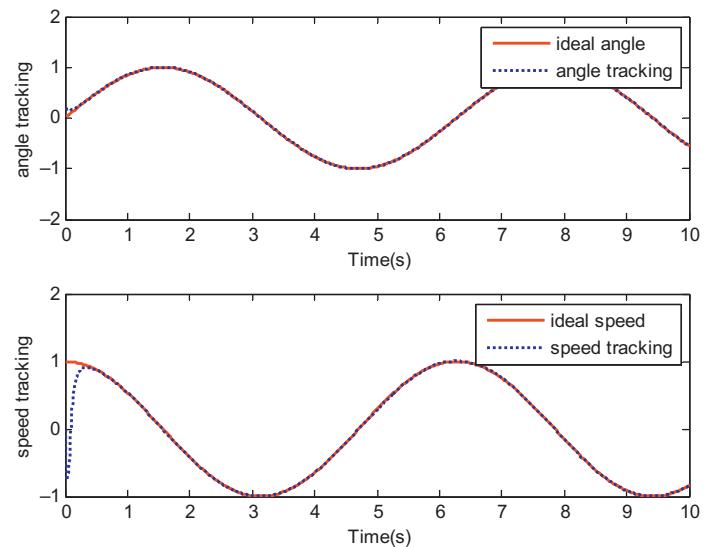
$$\begin{aligned}\mu_{NM}(x_i) &= \exp[-((x_i + \pi/3)/(\pi/12))^2], \quad \mu_{NS}(x_i) = \exp[-((x_i + \pi/6)/(\pi/12))^2], \\ \mu_Z(x_i) &= \exp[-(x_i/(\pi/12))^2], \quad \mu_{PS}(x_i) = \exp[-((x_i - \pi/6)/(\pi/12))^2], \\ \mu_{PM}(x_i) &= \exp[-((x_i - \pi/3)/(\pi/12))^2].\end{aligned}$$

The membership functions are given in Fig. 7.5; its program is also chap7\_1mf.m.

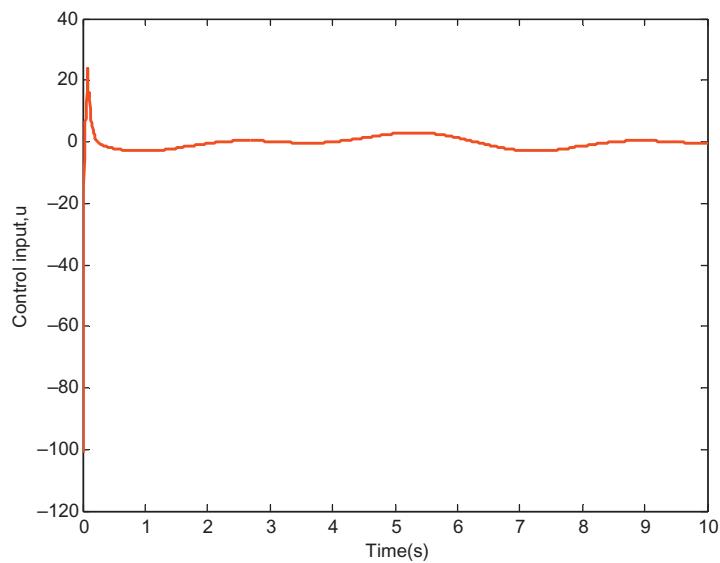
Choose the desired trajectory as  $x_d(t) = \sin t$ , the initial states are set as  $[0.20, 0]$ , use control law (7.23) and adaptive law (7.24), choose  $\eta = 1.1$ ,  $\mu = 10$  and  $\gamma = 100$ . To reduce chattering, use the saturation function instead of the sign function, and choose  $\Delta = 0.02$ , the simulation results are shown in Figs. 7.6–7.8.



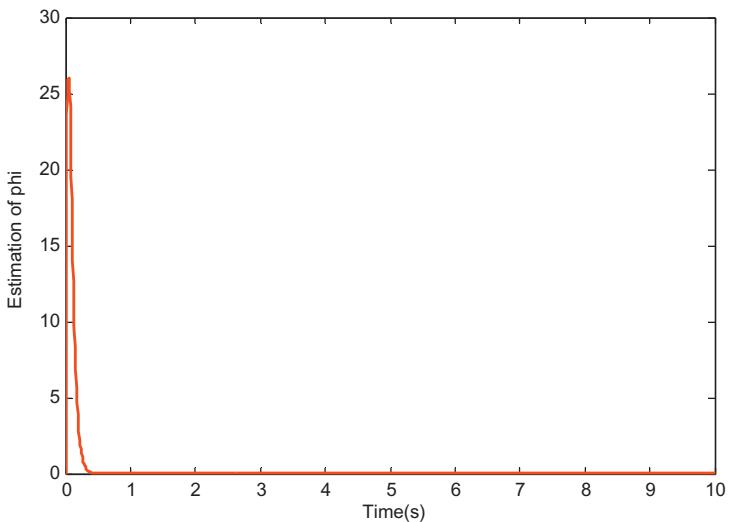
■ FIGURE 7.5 Membership function of  $x_i$ .



■ FIGURE 7.6 Position and speed tracking.



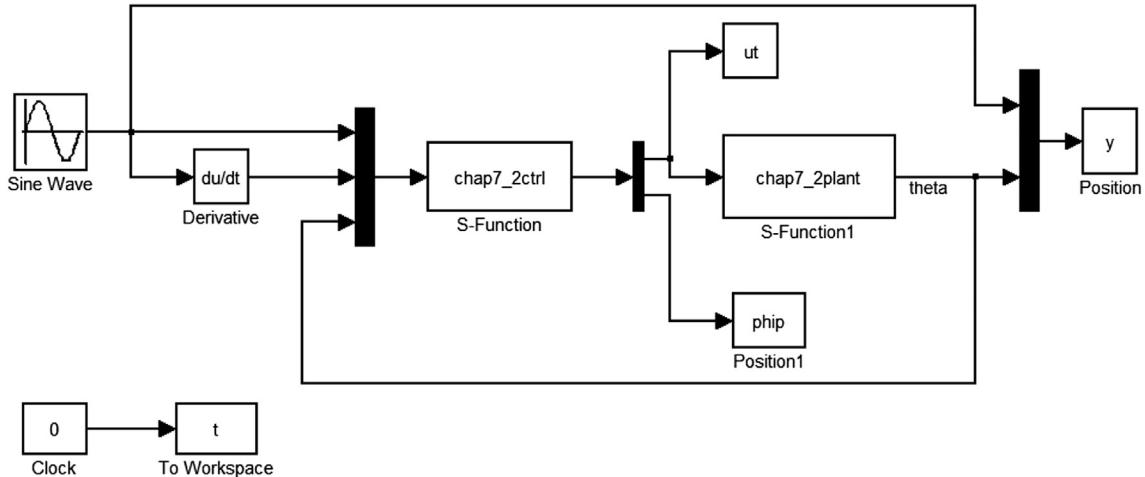
■ FIGURE 7.7 Control input.



■ FIGURE 7.8 Change of  $\hat{\phi}$ .

Simulation Programs:

## 1. Main Simulink program: chap7\_2sim.mdl



## 2. Control law program: chap7\_2ctrl.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 1;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);

```

```
x0 = 0;
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
x1d=u(1);
dx1d=u(2);
x1=u(3);
x2=u(4);
e=x1-x1d;
de=x2-dx1d;
c=50;
s=c*e+de;
xi=[x1;x2];
FS1=0;
for l1=1:1:5
 gs1=-[(x1+pi/3-(l1-1)*pi/6)/(pi/12)]^2;
 u1(l1)=exp(gs1);
end
for l2=1:1:5
 gs2=-[(x2+pi/3-(l2-1)*pi/6)/(pi/12)]^2;
 u2(l2)=exp(gs2);
end
for l1=1:1:5
 for l2=1:1:5
 FS2(5*(l1-1)+l2)=u1(l1)*u2(l2);
 FS1=FS1+u1(l1)*u2(l2);
 end
end
FS=FS2/(FS1+0.001);

miu=10;
gama=100;
k=2*miu/gama;
sys(1)=gama/2*s^2*FS*FS'-k*gama*x;

function sys=mdlOutputs(t,x,u)
x1d=u(1);
dx1d=u(2);
x1=u(3);
x2=u(4);
```

```

e = x1 - x1d;
de = x2 - dx1d;
x1d = sin(t);
dx1d = cos(t);
ddx1d = -sin(t);

c = 50;
s = c*e + de;
phi = x;
xi = [x1;x2];
xi = [x1;x2];

FS1 = 0;
for l1 = 1:1:5
 gs1 = -[(x1 + pi/3 - (l1-1)*pi/6)/(pi/12)]^2;
 u1(l1) = exp(gs1);
end
for l2 = 1:1:5
 gs2 = -[(x2 + pi/3 - (l2-1)*pi/6)/(pi/12)]^2;
 u2(l2) = exp(gs2);
end
for l1 = 1:1:5
 for l2 = 1:1:5
 FS2(5*(l1-1) + l2) = u1(l1)*u2(l2);
 FS1 = FS1 + u1(l1)*u2(l2);
 end
end
FS = FS2/(FS1 + 0.001);

xite = 1.0 + 0.10;
miu = 10;

%Saturated function
fai = 0.02;
if abs(s) <= fai
 sats = s/fai;
else
 sats = sign(s);
end
%ut = -0.5*s*phi*FS*FS' + ddx1d - c*de - xite*sign(s) - miu*s;
ut = -0.5*s*phi*FS*FS' + ddx1d - c*de - xite*sats - miu*s;

sys(1) = ut;
sys(2) = x(1);

```

**3. Plant S function: chap7\_2plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=2;
sizes.NumDiscStates=0;
sizes.NumOutputs=2;
sizes.NumInputs=1;
sizes.DirFeedthrough=0;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[0.20 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
%%%%%%%%%%%%%
dt=sin(t);
%%%%%%%%%%%%%
fx=3*x(1)*x(2);
sys(1)=x(2);
sys(2)=fx+u+dt;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);

```

**4. Plot program: chap7\_2plot.m**

```

close all;

figure(1);
subplot(211);
plot(t,y(:,1),'r',t,y(:,2),'b','linewidth',2);

```

```
xlabel('time(s)');ylabel('angle tracking');
legend('ideal angle','angle tracking');
subplot(212);
plot(t,cos(t),'r',t,y(:,3),'b:','linewidth',2);
xlabel('time(s)');ylabel('speed tracking');
legend('ideal speed','speed tracking');
figure(2);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input,u');
figure(3);
plot(t,phip(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Estimation of phi');
```

## REFERENCES

- [1] L.X. Wang, A Course in Fuzzy System and Control, Prentice Hall, Upper Saddle River, NJ, 1997.
- [2] L.X. Wang, Stable adaptive fuzzy control of nonlinear systems, IEEE Trans. Fuzzy Syst. 1 (2) (1993) 146–155.
- [3] J. Liu, X. Wang, Advanced Sliding Mode Control for Mechanical Systems\_Design, Analysis and Matlab Simulation, Tsinghua & Springer Press, Beijing, 2011.
- [4] Bing Chen, X. Liu, K. Liu, C. Lin, Direct adaptive fuzzy control of nonlinear strict-feedback systems, Automatica 45 (2009) 1530–1535.

# Sliding mode control of a class of underactuated systems

The past decade there has been increasing interest in underactuated systems. These systems are characterized by the fact that they have fewer actuators than the degrees of freedom to be controlled. Underactuated systems have very important applications such as free-flying space robots, underwater robots, surface vessels, manipulators with structural flexibility, etc. They are used for reducing weight, cost or energy consumption, while still maintaining an adequate degree of dexterity without reducing the reachable configuration space. Some other advantages of underactuated systems include less damage while hitting an object, and tolerance for failure of actuators [1,2].

Control researchers have given considerable attention to many examples of control problems associated with underactuated mechanical systems, and different control strategies have been proposed, in this chapter, mainly refer to [1], we discuss several methods to design the sliding mode control for underactuated systems.

## 8.1 SLIDING MODE CONTROL OF A CLASS OF UNDERACTUATED SYSTEM

### 8.1.1 System description

Consider an underactuated system as follows:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_1(x_1, x_2, x_3) \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= f_2(x_1, x_2, x_3) + b(x_1, x_2, x_3)u + d,\end{aligned}\tag{8.1}$$

where  $u$  is control input,  $d$  is disturbance,  $|d| \leq D$ .

The function  $f_1(x_1, x_2, x_3)$  must be satisfied following four assumptions as follows:

Assumption 1:  $f_1(0, 0, 0) \rightarrow 0$ ;

Assumption 2:  $\frac{\partial f_1}{\partial x_3}$  is invertible;

Assumption 3:  $x_3 \rightarrow 0$  if  $f_1(0, 0, x_3) \rightarrow 0$ .

Assumption 4:  $\left| \frac{\partial f_1}{\partial x_3} \right| \leq \beta_3, i = 1, 2, 3$ .

The control goals are  $x_i \rightarrow 0, i = 1, 2, 3, 4$  as  $t \rightarrow \infty$ . Define

$$\begin{aligned} e_1 &= x_1 \\ e_2 &= \dot{e}_1 = x_2 \\ e_3 &= \ddot{e}_1 = \dot{x}_2 = f_1(x_1, x_2, x_3) \\ e_4 &= \ddot{e}_1 = \dot{f}_1 + \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_2} f_1 + \frac{\partial f_1}{\partial x_3} x_4 \end{aligned} \quad (8.2)$$

### 8.1.2 Sliding mode controller design

Design the sliding mode function as

$$s = c_1 e_1 + c_2 e_2 + c_3 e_3 + e_4, \quad (8.3)$$

where  $c_i > 0$  and  $i = 1, 2, 3$ .

Then

$$\begin{aligned} \dot{s} &= c_1 \dot{e}_1 + c_2 \dot{e}_2 + c_3 \dot{e}_3 + \dot{e}_4 = c_1 x_2 + c_2 f_1 + c_3 \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_2} f_1 + \frac{\partial f_1}{\partial x_3} x_4 \right) \\ &\quad + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_1} x_2 \right] + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_2} f_1 \right] + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_3} \right] x_4 + \frac{\partial f_1}{\partial x_3} (f_2 + bu + d) \end{aligned} \quad (8.4)$$

Let  $\dot{s} = 0$ , then we can get equivalent control as

$$\begin{aligned} u_{eq} &= - \left[ \frac{\partial f_1}{\partial x_3} b \right]^{-1} \left\{ c_1 x_2 + c_2 f_1 + c_3 \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_2} f_1 + \frac{\partial f_1}{\partial x_3} x_4 \right) + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_1} x_2 \right] \right. \\ &\quad \left. + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_2} f_1 \right] + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_3} \right] x_4 + \frac{\partial f_1}{\partial x_3} f_2 \right\} \end{aligned} \quad (8.5)$$

To satisfy  $s\dot{s} \leq 0$ , controller can be designed as

$$u = u_{eq} + u_{sw}, \quad (8.6)$$

where  $u_{sw} = - \left[ \frac{\partial f_1}{\partial x_3} b \right]^{-1} [M \operatorname{sgn}(s) + \lambda s], \lambda > 0$ .

Substituting Eq. (8.6) into Eq. (8.4), we have

$$\begin{aligned}\dot{s} &= c_1\dot{e}_1 + c_2\dot{e}_2 + c_3\dot{e}_3 + \dot{e}_4 = c_1x_2 + c_2f_1 + c_3\left(\frac{\partial f_1}{\partial x_1}x_2 + \frac{\partial f_1}{\partial x_2}f_1 + \frac{\partial f_1}{\partial x_3}x_4\right) \\ &\quad + \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_1}x_2\right] + \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_2}f_1\right] + \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_3}\right]x_4 + \frac{\partial f_1}{\partial x_3}(f_2 + bu + d) \\ &= -M \operatorname{sgn}(s) - \lambda s + \frac{\partial f_1}{\partial x_3}d.\end{aligned}$$

Define

$$M = \beta_3 D + \rho, \rho > 0. \quad (8.7)$$

Design the Lyapunov function as  $V = \frac{1}{2}s^2$ , then

$$\begin{aligned}\dot{V} &= s\dot{s} = s\left(-(\beta_3 D + \rho)\operatorname{sgn}(s) - \lambda s + \frac{\partial f_1}{\partial x_3}d\right) \\ &= -(\beta_3 D + \rho)|s| - \lambda s^2 + s\frac{\partial f_1}{\partial x_3}d \leq -\rho|s| - \lambda s^2 \leq 0.\end{aligned}$$

Since

$$\begin{aligned}\dot{e}_1 &= x_2 \\ \dot{e}_2 &= \dot{x}_2 = f_1 \\ \dot{e}_3 &= \dot{f}_1 = \frac{\partial f_1}{\partial x_1}x_2 + \frac{\partial f_1}{\partial x_2}f_1 + \frac{\partial f_1}{\partial x_3}x_4 \\ \dot{e}_4 &= \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_1}x_2\right] + \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_2}f_1\right] + \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_3}\right]x_4 \\ &= \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_1}x_2\right] + \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_2}f_1\right] + \frac{d}{dt}\left[\frac{\partial f_1}{\partial x_3}\right]x_4 + \frac{\partial f_1}{\partial x_3}(f_2 + bu + d).\end{aligned}$$

From Eq. (8.2), we have  $\dot{e}_1 = e_2$ ,  $\dot{e}_2 = e_3$  and  $\dot{e}_3 = e_4$ .  $s\dot{s} \leq 0$  indicates that there exists  $s = 0$  as  $t > t_0$ , when  $s = 0$ , we have  $e_4 = -c_1e_1 - c_2e_2 - c_3e_3$ ,

define  $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -c_1 & -c_2 & -c_3 \end{bmatrix}$ ,  $E_1 = [e_1 \ e_2 \ e_3]^T$ , then we have

$$\dot{E}_1 = AE_1 \quad (8.8)$$

Choose  $Q = Q^T > 0$ , if we design  $A$  as Hurwitz, then there exists a Lyapunov equation  $A^T P + PA = -Q$ ,  $P = P^T > 0$ .

Design the Lyapunov function as  $V_1 = E_1^T P E_1$ , then

$$\begin{aligned}\dot{V}_1 &= \dot{E}_1^T P E_1 + E_1^T P \dot{E}_1 = (AE_1)^T P E_1 + E_1^T P (AE_1) \\ &= E_1^T A^T P E_1 + E_1^T P A E_1 = E_1^T (A^T P + P A) E_1 \\ &= -E_1^T Q E_1 \leq -\lambda_{\min}(Q) \|E_1\|_2^2 \leq 0,\end{aligned}$$

where  $\lambda_{\min}(Q)$  is the minimum eigenvalue of  $Q$ .

