9 Fuzzy Sliding Mode Control

Jinkun Liu
Beijing University of Aeronautics and Astronautics
P.R.China
E-mail: ljk@buaa.edu.cn

Xinhua Wang
National University of Singapore
Singapore
E-mail: wangxinhua04@gmail.com

Abstract This chapter introduces four kinds of fuzzy sliding mode controllers design, including fuzzy sliding mode control based on equivalent control, sliding mode control based on fuzzy switch-gain regulation, sliding mode control based on fuzzy system approximation and adaptive fuzzy control based on fuzzy compensation for manipulator.

Keywords fuzzy sliding mode control, equivalent control, fuzzy switch-gain, fuzzy system approximation, adaptive fuzzy control

Section 9.1 deals with the control system that comprises a logic fuzzy control design and an equivalent control. We introduce a fuzzy sliding mode control based on the equivalent control scheme for a class of nonlinear systems. The sliding mode control law in the existing equivalent sliding mode control system is directly substituted by a fuzzy logic controller. Hence, a control input without chattering is obtained in the systems with uncertainties. The chattering phenomenon in the sliding mode control is attenuated.

A chattering-free fuzzy sliding-mode control strategy for uncertain systems is introduced in section 9.2. The discontinuous switch gain in the traditional sliding-mode control is replaced by a fuzzy logic control. Hence, a control input without chattering is obtained in the systems with uncertainties. Based on the Lyapunov stability theory, we address the design schemes of the fuzzy sliding-mode control where the fuzzy control is designed by a set of linguistic rules and the control input is chattering free.

Past research of the universal approximation theorem[1] shows that any nonlinear function over a compact set with arbitrary accuracy can be approximated by a

J. Liu et al., *Advanced Sliding Mode Control for Mechanical Systems*

fuzzy system. There have been significant research efforts on the adaptive fuzzy control for nonlinear systems[2]. Section 9.3 and section 9.4 propose an adaptive fuzzy sliding mode control algorithm for a class of continuous time unknown nonlinear systems. The unknown nonlinearities are approximated by the fuzzy system with a set of fuzzy IF-THEN rules whose parameters are adjusted on-line according to some adaptive laws. This aids in controlling the output of the nonlinear system to track a given trajectory. The Lyapunov synthesis approach is used to develop an adaptive control algorithm which is based on the adaptive fuzzy model. The chattering action is attenuated and robust performance can be ensured. The stability analysis for the proposed control algorithm is provided.

9.1 Fuzzy Sliding Mode Control Based on Equivalent Control

In section 6 of Chapter 1, the equivalent sliding mode control has been described. The control law consists of an equivalent control u_{eq} and a switch control u_s. System states are kept on the sliding surface by an equivalent control and system states attain the sliding surface by a switch control. Using fuzzy rules, the fuzzy system is established based on the equivalent control and the switch control.

The sliding mode control based on an equivalent control uses the switching-gain switch control to reduce the chattering phenomenon and guarantee the Lyapunov stability. The small switching-gain switch control is adopted when the magnitudes of disturbances are small and the large switching-gain switch control is adopted when magnitudes of disturbances are large. This function can be realized by fuzzy system with fuzzy rules[3].

9.1.1 Design of Fuzzy Control

Consider the following system with disturbances and uncertainties:

$$\ddot{x} = f(x,t) + g(x,t)u(t) + d(t) \tag{9.1}$$

Let the tracking error be $e = x_d - x$, and the switch function be

$$s = ce + \dot{e} \tag{9.2}$$

According to the sliding mode control theory, a sliding mode controller consists of equivalent sliding mode control and switch control. The control law is shown as follows:

$$\text{If } s(t) \text{ is } N \text{ then } \mu \text{ is } P \tag{9.3a}$$

$$\text{If } s(t) \text{ is } Z \text{ then } \mu \text{ is } Z \tag{9.3b}$$

$$\text{If } s(t) \text{ is } P \text{ then } \mu \text{ is } P \tag{9.3c}$$

where the fuzzy sets Z, N and P denote "zero", "negative", and "positive" respectively.

Fuzzy rule (9.3b) states that the fuzzy controller is the equivalent control u_{eq} when the switch function is equal to zero. Also, fuzzy rules (9.3a) and (9.3c) state that the fuzzy controller is the equivalent control $u_{\text{eq}} + \mu \times$ the switch control u_{s} when switch function is not equal to zero.