In the above analysis, three remarks are given as follows.

**Remark 1:** From  $\dot{V}_1 \leq 0$ , we have  $e_1 \rightarrow 0$ ,  $e_2 \rightarrow 0$ ,  $e_3 \rightarrow 0$ , consider  $s = c_1 e_1 + c_2 e_2 + c_3 e_3 + e_4 \rightarrow 0$ , we have  $e_4 \rightarrow 0$ . From  $e_1 \rightarrow 0$  and  $e_2 \rightarrow 0$ , we can get  $x_1 \rightarrow 0$ ,  $x_2 \rightarrow 0$ , consider  $e_3 = f_1(0, 0, x_3) \rightarrow 0$  and Assumption 3, we have  $x_3 \rightarrow 0$ . From Eq. (8.2), we have  $x_4 \rightarrow 0$ .

**Remark 2:** to guarantee that  $A$  is Hurwitz, let

$$|A - \lambda I| = \begin{vmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \\ -c_1 & -c_2 & -c_3 - \lambda \end{vmatrix} = \lambda^2(-c_3 - \lambda) - c_1 - c_2\lambda = -\lambda^3 - c_3\lambda^2 - c_2\lambda - c_1 = 0, \text{ from } (\lambda + 3)^3 = 0, \text{ we have } \lambda^3 + 9\lambda^2 + 27\lambda + 27 = 0 \text{ then we can set } c_1 = 27 \text{ and } c_2 = 27, c_3 = 9.$$

**Remark 3:** three assumptions can be explained as follows:

1. The closed system stability requires that  $e_3 = 0$ ,  $x_1 = 0$ ,  $x_2 = 0$ ,  $x_3 = 0$  exist at the same time, i.e., Assumption 1  $f_1(0, 0, 0) \rightarrow 0$  must be satisfied;
2. only Assumption 2 exists, i.e.,  $\frac{\partial f_1}{\partial x_3}$  must be invertible, control law (8.6) can be effective;
3. only Assumption 3 exists, i.e.,  $f_1(0, 0, x_3) = 0$  and  $x_3 \rightarrow 0$  exists at the same time,  $x_3 \rightarrow 0$  can be guaranteed.

### 8.1.3 Position tracking

If the tracking control goals are set as  $x_1 \rightarrow x_d$  and  $x_2 \rightarrow \dot{x}_d$ , then we have

$$\begin{aligned} e_1 &= x_1 - x_d \\ e_2 &= \dot{e}_1 = x_2 - \dot{x}_d \\ e_3 &= \ddot{e}_1 = \dot{x}_2 - \ddot{x}_d = f_1(x_1, x_2, x_3) - \ddot{x}_d \\ e_4 &= \ddot{e}_1 = \dot{f}_1 - \ddot{x}_d = \frac{\partial f_1}{\partial x_1}x_2 + \frac{\partial f_1}{\partial x_2}f_1 + \frac{\partial f_1}{\partial x_3}x_4 - \ddot{x}_d \end{aligned}$$

and

$$\begin{aligned} \dot{s} &= c_1 \dot{e}_1 + c_2 \dot{e}_2 + c_3 \dot{e}_3 + \dot{e}_4 \\ &= c_1 \dot{x}_1 + c_2 \dot{x}_2 + c_3 \ddot{x}_2 + \dot{f}_1 - c_1 \dot{x}_d - c_2 \ddot{x}_d - c_3 \ddot{x}_d - \ddot{x}_d \end{aligned}$$

Let  $\dot{s} = 0$ ; we have

$$\begin{aligned} u_{eq} &= - \left[ \frac{\partial f_1}{\partial x_3} b \right]^{-1} \left\{ c_1 x_2 + c_2 f_1 + c_3 \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_2} f_1 + \frac{\partial f_1}{\partial x_3} x_4 \right) + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_1} x_2 \right] \right. \\ &\quad \left. + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_2} f_1 \right] + \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_3} \right] x_4 + \frac{\partial f_1}{\partial x_3} f_2 - c_1 \dot{x}_d - c_2 \ddot{x}_d - c_3 \ddot{x}_d - \ddot{x}_d \right\} \end{aligned} \quad (8.9)$$

### 8.1.4 Simulation example

#### 8.1.4.1 First example

Consider a simple underactuated system as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= g \sin x_1/l + x_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u - d,\end{aligned}$$

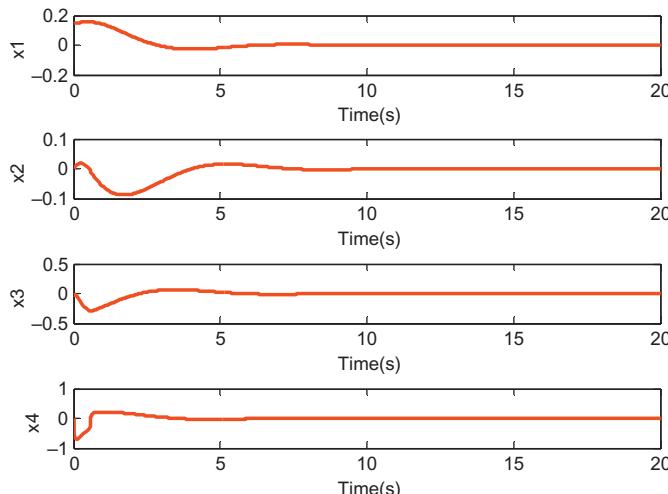
where  $f_1 = g \sin x_1/l + x_3$ ,  $f_2 = 0$ ,  $d = 10 \sin t$ .

Consider control goals as  $x_i \rightarrow 0, i = 1, 2, 3, 4$ . From the model, we have

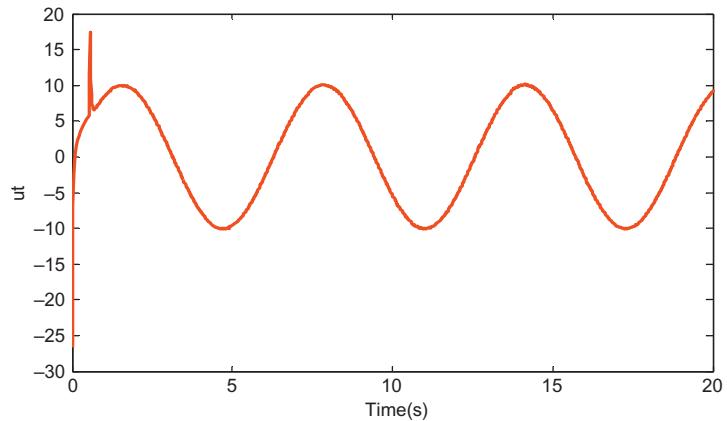
$$\begin{aligned}\frac{\partial f_1}{\partial x_1} &= g \cos x_1/l, \quad \frac{\partial f_1}{\partial x_2} = 0, \quad \frac{\partial f_1}{\partial x_3} = 1, \quad \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_1} x_2 \right] = \frac{g}{l} \frac{d}{dt} (x_2 \cos x_1) = \\ &\frac{g}{l} (f_1 \cos x_1 - x_2 \sin x_1), \quad \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_2} f_1 \right] = 0, \quad \frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_3} \right] x_4 = 0.\end{aligned}$$

The initial system states are set as  $[0.15 \ 0 \ 0 \ 0]$ .

From  $\frac{\partial f_1}{\partial x_3} = 1$  we can choose  $\beta_3 = 1.1$ , use controllers (8.5) and (8.6), design  $M$  from Eq. (8.7), and choose  $c_1 = 27$ ,  $c_2 = 27$ ,  $c_3 = 9$ ,  $\rho = 1.0$ ,  $\lambda = 1.0$  and  $D = 10$ , adapt the saturation function instead of the switch function, set  $\Delta = 0.05$ , simulation results are shown in Figs. 8.1 and 8.2.



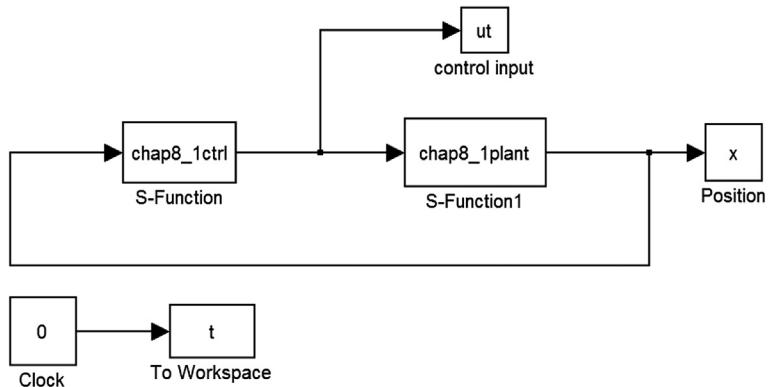
■ FIGURE 8.1 System states response.



■ FIGURE 8.2 Control input.

Simulation Programs:

1. Main program: chap8\_1sim.mdl



2. S function of controller: chap8\_1ctrl.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
 sys = mdlOutputs(t,x,u);
case {2,4,9}
 sys = [];
end

```

```
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
x1=u(1);
x2=u(2);
x3=u(3);
x4=u(4);
c1=27;c2=27;c3=9;

l=10;g=9.8;
f1=g*sin(x1)/l+x3;

b=1;
f2=0;
f1_x1=g*cos(x1)/l;
f1_x2=0;
f1_x3=1;
beta3=1.0+0.1;
D=-g/l*sin(x1)*x2^2+g/l*cos(x1)*f1;

ueq=-inv(f1_x3*b)*(c1*x2+c2*f1+c3*f1_x1*x2+c3*f1_x2
*f1+c3*f1_x3*x4+D);

e1=x1;
e2=x2;
e3=f1;
e4=f1_x1*x2+f1_x2*f1+f1_x3*x4;
s=c1*e1+c2*e2+c3*e3+e4;
```

```

d_up = 10;

rou = 1.0;
M = beta3*d_up + rou;
nmn = 1.0;

S = 2;
if S == 1
 sat = sign(s);
elseif S == 2 %Saturated function
 fai = 0.05;
 if abs(s) <= fai
 sat = s/fai;
 else
 sat = sign(s);
 end
end
usw = -inv(f1_x3*b)*(M*sat + nmn*s);
ut = ueq + usw;
sys(1) = ut;

```

**3. S function of plant: chap8\_1plant.m**

```

function [sys,x0,str,ts] = s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
 sys = mdlDerivatives(t,x,u);
case 3,
 sys = mdlOutputs(t,x,u);
case {2, 4, 9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

```

```

sys=simsizes(sizes);
x0=[0.15;0;0;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
l=10;
g=9.8;
f1=g*sin(x(1))/l+x(3);
sys(1)=x(2);
sys(2)=f1;
sys(3)=x(4);
sys(4)=ut-10*sin(t);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);

```

**4. Plot program: chap8\_1plot.m**

```

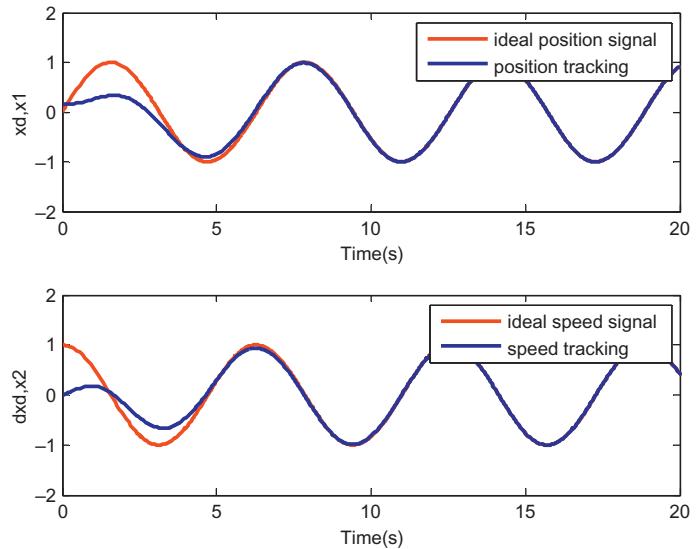
close all;

figure(1);
subplot(411);
plot(t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('x1');
subplot(412);
plot(t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('x2');
subplot(413);
plot(t,x(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x3');
subplot(414);
plot(t,x(:,4),'r','linewidth',2);
xlabel('time(s)');ylabel('x4');

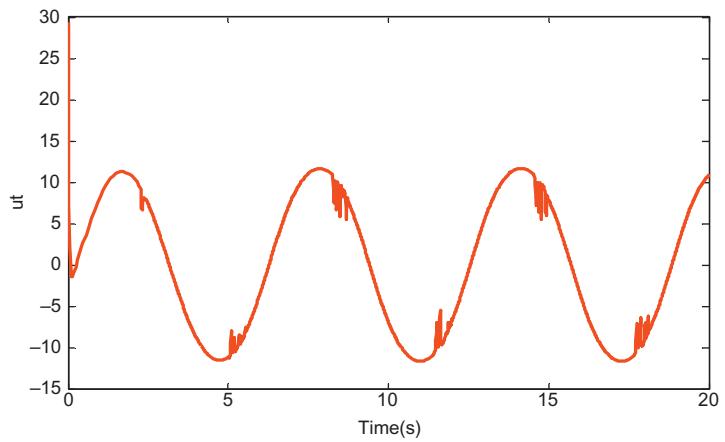
figure(2);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('ut');

```

Consider position tracking and speed tracking, let ideal position signal as  $x_1 \rightarrow x_d$  and  $x_2 \rightarrow \dot{x}_d$ . The initial system states are set as  $[0.15 \ 0 \ 0 \ 0]$ . Use controller (8.6), design  $M$  from Eq. (8.7), and



■ FIGURE 8.3 Position and speed tracking.

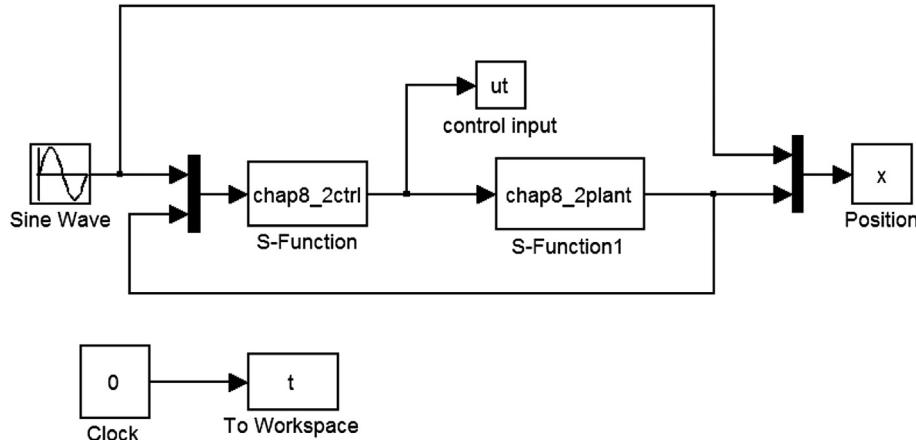


■ FIGURE 8.4 Control input.

design  $u_{eq}$  from Eq. (8.9), choose  $\beta_3 = 1.1$ ,  $c_1 = 27$ ,  $c_2 = 27$ ,  $c_3 = 9$ ,  $\rho = 1.0$ ,  $\lambda = 1.0$ , and  $D = 10$ , adapt the saturation function instead of the switch function, and set  $\Delta = 0.05$ . Simulation results are shown in Figs. 8.3 and 8.4.

Simulation Programs:

**1. Main program: chap8\_2sim.mdl**



**2. S function of controller: chap8\_2ctrl.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)

```

```

xd = sin(t);
dxd = cos(t);
ddxd = -sin(t);
dddx = -cos(t);
ddddx = sin(t);
x1 = u(2);
x2 = u(3);
x3 = u(4);
x4 = u(5);
c1 = 27; c2 = 27; c3 = 9;

l = 10; g = 9.8;
f1 = g*sin(x1)/l + x3;

b = 1;
f2 = 0;
f1_x1 = g*cos(x1)/l;
f1_x2 = 0;
f1_x3 = 1;
beta3 = 1.0 + 0.1;
D = -g/l*sin(x1)*x2^2 + g/l*cos(x1)*f1;

ueq = - inv(f1_x3*b)*(c1*x2 + c2*f1 + c3*f1_x1*x2 +
c3*f1_x2*f1 + c3*f1_x3*x4 + D - c1*dxd - c2*ddxd -
c3*dddx - dddd);

e1 = x1 - xd;
e2 = x2 - dxd;
e3 = f1 - ddx;
e4 = f1_x1*x2 + f1_x2*f1 + f1_x3*x4 - dddx;
s = c1*e1 + c2*e2 + c3*e3 + e4;

d_up = 10;
rou = 1.0;
M = beta3*d_up + rou;
nmn = 1.0;

S = 2;
if S == 1
 sat = sign(s);
elseif S == 2 %Saturated function
 fai = 0.05;
 if abs(s) <= fai

```

```

 sat=s/fai;
 else
 sat=sign(s);
 end
end
usw=-inv(f1_x3*b)*(M*sat+nmn*s);
ut=ueq+usw;
sys(1)=ut;
3. S function of plant: chap8_2plant.m
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=4;
sizes.NumDiscStates=0;
sizes.NumOutputs=4;
sizes.NumInputs=1;
sizes.DirFeedthrough=0;
sizes.NumSampleTimes=0;
sys=simsizes(sizes);
x0=[0.15;0;0;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
l=10;
g=9.8;
f1=g*sin(x(1))/l+x(3);
sys(1)=x(2);
sys(2)=f1;
sys(3)=x(4);
sys(4)=ut-10*sin(t);

```

```

function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);

4. Plot program: chap8_2plot.m
close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r',t,x(:,2),'b','linewidth',2);
legend('ideal position signal','position tracking');
xlabel('time(s)');ylabel('xd,x1');
subplot(212);
plot(t,cos(t),'r',t,x(:,3),'b','linewidth',2);
legend('ideal speed signal','speed tracking');
xlabel('time(s)');ylabel('dxd,x2');

figure(2);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('ut');

```

#### 8.1.4.2 **Second example**

Consider a single pendulum cart system, the dynamic model is

$$\ddot{\theta} = \frac{m(m + M_c)gl}{(M_c + m)I + M_c ml^2} \theta - \frac{ml}{(M_c + m)I + M_c ml^2} v,$$

$$\ddot{x} = -\frac{m^2 gl^2}{(M_c + m)I + M_c ml^2} \theta + \frac{I + ml^2}{(M_c + m)I + M_c ml^2} v,$$

where  $I = \frac{1}{12}mL^2$ ,  $l = \frac{1}{2}L$ ,  $\theta$  denotes pendulum angle, and  $x$  denotes cart position,  $g = 9.8$  and  $v$  is the control input. The mass of cart is  $M_c = 1$ , the mass of pendulum is  $m = 0.10$  and the length of one half of the pendulum is  $l = 0.5$ .

The control goals are  $\theta \rightarrow 0$ ,  $\dot{\theta} \rightarrow 0$ ,  $x \rightarrow 0$  and  $\dot{x} \rightarrow 0$ . Define  $z_1 = x$ ,  $z_2 = \dot{x}$ ,  $\theta_1 = \theta$ ,  $\theta_2 = \dot{\theta}$ , then

$$\begin{aligned}\dot{z}_1 &= z_2 \\ \dot{z}_2 &= t_2 \theta_1 + t_4 v \\ \dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= t_1 \theta_1 + t_3 v,\end{aligned}$$

$$\text{where } t_1 = \frac{m(m+M_c)gl}{(M_c+m)I+M_cml^2}, \quad t_2 = -\frac{m^2gl^2}{(M_c+m)I+M_cml^2}, \quad t_3 = \\ -\frac{ml}{(M_c+m)I+M_cml^2}, \quad t_4 = \frac{I+ml^2}{(M_c+m)I+M_cml^2}.$$

Define

$$\begin{aligned} x_1 &= z_1 - \frac{t_4}{t_3}\theta_1 \\ x_2 &= z_2 - \frac{t_4}{t_3}\theta_2 \\ x_3 &= \theta_1 \\ x_4 &= \theta_2. \end{aligned}$$

Let  $t_1\theta_1 + t_3v = u$ , then

$$v = \frac{u - t_1\theta_1}{t_3} \quad (8.10)$$

and

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \dot{z}_2 - \frac{t_4}{t_3}\dot{\theta}_2 = t_2\theta_1 + t_4v - \frac{t_4}{t_3}(t_1\theta_1 + t_3v) \\ &= \left(t_2 - \frac{t_4}{t_3}t_1\right)\theta_1 = \left(t_2 - \frac{t_1t_4}{t_3}\right)x_3 = Tx_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u, \end{aligned}$$

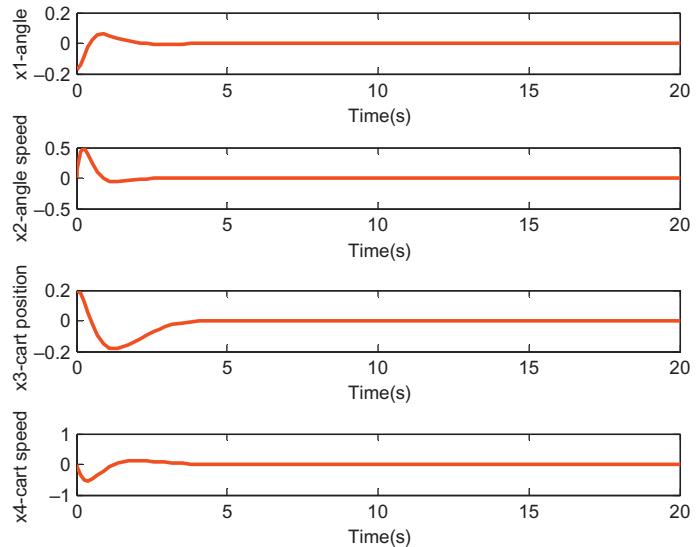
where  $T = t_2 - \frac{t_1t_4}{t_3}$ .

Define  $f_1 = Tx_3$ , and

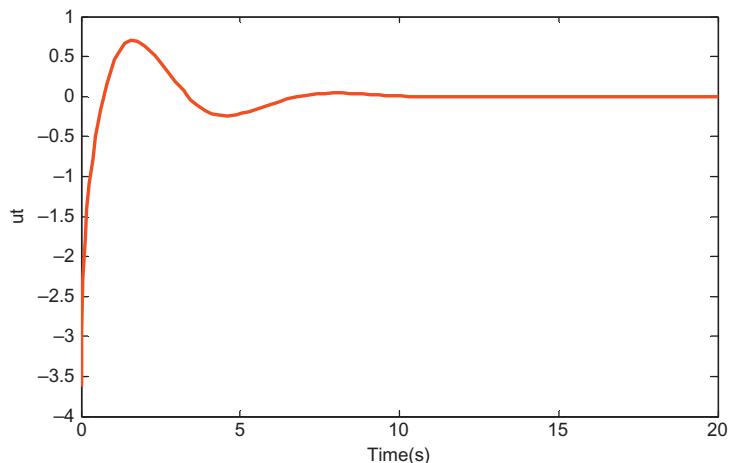
$$\begin{aligned} e_1 &= x_1 \\ e_2 &= x_2 \\ e_3 &= f_1 \\ e_4 &= \frac{\partial f_1}{\partial x_1}x_2 + \frac{\partial f_1}{\partial x_2}f_1 + \frac{\partial f_1}{\partial x_3}x_4 = Tx_4 \end{aligned}$$

Consider control goals as  $x_i \rightarrow 0, i = 1, 2, 3, 4$ . From  $f_1 = Tx_3$ , we have  $\frac{\partial f_1}{\partial x_1} = 0$ ,  $\frac{\partial f_1}{\partial x_2} = 0$ ,  $\frac{\partial f_1}{\partial x_3} = T$ , and  $\frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_1}x_2 \right] = 0$ ,  $\frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_2}f_1 \right] = 0$ ,  $\frac{d}{dt} \left[ \frac{\partial f_1}{\partial x_3}x_4 \right] = 0$ . The initial system states are set as  $\theta(0) = -10^\circ$ ,  $\dot{\theta}(0) = 0$  and  $x(0) = 0.20$ ,  $\dot{x}(0) = 0$ .

Using controller (8.6), design  $M$  from Eq. (8.7). Consider  $d = 0$ ; we choose  $M = \rho = 1.0$ , design  $u_{\text{eq}}$  from Eq. (8.9), choose  $c_1 = 27$ ,  $c_2 = 27$ ,  $c_3 = 9$ ,  $\rho = 1.0$  and  $\lambda = 1.0$ , design  $v$  from Eq. (8.10), adapt the saturation function instead of the switch function, set  $\Delta = 0.05$ , simulation results are shown in Figs. 8.5 and 8.6.



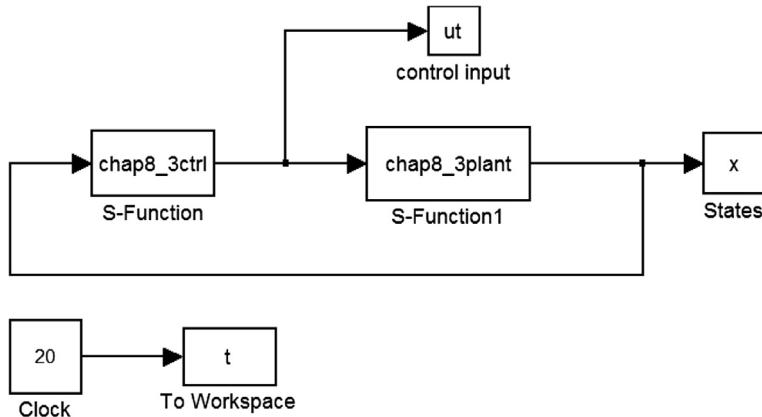
■ FIGURE 8.5 States response of cart pendulum system.



■ FIGURE 8.6 Control input.

Simulation Programs:

**1. Main program: chap8\_3sim.mdl**



**2. S function of controller: chap8\_3ctrl.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
c1=27;c2=27;c3=9;

```

```

g = 9.8; Mc = 1.0; m = 0.1; L = 0.5;
I = 1/12*m*L^2;
l = 1/2*L;
t1 = m*(Mc + m)*g*l / [(Mc + m)*I + Mc*m*l^2];
t2 = -m^2*g*l^2 / [(m + Mc)*I + Mc*m*l^2];
t3 = -m*l / [(Mc + m)*I + Mc*m*l^2];
t4 = (I + m*l^2) / [(m + Mc)*I + Mc*m*l^2];

th1 = u(1); %th
th2 = u(2); %dth
z1 = u(3); %x
z2 = u(4); %dx

x1 = z1 - t4/t3*th1;
x2 = z2 - t4/t3*th2;
x3 = th1;
x4 = th2;

T = t2 - t1*t4/t3;

b = 1;
f1 = T*x3;
f2 = 0;
f1_x1 = 0;
f1_x2 = 0;
f1_x3 = T;

ueq = -inv(f1_x3*b)*(c1*x2 + c2*f1 + c3*f1_x1*x2 + c3*f1_x2
 *f1 + c3*f1_x3*x4);
e1 = x1;
e2 = x2;
e3 = f1;
e4 = T*x4;
s = c1*e1 + c2*e2 + c3*e3 + e4;

rou = 1.0;
M = rou;
nmn = 1.0;

S = 2;
if S == 1
 sat = sign(s);
elseif S == 2 %Saturated function
 fai = 0.1;

```

```

if abs(s) <= fai
 sat = s/fai;
else
 sat = sign(s);
end
end
usw = - inv(f1_x3*b)*(M*sat + nmn*s);
ut = ueq + usw;

v = (ut - t1*th1)/t3;

sys(1) = v;
3. S function of plant: chap8_3plant.m
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [-10/57.3,0,0.20,0];
str = [];
ts = [0 0];
function sys=mdlDerivatives(t,x,u)
%Single Link Inverted Pendulum Parameters
g=9.8;Mc=1.0;m=0.1;L=0.5;

I=1/12*m*L^2;
l=1/2*L;

```

```
t1 = m*(Mc + m)*g*l / [(Mc + m)*I + Mc*m*l^2];
t2 = -m^2*g*l^2 / [(m + Mc)*I + Mc*m*l^2];
t3 = -m*l / [(Mc + m)*I + Mc*m*l^2];
t4 = (I + m*l^2) / [(m + Mc)*I + Mc*m*l^2];
```

```
A = [0,1,0,0;
 t1,0,0,0;
 0,0,0,1;
 t2,0,0,0];
B = [0;t3;0;t4];
C = [1,0,0,0;
 0,0,1,0];
v = u(1);
sys(1) = x(2);
sys(2) = t1*x(1) + t3*v; %ddth
sys(3) = x(4);
sys(4) = t2*x(1) + t4*v; %ddx
function sys = mdlOutputs(t,x,u)
sys(1) = x(1); %th
sys(2) = x(2);
sys(3) = x(3); %x
sys(4) = x(4);
```

**4. Plot program: chap8\_3plot.m**

```
close all;
```

```
figure(1);
subplot(411);
plot(t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('x1-angle');
subplot(412);
plot(t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('x2-angle speed');
subplot(413);
plot(t,x(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x3-cart position');
subplot(414);
plot(t,x(:,4),'r','linewidth',2);
xlabel('time(s)');ylabel('x4-cart speed');

figure(2);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('ut');
```

## 8.2 SLIDING MODE CONTROL BASED ON HURWITZ FOR AN UNDERACTUATED SYSTEM

In this section, we will consider how to design the sliding mode control based on Hurwitz for two kinds of underactuated systems.

### 8.2.1 Sliding mode control based on Hurwitz for a simple underactuated system

#### 8.2.1.1 System description

Consider the same **underactuated** system of [section 8.1.4.1](#) as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= g \sin x_1/l + x_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u - d,\end{aligned}\tag{8.11}$$

where  $d$  is disturbance,  $|d| \leq D$ .

The control goals are  $x_i \rightarrow 0, i = 1, 2, 3, 4$  as  $t \rightarrow \infty$ .

Different from 8.1.1, in this section, we introduce a sliding mode controller design by Hurwitz.

#### 8.2.1.2 Sliding mode controller design

Define the sliding mode function as

$$s = c_1x_1 + c_2x_2 + c_3x_3 + x_4,$$

where  $c_1, c_2$ , and  $c_3$  will be designed based on Hurwitz.

Then we have

$$\dot{s} = c_1\dot{x}_1 + c_2\dot{x}_2 + c_3\dot{x}_3 + \dot{x}_4 = c_1x_2 + c_2(g \sin x_1/l + x_3) + c_3x_4 + u - d.$$

Design the controller as

$$u = -c_1x_2 - c_2(g \sin x_1/l + x_3) - c_3x_4 - ks - \eta \text{sgn}(s),\tag{8.12}$$

where  $k > 0, \eta > D$ .

Substituting [Eq. \(8.12\)](#) into  $\dot{s}$ , we have  $\dot{s} = -ks - \eta \text{sgn}(s) - d$ , then  $s\dot{s} = -ks^2 - \eta|s| - sd \leq 0$ .

### 8.2.1.3 Hurwitz stability analysis

The term  $ss \leq 0$  indicates that there exists  $t_s$ , when  $t \geq t_s$ ,  $s = 0$ , i.e.,

$$s = c_1x_1 + c_2x_2 + c_3x_3 + x_4 = 0.$$

Then

$$x_4 = -c_1x_1 - c_2x_2 - c_3x_3. \quad (8.13)$$

The system equilibrium point is  $x_i \rightarrow 0, i = 1, 2, 3, 4$ , at the equilibrium point,  $\sin x_1 = x_1 + \varepsilon_1 x_1$ , then we have

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{g}{l}x_1 + x_3 + \varepsilon_1 \frac{g}{l}x_1 \\ \dot{x}_3 &= -c_1x_1 - c_2x_2 - c_3x_3,\end{aligned}$$

where  $\varepsilon_1$  is error caused by linearization.

Then we obtain a state equation as

$$\dot{\mathbf{x}} = \mathbf{Ax} + \varepsilon\mathbf{x}, \quad (8.14)$$

$$\text{where } \mathbf{x} = [x_1 \ x_2 \ x_3]^T, \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ g/l & 0 & 1 \\ -c_1 & -c_2 & -c_3 \end{bmatrix}, \varepsilon = \begin{bmatrix} 0 & 0 & 0 \\ \varepsilon_1 \frac{g}{l} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Since  $\varepsilon$  is very small value, if we design  $\mathbf{A}$  as Hurwitz, the stability of  $\dot{\mathbf{x}} = \mathbf{Ax} + \varepsilon\mathbf{x}$  can be guaranteed, and as  $t \rightarrow \infty$   $\mathbf{x} \rightarrow 0$ , i.e.,  $[x_1 \ x_2 \ x_3] \rightarrow 0$ .

$$\text{From } |\mathbf{A} - \lambda\mathbf{I}| = 0, \text{ we have } \begin{vmatrix} -\lambda & 1 & 0 \\ g/l & -\lambda & 1 \\ -c_1 & -c_2 & -c_3 - \lambda \end{vmatrix} = 0, \text{ then}$$

$$-\lambda^2(\lambda + c_3) - c_1 - c_2\lambda + (\lambda + c_3)\frac{g}{l} = 0$$

and

$$-\lambda^3 - c_3\lambda^2 + \left(-c_2 + \frac{g}{l}\right)\lambda - c_1 + c_3\frac{g}{l} = 0$$

i.e.,

$$\lambda^3 + c_3\lambda^2 + \left(c_2 - \frac{g}{l}\right)\lambda + c_1 - c_3\frac{g}{l} = 0$$

From  $(\lambda+3)^3 = 0$ , we have  $\lambda^3 + 9\lambda^2 + 27\lambda + 27 = 0$ , then we can get

$$\begin{cases} c_3 = 9 \\ c_2 = \frac{g}{l} + 27 \\ c_1 = c_3 \frac{g}{l} + 27. \end{cases} \quad (8.15)$$

Moreover, consider  $s \rightarrow 0$ , we have  $x_4 \rightarrow 0$ .

Compared with sliding mode controller design of the first example in section 8.1, the design method in this section is easy.

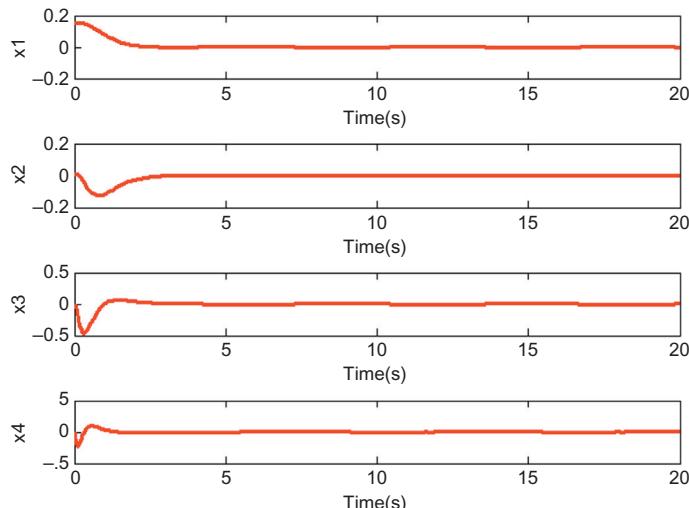
#### 8.2.1.4 Simulation example

Consider a system as

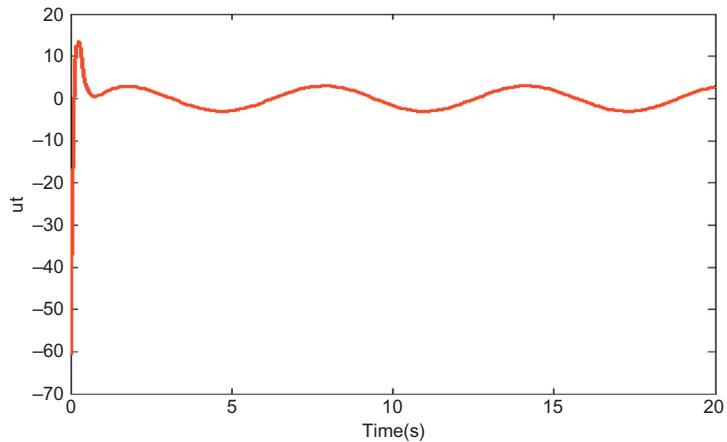
$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= g \sin x_1/l + x_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u - d \end{aligned}$$

The initial states are set as  $[0.15 \ 0 \ 0 \ 0]$ , and set  $d = 10 \sin t$ ,  $l = 10$ .

Using controller (8.12), and setting  $\eta = D + 0.10$ ,  $k = 10$ , adapt the saturation function instead of the switch function, and let  $\Delta = 0.10$ . Simulation results are shown in Figs. 8.7 and 8.8.



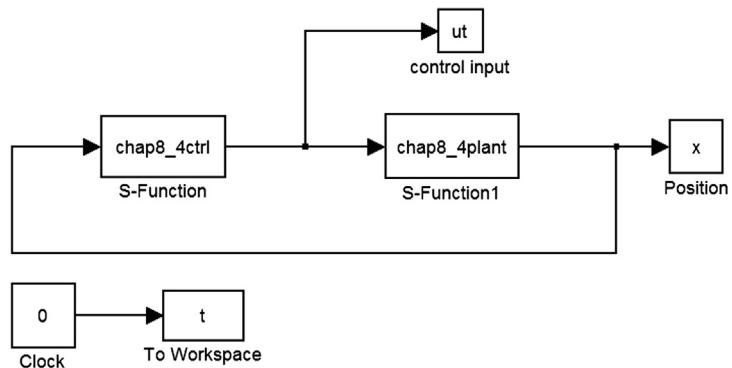
■ FIGURE 8.7 System states response.



■ FIGURE 8.8 Control input.

Simulation Programs:

1. Main program: chap8\_4sim.mdl



2. S function of controller: chap8\_4ctrl.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
 sys = mdlOutputs(t,x,u);
case {2,4,9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
```

```
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
l=10;g=9.8;

x1=u(1);x2=u(2);x3=u(3);x4=u(4);

c3=9;
c2=g/l+27;
c1=c3*g/l+27;
s=c1*x1+c2*x2+c3*x3+x4;

D=3.0;
xite=D+0.10;

S=2;
if S==1
 sat=sign(s);
elseif S==2 %Saturated function
 fai=0.10;
 if abs(s)<=fai
 sat=s/fai;
 else
 sat=sign(s);
 end
end

k=10;
ut=-c1*x2-c2*(g/l*sin(x1)+x3)-c3*x4-k*s-xite*sat;
sys(1)=ut;
```

3. S function of plant: chap8\_4plant.m

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0=[0.15;0;0;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
l=10;
g=9.8;
f1=g*sin(x(1))/l+x(3);
sys(1)=x(2);
sys(2)=f1;
sys(3)=x(4);
sys(4)=ut-3*sin(t);
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);
```

**4. Plot program: chap8\_4plot.m**

```

close all;

figure(1);
subplot(411);
plot(t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('x1');
subplot(412);
plot(t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('x2');
subplot(413);
plot(t,x(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x3');
subplot(414);
plot(t,x(:,4),'r','linewidth',2);
xlabel('time(s)');ylabel('x4');
figure(2);

plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('ut');

```

## 8.2.2 Sliding mode control based on Hurwitz for an inverted pendulum

### 8.2.2.1 System description

An inverted pendulum dynamic model is expressed as

$$\begin{bmatrix} M_c + m & ml \cos \theta \\ ml \cos \theta & ml^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} ml\dot{\theta}^2 \sin \theta \\ mgl \sin \theta \end{bmatrix} + \begin{bmatrix} u \\ 0 \end{bmatrix}, \quad (8.16)$$

i.e.,

$$(M_c + m)\ddot{x} + ml \ddot{\theta} \cos \theta = ml \dot{\theta}^2 \sin \theta + u$$

$$ml \cos \theta \ddot{x} + ml^2 \ddot{\theta} = mgl \sin \theta,$$

where  $M_c$  is mass of cart,  $m$  is mass of pendulum,  $x$  is position of cart, and  $\theta$  is pendulum angle,  $l$  denotes the pendulum length.

The control goals are set as  $x \rightarrow 0$ ,  $\dot{x} \rightarrow 0$ ,  $\theta \rightarrow 0$  and  $\dot{\theta} \rightarrow 0$  as  $t \rightarrow \infty$ .

### 8.2.2.2 Sliding mode controller design

Eq. (8.16) can be expressed as

$$\begin{bmatrix} M_{aa} & M_{au} \\ M_{au} & M_{uu} \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} f_a + u \\ f_u \end{bmatrix}, \quad (8.17)$$

where  $M_{aa} = M_c + m$ ,  $M_{au} = ml \cos \theta$ ,  $M_{uu} = ml^2$ ,  $f_a = ml \sin \theta \cdot \dot{\theta}^2$ ,  $f_u = mg l \sin \theta$ .

Define  $M = \begin{bmatrix} M_{aa} & M_{au} \\ M_{au} & M_{uu} \end{bmatrix}$ , then Eq. (8.17) becomes

$$\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = M(\theta)^{-1} \begin{bmatrix} f_a + u \\ f_u \end{bmatrix} = \frac{\begin{bmatrix} M_{uu} & -M_{au} \\ -M_{au} & M_{aa} \end{bmatrix}}{M_{aa}M_{uu} - M_{au}M_{au}} \begin{bmatrix} f_a + u \\ f_u \end{bmatrix}$$

$$= \frac{\begin{bmatrix} M_{uu}(f_a + u) - M_{au}f_u \\ -M_{au}(f_a + u) + M_{aa}f_u \end{bmatrix}}{M_{aa}M_{uu} - M_{au}M_{au}} = \frac{\begin{bmatrix} M_{uu}f_a - M_{au}f_u \\ -M_{au}f_a + M_{aa}f_u \end{bmatrix}}{M_{aa}M_{uu} - M_{au}M_{au}} + \frac{\begin{bmatrix} M_{uu}u \\ -M_{au}u \end{bmatrix}}{M_{aa}M_{uu} - M_{au}M_{au}}$$

i.e.,

$$\begin{aligned} \ddot{x} &= \frac{M_{uu}f_a - M_{au}f_u}{M_{aa}M_{uu} - M_{au}M_{au}} + \frac{M_{uu}}{M_{aa}M_{uu} - M_{au}M_{au}}u \\ \ddot{\theta} &= \frac{-M_{au}f_a + M_{aa}f_u}{M_{aa}M_{uu} - M_{au}M_{au}} + \frac{-M_{au}}{M_{aa}M_{uu} - M_{au}M_{au}}u. \end{aligned}$$

The above equation can be written as

$$\begin{aligned} \ddot{x} &= f_1 + b_1 u \\ \ddot{\theta} &= f_2 + b_2 u, \end{aligned} \quad (8.18)$$

where  $f_1 = \frac{M_{uu}f_a - M_{au}f_u}{M_{aa}M_{uu} - M_{au}M_{au}}$ ,  $b_1 = \frac{M_{uu}}{M_{aa}M_{uu} - M_{au}M_{au}}$ ,  $f_2 = \frac{-M_{au}f_a + M_{aa}f_u}{M_{aa}M_{uu} - M_{au}M_{au}}$   
and  $b_2 = \frac{-M_{au}}{M_{aa}M_{uu} - M_{au}M_{au}}$ .

For Eq. (8.18), define the sliding mode function as

$$s = \dot{x} + c_1 x + c_2 \dot{\theta} + c_3 \theta,$$

where  $c_1$ ,  $c_2$ , and  $c_3$  are designed by Hurwitz.

Then

$$\begin{aligned} \dot{s} &= \ddot{x} + c_2 \ddot{\theta} + c_1 \dot{x} + c_3 \dot{\theta} = f_1 + b_1 u + c_2(f_2 + b_2 u) + c_1 \dot{x} + c_3 \dot{\theta} \\ &= f_1 + c_2 f_2 + (b_1 + c_2 b_2)u + c_1 \dot{x} + c_3 \dot{\theta}. \end{aligned}$$

Design the controller as

$$u = -\frac{1}{b_1 + c_2 b_2} (f_1 + c_2 f_2 + c_1 \dot{x} + c_3 \dot{\theta} + \eta \operatorname{sgn}(s)), \quad (8.19)$$

where  $\eta > 0$ .

Substituting Eq. (8.19) into  $\dot{s}$ , we have

$$\dot{s} = -\eta \operatorname{sgn}(s), \text{ then we } s\dot{s} = -\eta s \cdot \operatorname{sgn}(s) = -\eta |s| \leq 0.$$

### 8.2.2.3 Hurwitz stability analysis

The term  $s\dot{s} \leq 0$  indicates that there exists  $t_s$ , when  $t \geq t_s$ , we have

$$s = \dot{x} + c_1 x + c_2 \dot{\theta} + c_3 \theta = 0,$$

i.e.,

$$\dot{x} = -c_1 x - c_2 \dot{\theta} - c_3 \theta.$$

Substituting Eq. (8.19) into Eq. (8.18), we obtain

$$\begin{aligned} \ddot{\theta} &= f_2 + b_2 \frac{-1}{b_1 + c_2 b_2} (f_1 + c_2 f_2 + c_1 \dot{x} + c_3 \dot{\theta}) \\ &= \frac{1}{b_1 + c_2 b_2} (f_2 b_1 + c_2 f_2 b_2 - f_1 b_2 - c_2 f_2 b_2 - b_2 (c_1 \dot{x} + c_3 \dot{\theta})) \\ &= \frac{1}{b_1 + c_2 b_2} (f_2 b_1 - f_1 b_2 - b_2 (c_1 \dot{x} + c_3 \dot{\theta})) \\ &= \frac{f_2 b_1 - f_1 b_2}{b_1 + c_2 b_2} - \frac{b_2}{b_1 + c_2 b_2} (c_1 \dot{x} + c_3 \dot{\theta}) \\ &= \frac{1}{M_{uu} - c_2 M_{au}} (f_u + M_{au} (c_1 \dot{x} + c_3 \dot{\theta})), \end{aligned}$$

where

$$\begin{aligned} f_2 M_{uu} + f_1 M_{au} &= \frac{-M_{au} f_a + M_{aa} f_u}{M_{aa} M_{uu} - M_{au} M_{au}} M_{uu} + \frac{M_{uu} f_a - M_{au} f_u}{M_{aa} M_{uu} - M_{au} M_{au}} M_{au} = \\ \frac{M_{aa} f_u M_{uu} - M_{au} f_u M_{au}}{M_{aa} M_{uu} - M_{au} M_{au}} &= f_u, \quad \frac{f_2 b_1 - f_1 b_2}{b_1 + c_2 b_2} = \frac{f_2 M_{uu} + f_1 M_{au}}{M_{uu} - c_2 M_{au}} = \frac{f_u}{M_{uu} - c_2 M_{au}} \\ \text{and } \frac{b_2}{b_1 + c_2 b_2} &= \frac{-M_{au}}{M_{uu} - c_2 M_{au}}. \end{aligned}$$

Adding  $M_{uu}$  and  $M_{au}$  into the above, we have

$$\begin{aligned}\ddot{\theta} &= \frac{1}{ml^2 - c_2 ml \cos \theta} (mgl \sin \theta + ml \cos \theta (c_1 \dot{x} + c_3 \dot{\theta})) \\ &= \frac{1}{l - c_2 \cos \theta} (g \sin \theta + \cos \theta (c_1 (-c_1 x - c_2 \dot{\theta} - c_3 \theta) + c_3 \dot{\theta})) \\ &= \frac{g \sin \theta + \cos \theta (-c_1 c_3 \theta + (-c_1 c_2 + c_3) \dot{\theta} - c_1 c_1 x)}{l - c_2 \cos \theta}.\end{aligned}$$

Define  $y_1 = \theta, y_2 = \dot{\theta}, y_3 = x$ , then

$$\begin{aligned}\dot{y}_1 &= y_2 \\ \dot{y}_2 &= \frac{g \sin y_1 + \cos y_1 (-c_1 c_3 y_1 + (-c_1 c_2 + c_3) y_2 - c_1 c_1 y_3)}{l - c_2 \cos \theta} \\ \dot{y}_3 &= -c_3 y_1 - c_2 y_2 - c_1 y_3.\end{aligned}\tag{8.20}$$

The system equilibrium point is  $\theta = 0, \dot{\theta} = 0, x = 0, \dot{x} = 0$ , i.e.,  $y_1 = 0, y_2 = 0, y_3 = 0$ . At the equilibrium point, let  $\sin \theta \approx \theta, \cos \theta \approx 1$ , then Eq. (8.20) becomes

$$\begin{aligned}\dot{y}_1 &= y_2 \\ \dot{y}_2 &= \frac{gy_1 - c_1 c_3 y_1 + (-c_2 c_1 + c_3) y_2 - c_1 c_1 y_3}{l - c_2} + \varepsilon_1 y_1 + \varepsilon_2 y_2 + \varepsilon_3 y_3, \\ \dot{y}_3 &= -c_3 y_1 - c_2 y_2 - c_1 y_3\end{aligned}$$

where  $\varepsilon_i$  is error caused by linearization,  $i = 1, 2, 3$ .

Then we obtain

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \boldsymbol{\varepsilon}\mathbf{y},\tag{8.21}$$

$$\text{where } \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ A_{21} & A_{22} & A_{23} \\ -c_3 & -c_2 & -c_1 \end{bmatrix}, \boldsymbol{\varepsilon} = \begin{bmatrix} 0 & 0 & 0 \\ \varepsilon_1 & \varepsilon_2 & \varepsilon_3 \\ 0 & 0 & 0 \end{bmatrix}.$$

Since  $\varepsilon_i$  is a very small value, if we design  $\mathbf{A}$  as Hurwitz, the stability of  $\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} + \boldsymbol{\varepsilon}\mathbf{y}$  can be guaranteed, then if  $t \rightarrow \infty$ , we have  $\mathbf{y} \rightarrow 0$ , i.e.,  $[y_1 \ y_2 \ y_3] \rightarrow 0$ , or  $x \rightarrow 0, \theta \rightarrow 0$  and  $\dot{\theta} \rightarrow 0$ . Moreover, consider  $s \rightarrow 0$ ; we have  $\dot{x} \rightarrow 0$ .

Let  $c_2 \neq l$ ,  $A_{21} = \frac{g - c_1 c_3}{l - c_2}$ ,  $A_{22} = \frac{-c_2 c_1 + c_3}{l - c_2}$  and  $A_{23} = -\frac{c_1^2}{l - c_2}$ , from

$|A - \lambda I| = 0$ , we have  $\begin{vmatrix} -\lambda & 1 & 0 \\ A_{21} & A_{22} - \lambda & A_{23} \\ -c_3 & -c_2 & -c_1 - \lambda \end{vmatrix} = 0$ , then

$$-\lambda(\lambda - A_{22})(\lambda + c_1) - c_3 A_{23} - c_2 \lambda A_{23} - (-c_1 - \lambda) A_{21} = 0,$$

i.e.,

$$\lambda^3 - (A_{22} - c_1)\lambda^2 + (-c_1A_{22} - A_{21} + c_2A_{23})\lambda - c_1A_{21} + c_3A_{23} = 0.$$

Compared with the above equation, from  $(\lambda + 1)(\lambda + 2)(\lambda + 3) = 1$ , i.e.,  $\lambda^3 + 6\lambda^2 + 11\lambda + 6 = 0$ , we can get equation sets as

$$\begin{cases} -(A_{22} - c_1) = 6 \\ -c_1A_{22} - A_{21} + c_2A_{23} = 11 \\ -c_1A_{21} + c_3A_{23} = 6. \end{cases}$$

Since

$$\begin{aligned} A_{22} - c_1 &= \frac{-c_2c_1 + c_3 - lc_1 + c_2c_1}{l - c_2} = \frac{c_3 - lc_1}{l - c_2} \\ -c_1A_{22} - A_{21} + c_2A_{23} &= -c_1 \frac{-c_2c_1 + c_3}{l - c_2} - \frac{g - c_1c_3}{l - c_2} - c_2 \frac{c_1^2}{l - c_2} \\ &= \frac{c_2c_1^2 - c_1c_3 - g + c_1c_3 - c_2c_1^2}{l - c_2} = -\frac{g}{l - c_2} \\ -c_1A_{21} + c_3A_{23} &= -c_1 \frac{g - c_1c_3}{l - c_2} - \frac{c_3c_1^2}{l - c_2} = \frac{-c_1g + c_1^2c_3 - c_3c_1^2}{l - c_2} = -\frac{c_1g}{l - c_2}, \end{aligned}$$

i.e.,

$$\begin{cases} -\frac{c_3 - lc_1}{l - c_2} = 6 \\ -\frac{g}{l - c_2} = 11 \\ -\frac{c_1g}{l - c_2} = 6. \end{cases}$$

Then

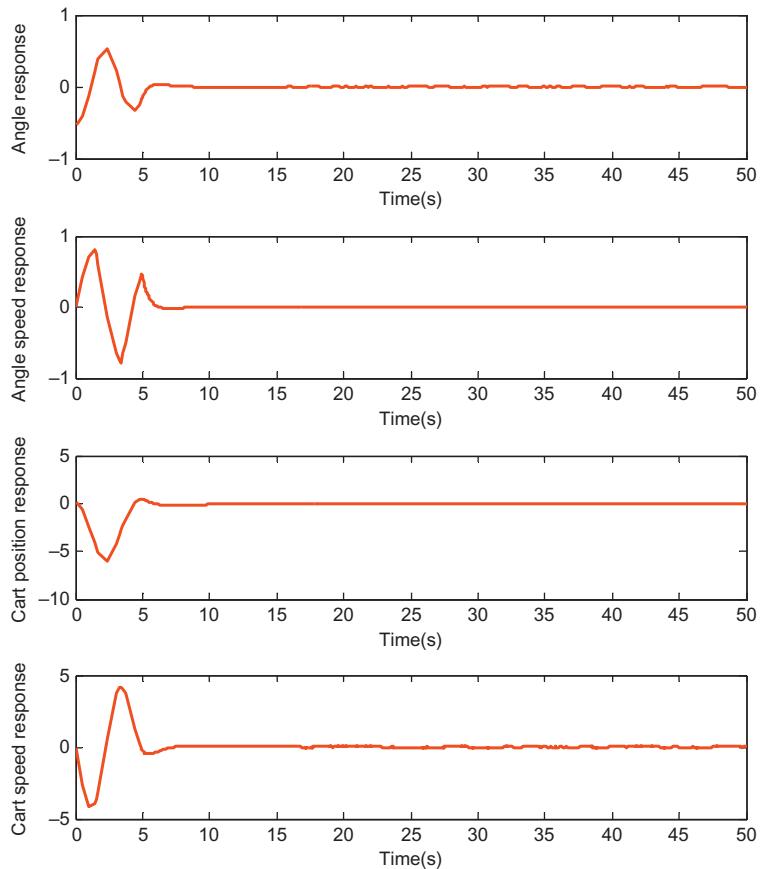
$$\begin{cases} lc_1 - c_3 = 6(l - c_2) \\ 11(l - c_2) = -g \\ c_1g = -6(l - c_2). \end{cases}$$

The solution of the above equations are

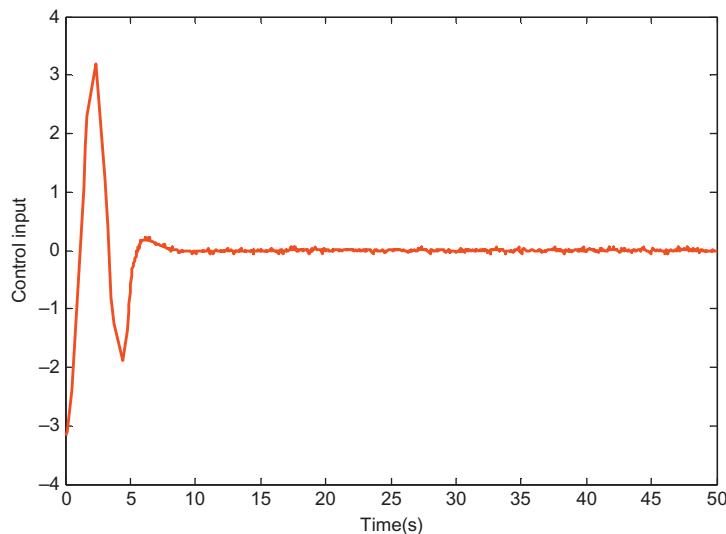
$$\begin{cases} c_2 = l + \frac{g}{11} \\ c_1 = \frac{6}{g}(c_2 - l) \\ c_3 = lc_1 + 6(c_2 - l). \end{cases} \quad (8.22)$$

### 8.2.2.4 Simulation example

Consider system (8.16); we choose  $m_c = 0.4$ ,  $m = 0.14$ ,  $l = 0.215$  and  $g = 9.8$ , and set initial system states as  $\begin{bmatrix} -\frac{\pi}{3} & 0 & 0.5 & 0 \end{bmatrix}$ , use Eq. (8.22) to get  $c_1$ ,  $c_2$ , and  $c_3$ . In controller (8.19), set  $\eta = 50$ , use the saturation function instead of the switch function, set  $\Delta = 0.10$ . The simulation results are shown in Figs. 8.9 and 8.10.



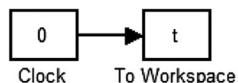
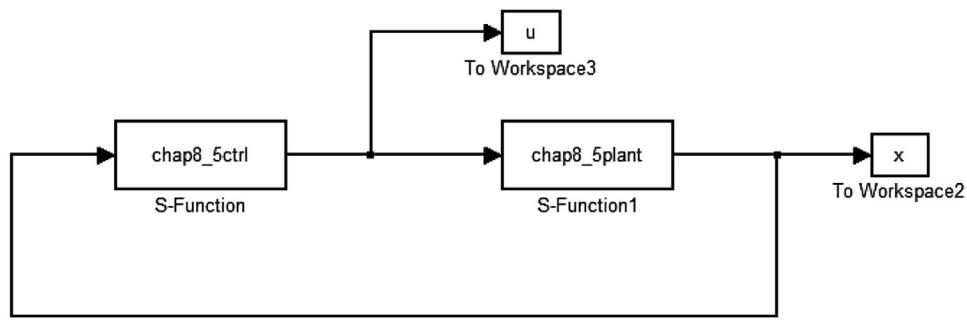
■ FIGURE 8.9 System states response.



■ FIGURE 8.10 Control input.

Simulation Programs:

1. Main program: chap8\_5sim.mdl



2. S function of controller: chap8\_5ctrl.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,

```

```

[sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=0;
sizes.NumDiscStates=0;
sizes.NumOutputs=1;
sizes.NumInputs=4;
sizes.DirFeedthrough=1;
sizes.NumSampleTimes=1;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[0 0];
function sys=mdlOutputs(t,x,u)
x1=u(1);x2=u(2); %Pendulum angle
x3=u(3);x4=u(4); %Cart

g=9.8;mc=0.4;m=0.14;l=0.215;

c2=l+g/11;
c1=6/g*(c2-l);
c3=l*c2+6*(c2-l);

s=x4+c1*x3+c2*x2+c3*x1;

Maa=mc+m;
Mau=m*l*cos(x1);
Muu=m*l^2;
fa=m*l*sin(x1)*x2^2;
fu=m*g*l*sin(x1);

xite=3;
fai=0.10;
if abs(s)<=fai
 sat=s/fai;

```

```

else
 sat = sign(s);
end

b1 = Muu/(Maa*Muu-Mau^2);
b2 = -Mau/(Maa*Muu-Mau^2);

f1 = (Muu*fa-Mau*fu)/(Maa*Muu-Mau^2);
f2 = (-Mau*fa + Maa*fu)/(Maa*Muu-Mau^2);
ut = -1/(b1 + c2*b2)*(f1 + c2*f2 + c1*x4 + c3*x2 + xite*sat);
sys(1) = ut;

```

**3. S function of plant: chap8\_5plant.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [-pi/3,0,0.5,0];
str = [];
ts = [];
function sys=mdlDerivatives(t,x,u)
g=9.8;mc=0.4;m=0.14;l=0.215;
th=x(1);
dth=x(2);

```

```

M = [mc + m*m*l*cos(theta);
 m*l*cos(theta) m*l^2];
fa = m*l*sin(theta)*dtheta^2;
fu = m*g*l*sin(theta);
F = [fa; fu];

U = [u(1);0];

a = inv(M)*(F+U);
ddx = a(1);
ddtheta = a(2);

sys(1)=x(2);
sys(2)=ddtheta; %Pendulum angle
sys(3)=x(4);
sys(4)=ddx; %Cart
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);

```

**4. Plot program: chap8\_5plot.m**

```
close all;
```

```

figure(1);
subplot(411);
plot(t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Angle response');
subplot(412);
plot(t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('Angle speed response');
subplot(413);
plot(t,x(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('Cart position response');
subplot(414);
plot(t,x(:,4),'r','linewidth',2);
xlabel('time(s)');ylabel('Cart speed response');

figure(2);
plot(t,u(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input');

```

### 8.3 SLIDING MODE CONTROL FOR A SPECIAL UNDERACTUATED SYSTEM

#### 8.3.1 System description

Consider a special underactuated nonlinear system as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= g \sin x_1/l + x_3^3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u,\end{aligned}\tag{8.23}$$

where  $f(x_1, x_3) = g \sin x_1/l + x_3^3$ ,  $u$  is control input.

The control goals are  $x_i \rightarrow 0, i = 1, 2, 3, 4$  as  $t \rightarrow \infty$ .

If we design the sliding mode controller according to traditional method, define the error state equation as

$$\begin{aligned}e_1 &= x_1 \\ e_2 &= \dot{e}_1 = x_2 \\ e_3 &= \ddot{e}_1 = \dot{x}_2 = f(x_1, x_3) \\ e_4 &= \ddot{e}_1 = \dot{f} = \frac{\partial f}{\partial x_1} x_2 + \frac{\partial f}{\partial x_3} x_4.\end{aligned}\tag{8.24}$$

Design the sliding mode function as

$$s = c_1 e_1 + c_2 e_2 + c_3 e_3 + e_4,\tag{8.25}$$

where  $c_i > 0, i = 1, 2, 3$ .

From Eq. (8.25), we have

$$\begin{aligned}\dot{s} &= c_1 \dot{e}_1 + c_2 \dot{e}_2 + c_3 \dot{e}_3 + \dot{e}_4 \\ &= c_1 x_2 + c_2 f + c_3 \left( \frac{\partial f}{\partial x_1} x_2 + \frac{\partial f}{\partial x_3} x_4 \right) + \frac{d}{dt} \left[ \frac{\partial f}{\partial x_1} x_2 \right] + \frac{d}{dt} \left[ \frac{\partial f}{\partial x_3} \right] x_4 + \frac{\partial f}{\partial x_3} u.\end{aligned}\tag{8.26}$$

or the first example in Section 8.1,  $\frac{\partial f_1}{\partial x_3} = 1$ , in this section,  $\frac{\partial f}{\partial x_3} = 3x_3^2$  is invertible,  $u$  will not be realized. In the following, reference to [1], a new sliding mode controller design method is introduced.

#### 8.3.2 Sliding mode controller design

Consider  $\frac{\partial f}{\partial x_3}$  is invertible, sliding mode control cannot be designed.

To overcome this question, we define a new function,  $f_1(x_1, x_3)$ , as

$$f(x_1, x_3) = g \sin x_1/l + x_3^3 + x_3 - x_3 = f_1(x_1, x_3) - x_3,$$

where  $f_1(x_1, x_3) = g \sin x_1/l + x_3^3 + x_3$ .

In the sliding mode controller design,  $f_1(x_1, x_3)$  must be satisfied as three assumptions as follows:

Assumption 1:  $f_1(0, 0) \rightarrow 0$ ;

Assumption 2:  $\frac{\partial f_1}{\partial x_3}$  is invertible;

Assumption 3: if  $f_1(0, x_3) \rightarrow 0$ , then  $x_3 \rightarrow 0$ .

Then Eq. (8.23) becomes

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_1(x_1, x_3) - x_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u.\end{aligned}\tag{8.27}$$

And we can get error state equation as

$$\begin{aligned}e_1 &= x_1 \\ e_2 &= \dot{e}_1 = x_2 \\ e_3 &= f_1(x_1, x_3) \\ e_4 &= \dot{e}_3 = \dot{f}_1(x_1, x_3) = \frac{\partial f_1}{\partial x_1}x_2 + \frac{\partial f_1}{\partial x_3}x_4.\end{aligned}\tag{8.28}$$

In the above equation, to prevent singularity in the controller design, we do not let  $e_3 = \dot{e}_2$ .

Design the sliding mode function as

$$s = c_1 e_1 + c_2 e_2 + c_3 e_3 + e_4,\tag{8.29}$$

where  $c_i > 0$ ,  $i = 1, 2, 3$ .

Then

$$\begin{aligned}\dot{s} &= c_1 \dot{e}_1 + c_2 \dot{e}_2 + c_3 \dot{e}_3 + \dot{e}_4 \\ &= c_1 x_2 + c_2 (f_1 - x_3) + c_3 \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_3} x_4 \right) + \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_3} x_4 \right),\end{aligned}\tag{8.30}$$

$$\text{where } \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_3} x_4 \right) = \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_1} \right) x_2 + \frac{\partial f_1}{\partial x_1} (f_1 - x_3) + \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_3} \right) x_4 + \frac{\partial f_1}{\partial x_3} u.$$

Then

$$\begin{aligned}\dot{s} &= c_1 x_2 + c_2 (f_1 - x_3) + c_3 \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_3} x_4 \right) \\ &\quad + \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_1} \right) x_2 + \frac{\partial f_1}{\partial x_1} (f_1 - x_3) + \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_3} \right) x_4 + \frac{\partial f_1}{\partial x_3} u.\end{aligned}\tag{8.31}$$

Define  $M = c_1x_2 + c_2(f_1 - x_3) + c_3\left(\frac{\partial f_1}{\partial x_1}x_2 + \frac{\partial f_1}{\partial x_3}x_4\right) + \frac{d}{dt}\left(\frac{\partial f_1}{\partial x_1}\right)x_2 + \frac{\partial f_1}{\partial x_1}(f_1 - x_3) + \frac{d}{dt}\left(\frac{\partial f_1}{\partial x_3}\right)x_4$ ; the sliding mode controller can be designed as

$$u = \left[\frac{\partial f_1}{\partial x_3}\right]^{-1} \{-M - \eta \text{sgn}s - ks\}, \quad (8.32)$$

where  $\eta > 0$ ,  $k > 0$ .

Substituting Eq. (8.32) into Eq. (8.31), we have  $\dot{s} = -\eta \text{sgn}(s) - ks$ . Design the Lyapunov function as  $V = \frac{1}{2}s^2$ , then we have

$$\dot{V} = s\dot{s} = -\eta|s| - ks^2 \leq 0.$$

### 8.3.3 Convergence analysis

From Eqs. (8.27) and (8.28), we have  $\dot{e}_1 = e_2$ ,  $\dot{e}_2 = e_3 - x_3$  and  $\dot{e}_3 = e_4$ . From  $s\dot{s} \leq 0$ , under controller (8.32), the system can reach and thereafter stay on the manifold  $s = 0$  in finite time. On the manifold, we have  $s = 0$ , i.e.,  $e_4 = -c_1e_1 - c_2e_2 - c_3e_3$ .

Define  $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -c_1 & -c_2 & -c_3 \end{bmatrix}$ , and design  $A$  as Hurwitz, then

$$\begin{aligned} \dot{e}_1 &= e_2 \\ \dot{e}_2 &= e_3 - x_3 \\ \dot{e}_3 &= -c_1e_1 - c_2e_2 - c_3e_3. \end{aligned} \quad (8.33)$$

Define  $E = [e_1 \ e_2 \ e_3]^T$ ,  $d_2 = -x_3$ , from  $e_3 = f_1(x_1, x_3) = g \sin x_1/l + x_3^3 + x_3$ , we have  $d_2 = -x_3 = g \sin x_1/l + x_3^3 - e_3$ , then  $x_3(1 + x_3^2) = e_3 - g \sin x_1/l$ , consider  $\sin x_1 < x_1$ , we can obtain

$$|x_3| \leq |x_3|(1 + x_3^2) = |e_3 - g \sin x_1/l| \leq |e_3| + g|e_1|/l.$$

Then we obtain

$$|d_2| \leq (1 + g/l)||E||_2.$$

Define  $D = [0 \ d_2 \ 0]^T$ ,  $\gamma = 1 + g/l$  and  $||D||_2 \leq \gamma ||E||_2$ , from Eq. (8.33), we have

$$\dot{E} = AE + D. \quad (8.34)$$

To guarantee  $E = [e_1 \ e_2 \ e_3]^T \rightarrow 0$ ,  $A$  must be designed as Hurwitz.

Since  $|A - \lambda I| = \begin{vmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \\ -c_1 & -c_2 & -c_3 - \lambda \end{vmatrix} = \lambda^2(-c_3 - \lambda) - c_1 - c_2\lambda = -\lambda^3 - c_3\lambda^2 - c_2\lambda - c_1 = 0$ , from  $(\lambda + a)^3 = 0$ , we have  $\lambda^3 + 3a\lambda^2 + 3a^2\lambda + a^3 = 0$ , then we can design  $c_i$  as follows:

$$c_1 = a^3, c_2 = 3a^2, c_3 = 3a, a > 0. \quad (8.35)$$

Let  $Q = Q^T > 0$ , since  $A$  is Hurwitz, there exists the Lyapunov equation  $A^T P + PA = -Q, P = P^T > 0$ .

Design the Lyapunov function as  $V_1 = E^T PE$ , then

$$\begin{aligned} \dot{V}_1 &= \dot{E}^T PE + E^T P \dot{E} = (AE + D)^T PE + E^T P(AE + D) \\ &= E^T A^T PE + D^T PE + E^T PAE + E^T PD = E^T (A^T P + PA)E + D^T PE + E^T PD \\ &= -E^T QE + D^T PE + E^T PD \leq -\lambda_{\min}(Q) \|E\|_2^2 + 2\lambda_{\max}(P)\gamma \|E\|_2^2 \\ &= (-\lambda_{\min}(Q) + 2\lambda_{\max}(P)\gamma) \|E\|_2^2, \end{aligned}$$

where  $D^T PE + E^T PD \leq 2\lambda_{\max}(P)\gamma \|E\|_2^2$ ,  $\lambda_{\min}(Q)$  is the minimum eigenvalue of  $Q$  and  $\lambda_{\max}(P)$  is the maximum eigenvalue of  $P$ .

To realize  $\dot{V}_1 \leq 0$ , let  $\frac{\lambda_{\min}(Q)}{2\lambda_{\max}(P)} > \gamma$ . Choose  $Q = I_3$ , thus  $\lambda_{\min}(Q) = 1$ ,  $\lambda_{\max}(P) = \frac{1}{2\lambda_{\text{left}}(-A)}$ , and  $\frac{\lambda_{\min}(Q)}{2\lambda_{\max}(P)} > \gamma$  becomes  $\lambda_{\text{left}}(-A) > \gamma$ , which can be realized by  $A$  design.

If we design  $A$  as Hurwitz,  $E = [e_1 \ e_2 \ e_3]^T \rightarrow 0$  will be satisfied, and then we have  $e_1 \rightarrow 0, e_2 \rightarrow 0, e_3 \rightarrow 0$ , from  $s \rightarrow 0$ , we have  $e_4 \rightarrow 0$ . Then we can get  $x_1 \rightarrow 0, x_2 \rightarrow 0$ , Consider  $e_3 = f_1(0, x_3) \rightarrow 0$ , assumption 3 and Eq. (8.24), we can get  $x_3 \rightarrow 0, x_4 \rightarrow 0$ .

### 8.3.4 Simulation example

Consider a system as

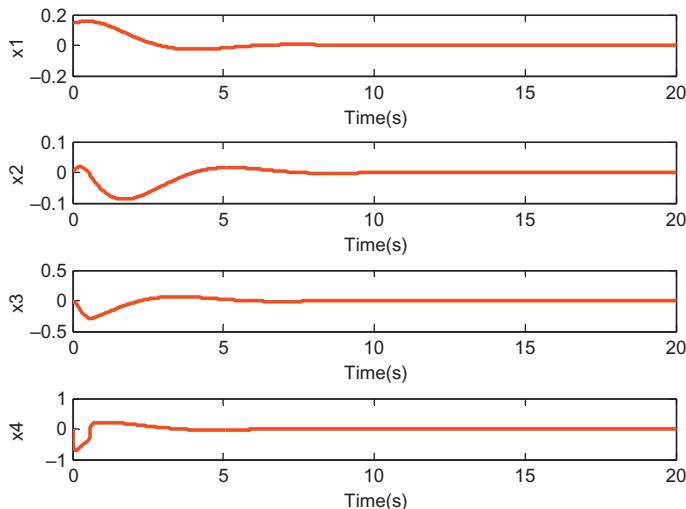
$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= g \sin x_1/l + x_3^3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u, \end{aligned}$$

where  $f_1(x_1, x_3) = g \sin x_1/l + x_3^3, l = 10$ .

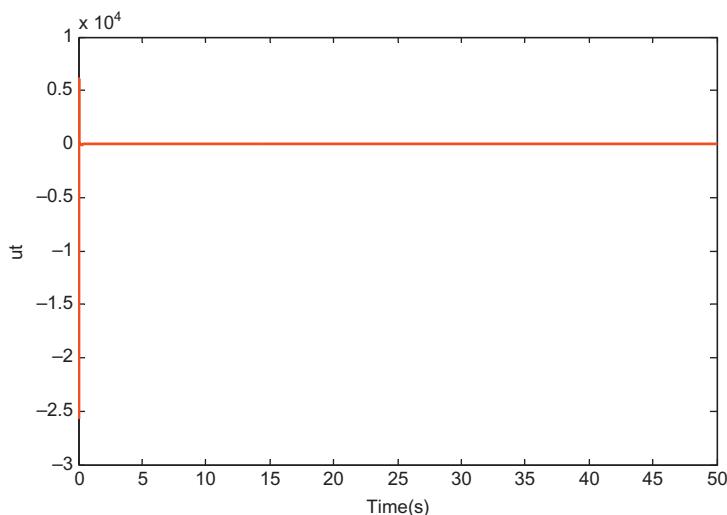
The control goals are  $x_i \rightarrow 0, i = 1, 2, 3, 4$ . From  $f_1(x_1, x_3)$  expression, we have  $\frac{\partial f_1}{\partial x_1} = g \cos x_1/l, \frac{\partial f_1}{\partial x_3} = 3x_3^2 + 1, \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_1} \right) = -\frac{g}{l} \sin x_1 \cdot x_2$

and  $\frac{d}{dt} \left( \frac{\partial f_1}{\partial x_3} \right) = 6x_3x_4$ , the initial system states are set as  $[0.1 \ 0.1 \ 0.1 \ 0.1]$ .

Considering  $\gamma = 1 + g/l = 1.98$ , choose  $a = 50$ , then we can get  $c_1$   $c_2$  and  $c_3$  by Eq. (8.35). Use controller (8.31), and set  $k = 1.0$ ,  $\eta = 5.0$ . Use the saturation function instead of the switch function, and set  $\Delta = 0.10$ . We obtain the simulation results, which are shown in Figs. 8.11 and 8.12.



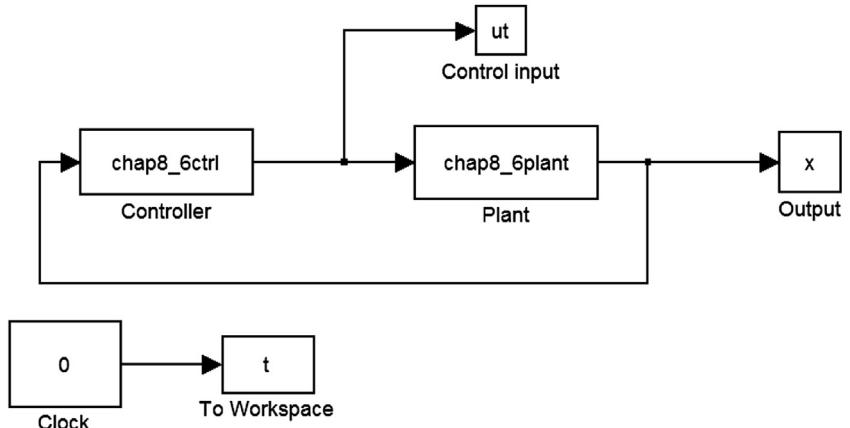
■ FIGURE 8.11 System states response.



■ FIGURE 8.12 Control input.

## Simulation Programs:

1. Main program: chap8\_6sim.mdl



2. S function of controller: chap8\_6ctrl.m

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];

```

```

ts = [0 0];
function sys=mdlOutputs(t,x,u)
x1=u(1);x2=u(2);x3=u(3);x4=u(4);
l=10;g=9.8;

f1=g*sin(x1)/l+x3^3+x3;
f1_x1=g*cos(x1)/l;
f1_x3=3*x3^2+1;

dt_f1_x1=-g*sin(x1)/l*x2;
dt_f1_x3=6*x3*x4;

e1=x1;
e2=x2;
e3=f1;
e4=f1_x1*x2+f1_x3*x4;

a=50;
c1=a^3;c2=3*a^2;c3=3*a;
s=c1*e1+c2*e2+c3*e3+e4;

M=c1*x2+c2*(f1-x3)+c3*(f1_x1*x2+f1_x3*x4)+dt_f1_x1
x2+f1_x1(f1-x3)+dt_f1_x3*x4;

k=1;xite=5;

```

```

fai=0.010;
if abs(s)<=fai
 sat=s/fai;
else
 sat=sign(s);
end

ut=1/f1_x3*(-M-k*s-xite*sat);
sys(1)=ut;

```

**3. S function of plant: chap8\_6plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);

```

```

case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.10;0.10;0.10;0.10];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
ut=u(1);
l=10;g=9.8;

sys(1)=x(2);
sys(2)=g*sin(x(1))/l+(x(3))^3;
sys(3)=x(4);
sys(4)=ut;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);
4. Plot program: chap8_6plot.m
close all;

figure(1);
subplot(411);
plot(t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('x1');
subplot(412);
plot(t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('x2');

```

```
subplot(413);
plot(t,x(:,3),'r','linewidth',2);
xlabel('time(s)');ylabel('x3');
subplot(414);
plot(t,x(:,4),'r','linewidth',2);
xlabel('time(s)');ylabel('x4');

figure(2);

plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('ut');
```

## REFERENCES

- [1] X. Rong, U. Ozguner, Sliding mode control of a class of underactuated systems, *Automatica* 44 (2008) 233–248.
- [2] H. Ashrafiuon, R.S. Erwin, Sliding mode control of underactuated multibody systems and its application to shape change control, *Int. J. Control.* 81 (12) (2008) 1849–1858.

# Sliding mode control for underactuated system with decoupling algorithm

In the paper [1,2], a general decoupling algorithm for under actuated system was proposed, refer to this paper, in this chapter, two sections are introduced as follows:

In [Section 9.1](#), general decoupling algorithm for underactuated system is introduced, an example is given. In [Section 9.2](#), consider a single inverted pendulum dynamic model, the standard decoupling algorithm is used to decouple the system, a sliding mode controller is designed by using Hurwitz, an example is given. In [Section 9.3](#), consider a translational oscillator with rotational actuator (TORA) system, the standard decoupling algorithm is used to decouple the system, a sliding mode controller is designed by using Hurwitz, an example is given.

## 9.1 GENERAL DECOUPLING ALGORITHM FOR UNDERACTUATED SYSTEM

R.O. Saber proposed a general decoupling algorithm for under actuated system [1,2].

Consider the following coupled under actuated system

$$\begin{aligned}\dot{q}_1 &= p_1 \\ \dot{p}_1 &= f_1(q, p) + g_1(q_2)u \\ \dot{q}_2 &= p_2 \\ \dot{p}_2 &= f_2(q, p) + g_2(q_2)u\end{aligned}, \quad (9.1)$$

where  $\mathbf{q} = [q_1 \ q_2]$ ,  $\mathbf{p} = [p_1 \ p_2]$ .

Model decoupling algorithm is designed as [1]:

$$\begin{aligned}x_1 &= q_1 - \int_0^{q_2} \frac{g_1(s)}{g_2(s)} ds \\x_2 &= p_1 - \frac{g_1(q_2)}{g_2(q_2)} p_2 \\x_3 &= q_2 \\x_4 &= p_2\end{aligned}\quad (9.2)$$

The essence of decoupling algorithm is to remove  $u$  in  $\dot{x}_2$ . From Eq. (9.2) we can get

$$\begin{aligned}\dot{x}_2 &= \dot{p}_1 - \left(\frac{g_1}{g_2}\right)' p_2 - \left(\frac{g_1}{g_2}\right) \dot{p}_2 \\&= f_1 + g_1 u - \left(\frac{g_1}{g_2}\right)' p_2 - \frac{g_1}{g_2} (f_2 + g_2 u) \\&= f_1 - \left(\frac{g_1}{g_2}\right)' p_2 - \frac{g_1}{g_2} f_2\end{aligned}$$

Then Eq. (9.1) can be decoupled as

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_1 - \left(\frac{g_1}{g_2}\right)' p_2 - \frac{g_1}{g_2} f_2 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= f_2 + g_2 u\end{aligned}\quad (9.3)$$

## 9.2 SLIDING MODE CONTROL FOR AN INVERTED PENDULUM

### 9.2.1 System description

Consider a single inverted pendulum dynamic model, near the equilibrium point, we have  $\sin \theta \approx \theta$ ,  $\cos \theta \approx 1$ , then we can get a linear equation for the single inverted pendulum as

$$\begin{aligned}\ddot{\theta} &= \frac{m(m+M)gl}{(M+m)I + Mml^2} \theta - \frac{ml}{(M+m)I + Mml^2} (u + d(t)) \\ \ddot{x} &= -\frac{m^2 gl^2}{(M+m)I + Mml^2} \theta + \frac{I + ml^2}{(M+m)I + Mml^2} (u + d(t))\end{aligned}, \quad (9.4)$$

where  $M$  is the mass of cart,  $m$  is the mass of pendulum,  $x$  is the position of cart,  $\theta$  is the pendulum angle,  $L$  denotes the pendulum length,  $l = \frac{1}{2}L$ ,  $u$  is the control input,  $I = \frac{1}{3}ml^2$  is the moment of inertia for the pendulum around its center of gravity and  $d(t)$  is disturbance.

The control goals are  $x \rightarrow 0$ ,  $\dot{x} \rightarrow 0$ ,  $\theta \rightarrow 0$  and  $\dot{\theta} \rightarrow 0$ .

### 9.2.2 Model decoupling

In order to design the sliding mode controller, model (9.4) needs to be decoupled. Define  $q_1 = \theta$ ,  $q_2 = x$ , then  $p_1 = \dot{\theta}$ ,  $p_2 = \dot{x}$ , define  $f_1(q, p) = \frac{m(m+M)gl}{(M+m)I + Mml^2} q_1$ ,  $g_1(q_2) = -\frac{ml}{(M+m)I + Mml^2}$ ,  $f_2(q, p) = -\frac{m^2 gl^2}{(M+m)I + Mml^2} q_1$  and  $g_2(q_2) = \frac{I+ml^2}{(M+m)I + Mml^2}$ , then  $\frac{g_1(q_2)}{g_2(q_2)} = \frac{\frac{ml}{(M+m)I + Mml^2}}{\frac{I+ml^2}{(M+m)I + Mml^2}} = -\frac{ml}{I+ml^2}$ .

Using the standard decoupling algorithm (9.2), we can get a decoupling algorithm as

$$\begin{aligned} z_1 &= q_1 - \int_0^{q_2} \frac{g_1(s)}{g_2(s)} ds = q_1 + \frac{ml}{I+ml^2} q_2 \\ z_2 &= p_1 - \frac{g_1(q_2)}{g_2(q_2)} p_2 = p_1 + \frac{ml}{I+ml^2} p_2 \quad . \\ \xi_1 &= q_2 \\ \xi_2 &= p_2 \end{aligned} \quad (9.5)$$

Then the balance point of the inverted pendulum  $\theta \rightarrow 0$ ,  $x \rightarrow 0$ ,  $\dot{\theta} \rightarrow 0$  and  $\dot{x} \rightarrow 0$  is equivalent to  $z_1 \rightarrow 0$ ,  $z_2 \rightarrow 0$  and  $\xi_1 \rightarrow 0$ ,  $\xi_2 \rightarrow 0$ .

Using the decoupling algorithm, we obtain

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= \left( \frac{m(m+M)gl}{(M+m)I + Mml^2} - \frac{ml}{I+ml^2} \frac{m^2 gl^2}{(M+m)I + Mml^2} \right) q_1 \\ \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= -\frac{m^2 gl^2}{(M+m)I + Mml^2} q_1 + \frac{I+ml^2}{(M+m)I + Mml^2} (u + d(t)) \end{aligned} \quad (9.6)$$

Defining  $T_1 = \frac{m(m+M)gl}{(M+m)I + Mml^2} - \frac{ml}{I+ml^2} \frac{m^2 gl^2}{(M+m)I + Mml^2}$ ,  $T_2 = -\frac{m^2 gl^2}{(M+m)I + Mml^2}$  and  $T_3 = \frac{I+ml^2}{(M+m)I + Mml^2}$ , Eq. (9.6) then becomes

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= T_1 q_1 \\ \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= T_2 q_1 + T_3 (u + d(t)) \end{aligned} \quad (9.7)$$

Considering  $q_1 = z_1 - \frac{ml}{I+ml^2} q_2$ ,  $q_2 = z_1 - \frac{ml}{I+ml^2} \xi_1$ ,  $\xi_1 = z_1 + T_4 \xi_1$  and  $T_4 = -\frac{ml}{I+ml^2}$ , then Eq. (9.7) becomes

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= T_1 z_1 + T_1 T_4 \xi_1 \\ \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= T_2 z_1 + T_2 T_4 \xi_1 + T_3 (u + d(t)) \end{aligned} \quad (9.8)$$

### 9.2.3 Sliding mode controller design

For Eq. (9.8), define  $\mu_1 = \xi_2$ ,  $\mu_2 = [z_1 \ z_2 \ \xi_1]^T$ , design the sliding mode controller as

$$\sigma = \mu_1 - C\mu_2, \quad (9.9)$$

where  $C = [c_1 \ c_2 \ c_3]$ .

Then we have

$$\dot{\sigma} = \dot{\mu}_1 - C\dot{\mu}_2 = T_2 z_1 + T_2 T_4 \xi_1 + T_3(u + d(t)) - C\dot{\mu}_2.$$

Design the sliding mode controller as

$$u = \frac{1}{T_3}(-T_2 z_1 - T_2 T_4 \xi_1 + C\dot{\mu}_2 - h\text{sgn}(\sigma)), \quad (9.10)$$

where  $h \geq |d(t)|$ .

Design the Lyapunov function as

$$V = \frac{1}{2}\sigma^2,$$

then

$$\sigma\dot{\sigma} = \sigma(-h\text{sgn}(\sigma) + d(t)) \leq 0.$$

From  $\sigma\dot{\sigma} \leq 0$ , under controller (9.10), the system can reach and thereafter stay on the manifold  $\sigma = 0$  in finite time  $t_s$ . On the manifold, we have  $\sigma = \mu_1 - C\mu_2 = 0$ , i.e.,  $\mu_1 = C\mu_2$ .

### 9.2.4 $C$ design

When  $t \geq t_s$ , we have

$$\dot{\mu}_2 = \begin{bmatrix} z_2 \\ T_1 z_1 + T_1 T_4 \xi_1 \\ \xi_2 \end{bmatrix} = A_1 \mu_1 + A_2 \mu_2 = (A_1 C + A_2) \mu_2, \quad (9.11)$$

$$\text{where } A_1 = [0 \ 0 \ 1]^T, A_2 = \begin{bmatrix} 0 & 1 & 0 \\ T_1 & 0 & T_1 T_4 \\ 0 & 0 & 0 \end{bmatrix}.$$

To guarantee  $\mu_2 \rightarrow 0$ ,  $A_1 C + A_2$  should be Hurwitz, thus we can get  $\mu_1 = C\mu_2 \rightarrow 0$ .

From  $\mu_1 = \xi_2$ ,  $\mu_2 = [z_1 \ z_2 \ \xi_1]^T$ , we know if  $t \rightarrow \infty$ , we have  $z_1 \rightarrow 0$ ,  $z_2 \rightarrow 0$ ,  $\xi_2 \rightarrow 0$  and  $\dot{\xi}_2 \rightarrow 0$ , that is,  $\theta \rightarrow 0$ ,  $\dot{\theta} \rightarrow 0$ ,  $x \rightarrow 0$  and  $\dot{x} \rightarrow 0$ .

$$\begin{aligned} A_1 C + A_2 &= [0 \ 0 \ 1]^T [c_1 \ c_2 \ c_3] + \begin{bmatrix} 0 & 1 & 0 \\ T_1 & 0 & T_1 T_4 \\ 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ c_1 & c_2 & c_3 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ T_1 & 0 & T_1 T_4 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ T_1 & 0 & T_1 T_4 \\ c_1 & c_2 & c_3 \end{bmatrix}. \end{aligned}$$

The poles of  $A_1 C + A_2$  can be designed by the following:

$$\begin{aligned} |sI - (A_1 C + A_2)| &= \begin{vmatrix} s & -1 & 0 \\ -T_1 & s & -T_1 T_4 \\ -c_1 & -c_2 & s - c_3 \end{vmatrix} \\ &= s^3 - c_3 s^2 - T_1 T_4 c_1 - c_2 T_1 T_4 s - T_1(s - c_3) \\ &= s^3 - c_3 s^2 - (c_2 T_1 T_4 + T_1)s - T_1 T_4 c_1 + T_1 c_3 = 0 \end{aligned} \quad . \quad (9.12)$$

From  $(s+k)^3 = 0$ ,  $k > 0$ , we have  $s^3 + 3ks^2 + 3k^2s + k^3 = 0$ ; comparing with Eq. (9.12), we can get

$$\begin{cases} -c_3 = 3k \\ -c_2 T_1 T_4 - T_1 = 3k^2 \\ -T_1 T_4 c_1 + T_1 c_3 = k^3 \end{cases}$$

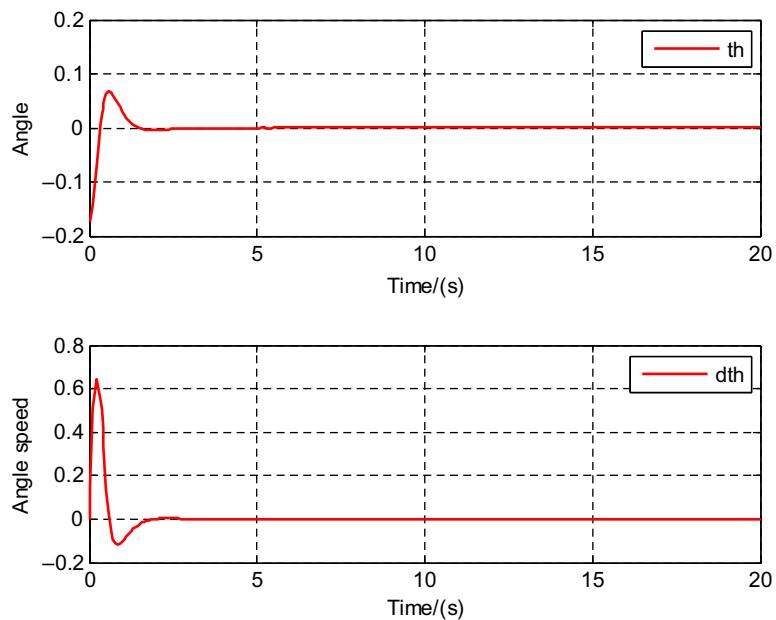
To make  $A_1 C + A_2$  Hurwitz, the sliding mode parameters are designed as follows:

$$\begin{cases} c_3 = -3k \\ c_2 = -\frac{3k^2 + T_1}{T_1 T_4} \\ c_1 = -\frac{k^3 - T_1 c_3}{T_1 T_4} \end{cases} \quad . \quad (9.13)$$

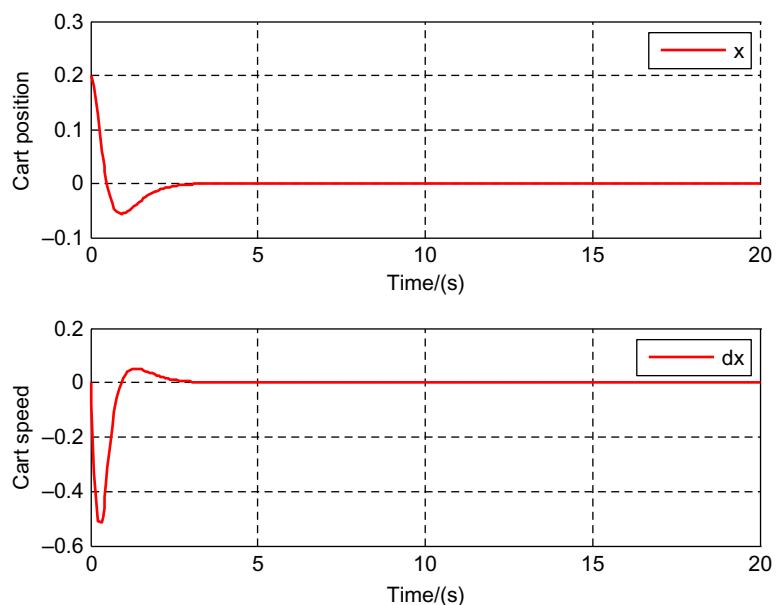
### 9.2.5 Simulation example

Considering plant as (9.4), choose  $M = 1$ ,  $m = 0.10$ ,  $L = 0.50$ , the initial states are set as  $\theta(0) = -10^\circ$ ,  $\dot{\theta}(0) = 0$ ,  $x(0) = 0.20$ ,  $\dot{x}(0) = 0$ , and set  $d(t) = \sin t$ .

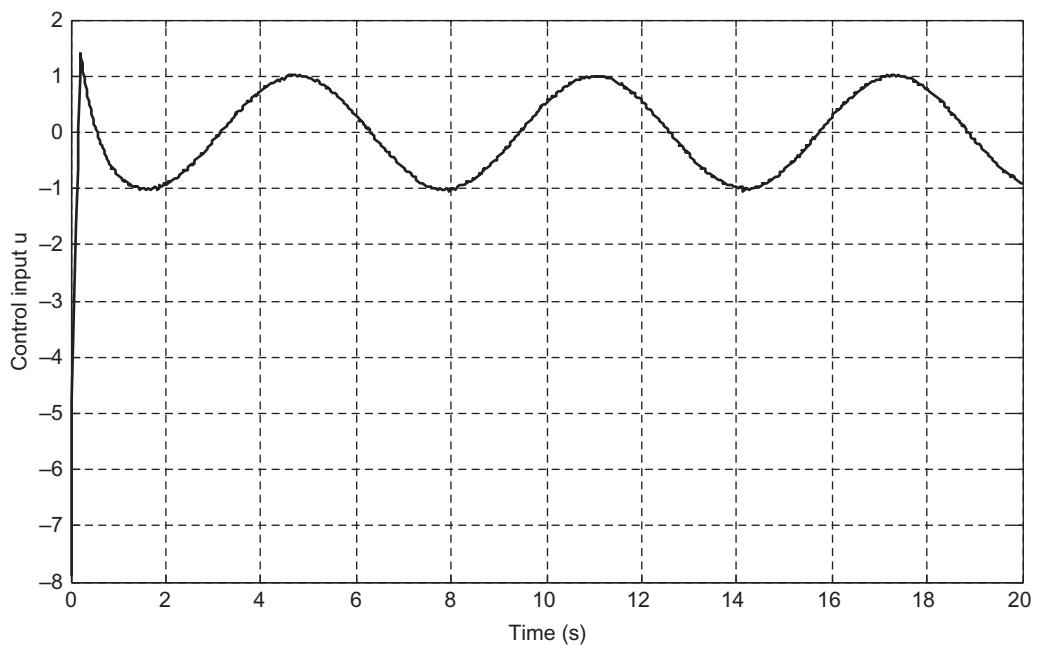
Use controller (9.10), design  $C$  by Eq. (9.13), choose  $k = 5$  and  $h = 1.5$ . Use the saturation function instead of the switch function, and set  $\Delta = 0.01$ . We can then obtain the simulation results, which are shown in Figs. 9.1–9.3.



■ FIGURE 9.1 Angle and angle speed of pendulum.



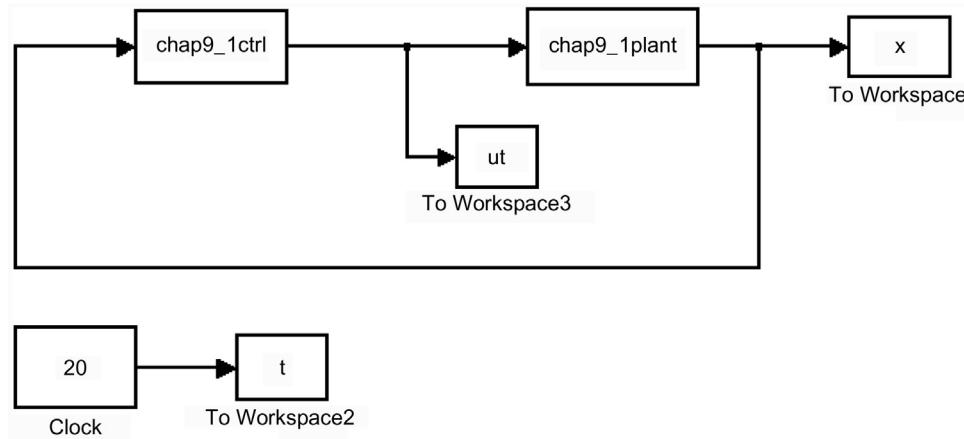
■ FIGURE 9.2 Position and speed of cart.



■ FIGURE 9.3 Control input.

Matlab programs:

1. Simulink main program: chap9\_1sim.mdl



```

2. S function for controller; chap9_1ctrl.m
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {1, 2, 4, 9}
 sys = [];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0=[];
str=[];
ts=[0 0];
function sys=mdlOutputs(t,x,u)
g=9.8;M=1.0;m=0.1;L=0.5;
I=1/12*m*L^2;l=1/2*L;

T4=-m*l/(I+m*l^2);
T2=-m^2*g*l^2/[(m+M)*I+M*m*l^2];
T3=(I+m*l^2)/[(m+M)*I+M*m*l^2];
T1=m*(M+m)*g*l/[(M+m)*I+M*m*l^2]-T4*T2;

th=u(1);dth=u(2);
x=u(3);dx=u(4);

q1=th;q2=x;
p1=dth;p2=dx;

z1=q1+m*l/(I+m*l^2)*q2;
z2=p1+m*l/(I+m*l^2)*p2;
kesi1=q2;
kesi2=p2;

```

```

miu1=kesi2;
miu2=[z1 z2 kesi1]';
%%%%%%%%%%%%%%%
k = 5;
c3 = - 3*k;
c2 = -(3*k^2 + T1)/(T1*T4);
c1 = -(k^3 - T1*c3)/(T1*T4);
C=[c1 c2 c3];
%%%%%%%%%%%%%%%
sigma=miu1-C*miu2;

dmiu2=[z2;T1*z1+T1*T4*kesi1;kesi2];

h=1.5;
fai=0.01;
if abs(sigma)<=fai
 sat=sigma/fai;
else
 sat=sign(sigma);
end
ut=1/T3*(-T2*z1-T2*T4*kesi1+C*dmiu2-h*sat);

sys(1)=ut;

```

**3. S function for the plant: chap9\_1plant.m**

```

function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 4;
sizes.NumInputs = 1;

```

```

sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [-10/57.3,0,0.20,0];
% $x_0 = [0.01, 0, 0.020, 0]$;
str = [];
ts = [0 0];
function sys = mdlDerivatives(t,x,u)
%Single Link Inverted Pendulum Parameters
g = 9.8; M = 1.0; m = 0.1; L = 0.5;

I = 1/12*m*L^2;
l = 1/2*L;
t1 = m*(M+m)*g*l/[(M+m)*I + M*m*l^2];
t2 = -m^2*g*l^2/[(m+M)*I + M*m*l^2];
t3 = -m*l/[(M+m)*I + M*m*l^2];
t4 = (I + m*l^2)/[(m+M)*I + M*m*l^2];

ut = u(1);
sys(1) = x(2);
sys(2) = t1*x(1) + t3*(ut + sin(t)); %ddth
sys(3) = x(4);
sys(4) = t2*x(1) + t4*(ut + sin(t)); %ddx
function sys = mdlOutputs(t,x,u)
sys(1) = x(1); %th
sys(2) = x(2);
sys(3) = x(3); %x
sys(4) = x(4);

```

**4. Plot program: chap9\_1plot.m**

```

close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r','linewidth',2);
legend('th'); xlabel('time/(s)'); ylabel('angle');
grid on;
subplot(212);
plot(t,x(:,2),'r','linewidth',2);
legend('dth'); xlabel('time/(s)'); ylabel('angle speed');
grid on;

figure(2);
subplot(211);
plot(t,x(:,3),'r','linewidth',2);

```

```

legend('x'); xlabel('time/(s)'); ylabel('cart position');
grid on;

subplot(212);
plot(t,x(:,4),'r','linewidth',2);
legend('dx'); xlabel('time/(s)'); ylabel('cart speed');
grid on;
figure(3);
plot(t,ut(:,1),'k','linewidth',2);
xlabel('time(s)'); ylabel('control input u');
grid on;

```

### 9.3 SLIDING MODE CONTROL FOR A TORA SYSTEM

A translational oscillator with rotational actuator (TORA) system has been constructed as a benchmark system to evaluate the performance of various nonlinear controllers.

#### 9.3.1 System description

The dynamics of the TORA system are described as

$$\begin{aligned}\dot{z}_1 &= z_2 \\ \dot{z}_2 &= \frac{-z_1 + \varepsilon\theta_2^2 \sin \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1} - \frac{\varepsilon \cos \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1} v \\ \dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= \frac{\varepsilon \cos \theta_1(z_1 - \varepsilon\theta_2^2 \sin \theta_1)}{1 - \varepsilon^2 \cos^2 \theta_1} + \frac{1}{1 - \varepsilon^2 \cos^2 \theta_1} v\end{aligned}, \quad (9.14)$$

where  $z_1$  is the normalized displacement of the platform from the equilibrium position,  $z_2 = \dot{z}_1$ ,  $\theta_1$  is the angle of the rotor,  $z_2 = \dot{z}_1$  and  $v$  is control input.

The control goals are  $z_1 \rightarrow 0$ ,  $\dot{z}_1 \rightarrow 0$ ,  $\theta_1 \rightarrow 0$  and  $\dot{\theta}_1 \rightarrow 0$  as  $t \rightarrow \infty$ .

#### 9.3.2 Model decoupling

Apply the decoupling algorithm in Eq. (9.2) to Eq. (9.14), and let  $f_1 = \frac{-z_1 + \varepsilon\theta_2^2 \sin \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1}$ ,  $g_1 = -\frac{\varepsilon \cos \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1}$ ,  $f_2 = \frac{\varepsilon \cos \theta_1(z_1 - \varepsilon\theta_2^2 \sin \theta_1)}{1 - \varepsilon^2 \cos^2 \theta_1}$  and  $g_2 = -\frac{1}{1 - \varepsilon^2 \cos^2 \theta_1}$ . We then have  $\frac{g_1}{g_2} = \varepsilon \cos \theta_1$ .

Define

$$x_1 = z_1 + \varepsilon \sin \theta_1, x_2 = z_2 + \varepsilon \theta_2 \cos \theta_1, x_3 = \theta_1, x_4 = \theta_2. \quad (9.15)$$

From Eq. (9.15), we can see that the control goals  $z_1 \rightarrow 0$ ,  $z_2 \rightarrow 0$ ,  $\theta_1 \rightarrow 0$ , and  $\theta_2 \rightarrow 0$  are equivalent to  $x_i \rightarrow 0$ ,  $i = 1, 2, 3, 4$ .

Since

$$\begin{aligned}
 \dot{x}_2 &= \dot{z}_2 + \varepsilon \dot{\theta}_2 \cos \theta_1 - \varepsilon \theta_2^2 \sin \theta_1 \\
 &= \frac{-z_1 + \varepsilon \theta_2^2 \sin \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1} - \frac{\varepsilon \cos \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1} v \\
 &\quad + \varepsilon \left( \frac{\varepsilon \cos \theta_1 (z_1 - \varepsilon \theta_2^2 \sin \theta_1)}{1 - \varepsilon^2 \cos^2 \theta_1} + \frac{1}{1 - \varepsilon^2 \cos^2 \theta_1} v \right) \cos \theta_1 - \varepsilon \theta_2^2 \sin \theta_1 \\
 &= \frac{-z_1 + \varepsilon \theta_2^2 \sin \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1} + \frac{\varepsilon^2 \cos \theta_1 (z_1 - \varepsilon \theta_2^2 \sin \theta_1)}{1 - \varepsilon^2 \cos^2 \theta_1} - \varepsilon \theta_2^2 \sin \theta_1 \\
 &= \frac{-z_1 (1 - \varepsilon^2 \cos \theta_1) + \varepsilon \theta_2^2 \sin \theta_1 (1 - \varepsilon^2 \cos \theta_1)}{1 - \varepsilon^2 \cos^2 \theta_1} - \varepsilon \theta_2^2 \sin \theta_1 \\
 &= -z_1 = -x_1 + \varepsilon \sin x_3
 \end{aligned}$$

let  $u = \frac{\varepsilon \cos \theta_1 (z_1 - \varepsilon \theta_2^2 \sin \theta_1)}{1 - \varepsilon^2 \cos^2 \theta_1} - \frac{1}{1 - \varepsilon^2 \cos^2 \theta_1} v$ , i.e.,  $v = \varepsilon \cos \theta_1 (z_1 - \varepsilon \theta_2^2 \sin \theta_1) - (1 - \varepsilon^2 \cos^2 \theta_1) u$ , from  $z_1 = x_1 - \varepsilon \sin x_3$ , we have

$$v = \varepsilon \cos x_3 (x_1 - (1 + x_4^2) \varepsilon \sin x_3) - (1 - \varepsilon^2 \cos^2 x_3) u. \quad (9.16)$$

From the above analysis, Eq. (9.14) can be decoupled as

$$\begin{aligned}
 \dot{x}_1 &= x_2 \\
 \dot{x}_2 &= f_1(x_1, x_3) \\
 \dot{x}_3 &= x_4 \\
 \dot{x}_4 &= u
 \end{aligned}, \quad (9.17)$$

where  $f_1(x_1, x_3) = -x_1 + \varepsilon \sin x_3$ .

In paper [3], a sliding mode controller is designed for a kind of underactuated system, in which  $f_1(x_1, x_3)$  must be satisfied as three assumptions as follows:

- Assumption 1:  $f_1(0, 0) \rightarrow 0$ ;
- Assumption 2:  $\frac{\partial f_1}{\partial x_3}$  is invertible;
- Assumption 3: if  $f_1(0, x_3) \rightarrow 0$ , then  $x_3 \rightarrow 0$ .

Obviously if Eq. (9.17),  $\frac{\partial f_1}{\partial x_3}$  is not satisfied by Assumption 2, we can redefine  $f_1(x_1, x_3)$  as

$$f_1(x_1, x_3) = -x_1 + \varepsilon \sin x_3 + 11\varepsilon x_3.$$

Then  $\frac{\partial f_1}{\partial x_3} = \varepsilon \cos x_3 + 11\varepsilon$ , and Eq. (9.17) becomes

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_1(x_1, x_3) - 11\varepsilon x_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= u\end{aligned}. \quad (9.18)$$

### 9.3.3 Sliding mode controller design

To realize  $x_i \rightarrow 0$ , define the error equation as

$$\begin{aligned}e_1 &= x_1 \\ e_2 &= \dot{e}_1 = x_2 \\ e_3 &= f_1(x_1, x_3) \\ e_4 &= \dot{e}_3 = \dot{f}_1(x_1, x_3) = \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_3} x_4\end{aligned}. \quad (9.19)$$

In the above equation, to prevent singularity in the controller design, we do not let  $e_3 = \dot{e}_2$ .

Design the sliding mode function as

$$s = c_1 e_1 + c_2 e_2 + c_3 e_3 + e_4, \quad (9.20)$$

where  $c_i > 0$ ,  $i = 1, 2, 3$ .

From  $\frac{d}{dt} \left( \frac{\partial f_1}{\partial x_1} \right) = 0$ , we have

$$\begin{aligned}\dot{s} &= c_1 \dot{e}_1 + c_2 \dot{e}_2 + c_3 \dot{e}_3 + \dot{e}_4 \\ &= c_1 x_2 + c_2 (f_1 - 11\varepsilon x_3) + c_3 e_4 + \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_3} x_4 \right),\end{aligned} \quad (9.21)$$

where  $\frac{d}{dt} \left( \frac{\partial f_1}{\partial x_1} x_2 + \frac{\partial f_1}{\partial x_3} x_4 \right) = \frac{\partial f_1}{\partial x_1} (f_1 - 11\varepsilon x_3) + \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_3} \right) x_4 + \frac{\partial f_1}{\partial x_3} u$ .

Then

$$\dot{s} = c_1 x_2 + c_2 (f_1 - 11\varepsilon x_3) + c_3 e_4 + \frac{\partial f_1}{\partial x_1} (f_1 - 11\varepsilon x_3) + \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_3} \right) x_4 + \frac{\partial f_1}{\partial x_3} u$$

Let  $M = c_1 x_2 + c_2 (f_1 - 11\varepsilon x_3) + c_3 e_4 + \frac{\partial f_1}{\partial x_1} (f_1 - 11\varepsilon x_3) + \frac{d}{dt} \left( \frac{\partial f_1}{\partial x_3} \right) x_4$ , and design the sliding mode controller as

$$u = \left[ \frac{\partial f_1}{\partial x_3} \right]^{-1} (-M - \eta \text{sgn}s - ks), \quad (9.22)$$

where  $\eta > 0$  and  $k > 0$ .

Substituting Eq. (9.22) into  $\dot{s}$ , we have  $\dot{s} = -\eta \text{sgn}(s) - ks$ . Design the Lyapunov function  $V = \frac{1}{2}s^2$ , then

$$\dot{V} = s\dot{s} = -\eta|s| - ks^2 \leq 0.$$

### 9.3.4 Convergence analysis

From Eq. (9.19), we have  $\dot{e}_1 = e_2$ ,  $\dot{e}_2 = e_3 + 11\varepsilon x_3$ ,  $\dot{e}_3 = e_4$ . From  $s\dot{s} \leq 0$ , under controller (9.22), the system can reach and thereafter stay on the manifold  $s = 0$  in finite time. On the manifold  $s = 0$ , we have  $e_4 = -c_1 e_1 - c_2 e_2 - c_3 e_3$ .

Define  $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -c_1 & -c_2 & -c_3 \end{bmatrix}$  and design  $c_i$  to guarantee that  $A$  is Hurwitz; we then get the error state equation as

$$\begin{aligned} \dot{e}_1 &= e_2 \\ \dot{e}_2 &= e_3 - 11\varepsilon x_3 \\ \dot{e}_3 &= -c_1 e_1 - c_2 e_2 - c_3 e_3 \end{aligned} \quad . \quad (9.23)$$

Defining  $E = [e_1 \ e_2 \ e_3]^T$ ,  $d_2 = -11\varepsilon x_3$ , from  $e_3 = f_1(x_1, x_3) = -x_1 + \varepsilon \sin x_3 + 11\varepsilon x_3$ , then  $d_2 = -11\varepsilon x_3 = -e_3 - x_1 + \varepsilon \sin x_3$ .

Considering  $\sin x_3 < x_3$ , we then have

$$|d_2| = 11\varepsilon|x_3| = |-e_3 - x_1 + \varepsilon \sin x_3| \leq |e_3| + |e_1| + \varepsilon|x_3|.$$

Then  $10\varepsilon|x_3| \leq |e_3| + |e_1|$ , and

$$|d_2| = 11\varepsilon|x_3| = 1.1 \times 10\varepsilon|x_3| \leq 1.1(|e_3| + |e_1|) \leq 2.2||E||_2.$$

Let  $D = [0 \ d_2 \ 0]^T$ ; we have  $||D||_2 \leq \gamma ||E||_2$ ,  $\gamma = 2.2$ . From Eq. (9.23), we have

$$\dot{E} = AE + D. \quad (9.24)$$

To guarantee that  $E = [e_1 \ e_2 \ e_3]^T \rightarrow 0$ ,  $A$  must be Hurwitz, i.e., the real part of the characteristic value of  $A$  must be negative.

From  $|A - \lambda I| = \begin{vmatrix} -\lambda & 1 & 0 \\ 0 & -\lambda & 1 \\ -c_1 & -c_2 & -c_3 - \lambda \end{vmatrix} = \lambda^2(-c_3 - \lambda) - c_1 - c_2\lambda = -\lambda^3 - c_3\lambda^2 - c_2\lambda - c_1 = 0$ , i.e.,  $\lambda^3 + c_3\lambda^2 + c_2\lambda + c_1 = 0$ . From  $(\lambda + a)^3 = 0$ , we have  $\lambda^3 + 3a\lambda^2 + 3a^2\lambda + a^3 = 0$ .

If we choose  $c_1 = a^3$ ,  $c_2 = 3a^2$ ,  $c_3 = 3a$ , and let  $a > 0$ , then  $A$  will be Hurwitz. Defining  $\mathbf{Q} = \mathbf{Q}^T > 0$ , there exists a Lyapunov equation  $A^T \mathbf{P} + \mathbf{P}A = -\mathbf{Q}$ ,  $\mathbf{P} = \mathbf{P}^T > 0$ .

Design the Lyapunov function as  $V_1 = \mathbf{E}^T \mathbf{P} \mathbf{E}$ , then we have

$$\begin{aligned}\dot{V}_1 &= \dot{\mathbf{E}}^T \mathbf{P} \mathbf{E} + \mathbf{E}^T \mathbf{P} \dot{\mathbf{E}} = (\mathbf{A}\mathbf{E} + \mathbf{D})^T \mathbf{P} \mathbf{E} + \mathbf{E}^T \mathbf{P}(\mathbf{A}\mathbf{E} + \mathbf{D}) \\ &= \mathbf{E}^T \mathbf{A}^T \mathbf{P} \mathbf{E} + \mathbf{D}^T \mathbf{P} \mathbf{E} + \mathbf{E}^T \mathbf{P} \mathbf{A} \mathbf{E} + \mathbf{E}^T \mathbf{P} \mathbf{D} = \mathbf{E}^T (\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A}) \mathbf{E} + \mathbf{D}^T \mathbf{P} \mathbf{E} + \mathbf{E}^T \mathbf{P} \mathbf{D} \\ &= -\mathbf{E}^T \mathbf{Q} \mathbf{E} + \mathbf{D}^T \mathbf{P} \mathbf{E} + \mathbf{E}^T \mathbf{P} \mathbf{D} \leq -\lambda_{\min}(\mathbf{Q}) \|\mathbf{E}\|_2^2 + 2\lambda_{\max}(\mathbf{P})\gamma \|\mathbf{E}\|_2^2 \\ &= (-\lambda_{\min}(\mathbf{Q}) + 2\lambda_{\max}(\mathbf{P})\gamma) \|\mathbf{E}\|_2^2\end{aligned},$$

where  $\mathbf{D}^T \mathbf{P} \mathbf{E} + \mathbf{E}^T \mathbf{P} \mathbf{D} \leq 2\lambda_{\max}(\mathbf{P})\gamma \|\mathbf{E}\|_2^2$ ,  $\lambda_{\min}(\mathbf{Q})$  is the minimum eigenvalue of positive definite matrix  $\mathbf{Q}$  and  $\lambda_{\max}(\mathbf{P})$  is the maximum eigenvalue of positive definite matrix  $\mathbf{P}$ .

Let  $\frac{\lambda_{\min}(\mathbf{Q})}{2\lambda_{\max}(\mathbf{P})} > \gamma$ , then we have  $\dot{V}_1 < 0$ .

If we let  $\mathbf{Q} = \mathbf{I}_3$ , we have  $\lambda_{\min}(\mathbf{Q}) = 1$ ,  $\lambda_{\max}(\mathbf{P}) = \frac{1}{2\lambda_{\text{left}}(-\mathbf{A})}$ , then  $\frac{\lambda_{\min}(\mathbf{Q})}{2\lambda_{\max}(\mathbf{P})} > \gamma$  becomes  $\lambda_{\text{left}}(-\mathbf{A}) > \gamma$ , which can be guaranteed by  $A$  design.

Since  $\mathbf{A}$  is Hurwitz, then  $\mathbf{E} = [e_1 \ e_2 \ e_3]^T \rightarrow 0$  will be satisfied, i.e.,  $e_1 \rightarrow 0$ ,  $e_2 \rightarrow 0$  and  $e_3 \rightarrow 0$ ; moreover, from  $s \rightarrow 0$ , we have  $e_4 \rightarrow 0$ .

From  $e_1 \rightarrow 0$  and  $e_2 \rightarrow 0$ , we can get  $x_1 \rightarrow 0$ ,  $x_2 \rightarrow 0$ , from  $e_3 = f_1(0, 0, x_3) \rightarrow 0$ , Assumption 3 and Eq. (9.19), we can get  $x_3 \rightarrow 0$ ,  $x_4 \rightarrow 0$ .

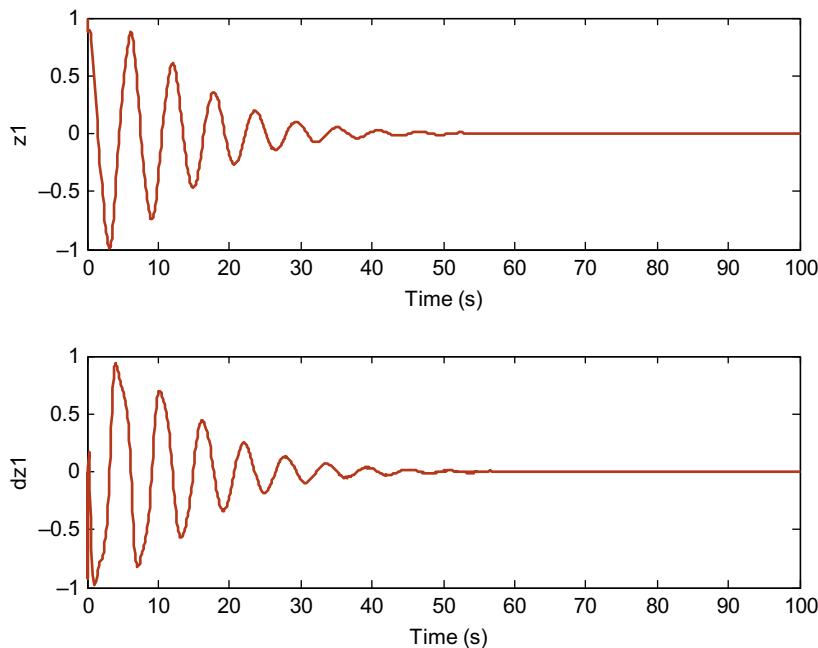
### 9.3.5 Simulation example

For a TORA system as

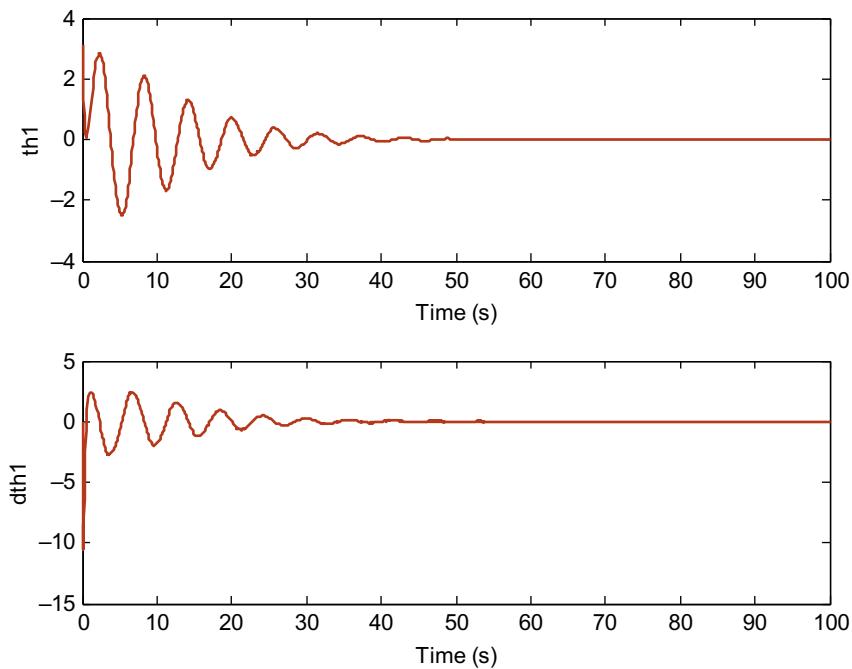
$$\begin{aligned}\dot{z}_1 &= z_2 \\ \dot{z}_2 &= \frac{-z_1 + \varepsilon \theta_2^2 \sin \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1} - \frac{\varepsilon \cos \theta_1}{1 - \varepsilon^2 \cos^2 \theta_1} v \\ \dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= \frac{\varepsilon \cos \theta_1 (z_1 - \varepsilon \theta_2^2 \sin \theta_1)}{1 - \varepsilon^2 \cos^2 \theta_1} - \frac{1}{1 - \varepsilon^2 \cos^2 \theta_1} v\end{aligned},$$

the initial states are set as  $[1 \ 0 \ \pi \ 0]$ . To satisfy  $\lambda_{\text{left}}(-\mathbf{A}) > \gamma$ , choose  $a = 3.0$ , then we have  $c_1 = a^3$  and  $c_2 = 3a^2$ ,  $c_3 = 3a$ .

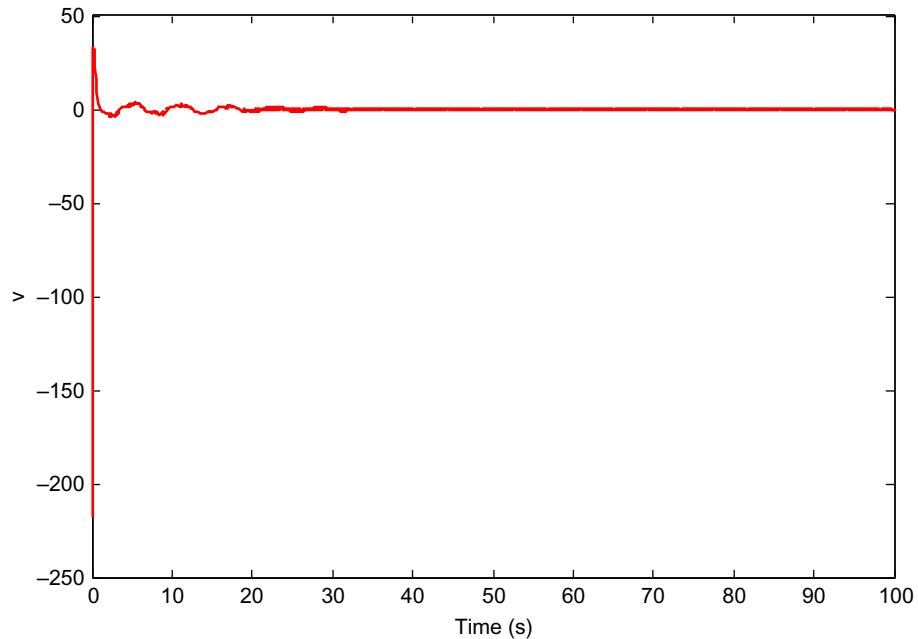
Using control law (9.16) and Eq. (9.22), set  $k = 5.0$ ,  $\eta = 0.50$ , use the saturation function instead of the switch function and let  $\Delta = 0.10$ . The simulation results are shown in Figs. 9.4–9.6.



■ FIGURE 9.4 Response of  $z_1$  and  $\dot{z}_1$ .



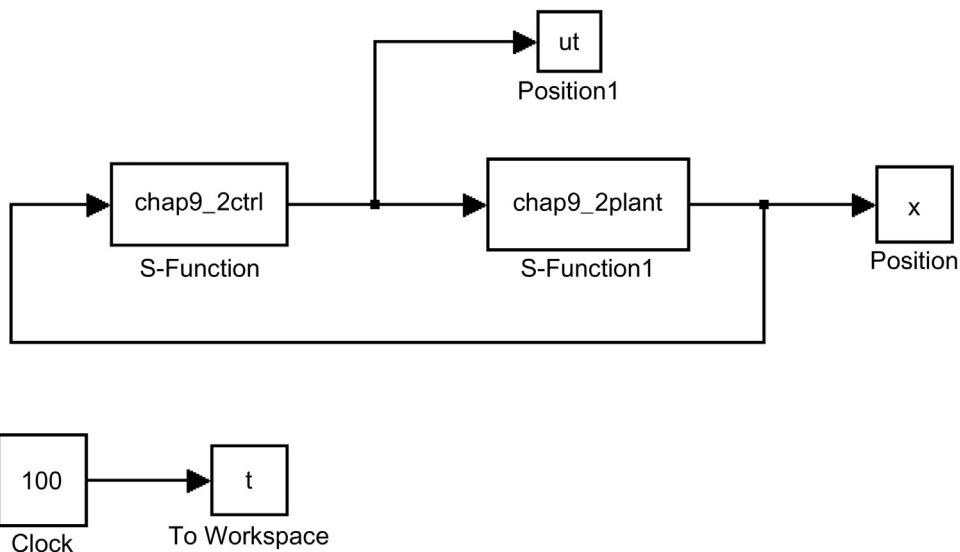
■ FIGURE 9.5 Response of  $\theta_1$  and  $\dot{\theta}_1$ .



■ FIGURE 9.6 Control input.

Matlab programs:

1. Simulink main program: chap9\_2sim.mdl



2. S function for controller; chap9\_2ctrl.m

```
function [sys,x0,str,ts] = s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
 sys=mdlOutputs(t,x,u);
case {2,4,9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)
epc=0.1;
z1=u(1);
z2=u(2);
theta1=u(3);
theta2=u(4);

x1=z1+epc*sin(theta1);
x2=z2+epc*theta2*cos(theta1);
x3=theta1;
x4=theta2;

f1=-x1+epc*sin(x3)+11*epc*x3;
f1_x1=-1;
f1_x3=epc*(cos(x3)+11);

dt_f1_x1=0;
dt_f1_x3=-epc*sin(x3)*x4;
```

```

e1=x1;
e2=x2;
e3=f1;
e4=f1_x1*x2+f1_x3*x4;

a=3.0;
c1=a^3;c2=3*a^2;c3=3*a;
s=c1*e1+c2*e2+c3*e3+e4;

M=c1*x2+c2*(f1-11*epc*x3)+c3*e4+f1_x1*
(f1-11*epc*x3)+dt_f1_x3*x4;

k=5;xite=0.5;

fai=0.10;
if abs(s)<=fai
 sat=s/fai;
else
 sat=sign(s);
end

ut=1/f1_x3*(-M-k*s-xite*sat);
v=(1-epc^2*(cos(theta1))^2)*ut-epc*cos(theta1)*
(z1-epc*theta2^2*sin(theta1));

sys(1)=v;

```

**3. S function for the plant: chap9\_2plant.m**

```

function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
 [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
 sys=mdlDerivatives(t,x,u);
case 3,
 sys=mdlOutputs(t,x,u);
case {2, 4, 9}
 sys=[];
otherwise
 error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates=4;
sizes.NumDiscStates=0;
sizes.NumOutputs=4;

```

```

sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[1;0;pi;0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
z1=x(1);
thetal1=x(3);
theta2=x(4);

epc=0.1;
v=u(1);

den=1-epc^2*(cos(theta1))^2;
dz2=(-z1+epc*theta2^2*sin(theta1)-
epc*cos(theta1)*v)/den;
dth2=(epc*cos(theta1)*(z1-epc*theta2^2*
sin(theta1))+v)/den;

sys(1)=x(2);
sys(2)=dz2;
sys(3)=x(4);
sys(4)=dth2;

```

**4. Plot program: chap9\_2plot.m**

```

close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('z1');
subplot(212);
plot(t,x(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('dz1');

figure(2);
subplot(211);
plot(t,x(:,3),'r','linewidth',2);

```

```
xlabel('time(s)');ylabel('th1');
subplot(212);
plot(t,x(:,4),'r','linewidth',2);
xlabel('time(s)');ylabel('dth1');

figure(3);
plot(t,ut(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('v');
```

## REFERENCES

- [1] R.O. Saber, Global configuration stabilization for the VTOL aircraft with strong input coupling, *IEEE Trans. Autom. Control.* 47 (11) (2002) 1949–1959.
- [2] R.O. Saber, Normal forms for underactuated mechanical systems with symmetry, *IEEE Trans. Autom. Control.* 47 (2) (2002) 305–308.
- [3] X. Rong, U. Ozguner, Sliding mode control of a class of underactuated systems, *Automatica* 44 (2008) 233–248.