The output of the fuzzy deduce system is membership μ. Adopting reverse fuzzification method, the fuzzy control is designed as:

$$u = u_{\text{eq}} + \mu \cdot u_{\text{s}} \tag{9.4}$$

When $\mu = 1$, $u = u_{\text{eq}} + u_{\text{s}}$. Therefore, the control law is the traditional equivalent sliding mode control. When $\mu \neq 1$, the chattering phenomenon can be reduced by the variant membership μ.

9.1.2 Simulation Example

Consider the kinetic equation of a single-stage inverted pendulum as follows:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(\boldsymbol{x}) + g(\boldsymbol{x}) \cdot u + d(t) \end{cases}$$

where

$$f(\boldsymbol{x}) = \frac{g \sin x_1 - mlx_2^2 \cos x_1 \sin x_1 /(m_{\text{c}} + m)}{l(4/3 - m\cos^2 x_1 /(m_{\text{c}} + m))}$$

$$g(\boldsymbol{x}) = \frac{\cos x_1 /(m_{\text{c}} + m)}{l(4/3 - m\cos^2 x_1 /(m_{\text{c}} + m))}$$

$\boldsymbol{x} = [x_1 \quad x_2]$, x_1 and x_2 are the roll angle and roll rate respectively, $g = 9.8\ \text{m/s}^2$, u is the control input, $m_{\text{c}} = 1\ \text{kg}$ is the mass of the vehicle, $m = 0.1\ \text{kg}$ is the mass of the pendulum, $l = 0.5\ \text{m}$ is the length of one half of the pendulum.

Consider the disturbance $d(t)$ with the form of Gauss function in the following:

$$d(t) = 5\exp\left(-\frac{(t - c_i)^2}{2b_i^2}\right)$$

Let $b_i = 0.50$, $c_i = 5.0$ and $\eta = 0.15$. The upper bound of disturbance is $D = \max(|d(t)|) + \eta = 5.15$. The disturbance $d(t)$ is shown in Fig. 9.1. The desired position trajectory is $x_{\text{d}} = \sin t$.

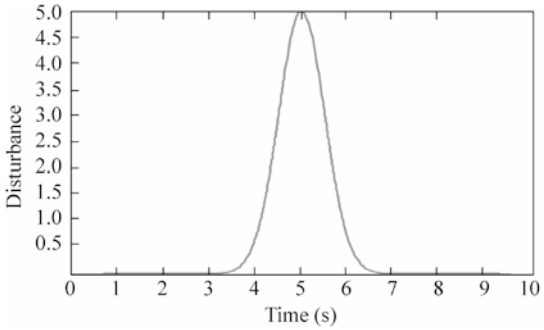


Figure 9.1 The disturbance with the form of Gauss function

The fuzzy system is established in S-function chap4_5s.m and the rule library is kept running by the command of persistent. The membership functions of input and output of fuzzy system are in Figs. 9.2 and 9.3 respectively. Fuzzy rule is designed as:

(1) If (*s* is *N*) then (*Mu* is *P*);

(2) If (*s* is *Z*) then (*u* is *Z*);

(3) If (*s* is *P*) then (*Mu* is *P*).

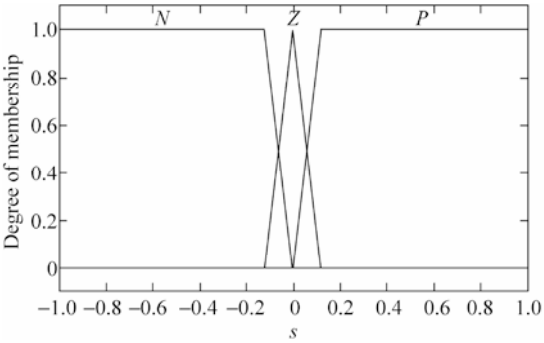


Figure 9.2 The membership function of fuzzy input *s*

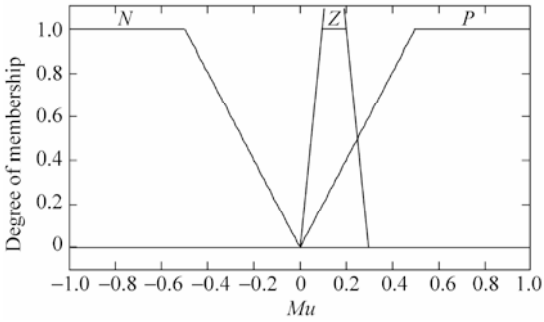


Figure 9.3 The membership function of fuzzy output μ

Use control law (9.4) with $\mu = 1$, and let $c = 25$, simulation results are shown in Fig. 9.4 and Fig. 9.5. Using control law (9.4) with $\mu \neq 1$, and let $c = 25$, simulation results are shown in Fig. 9.6 – Fig. 9.8. It can be found that the chattering

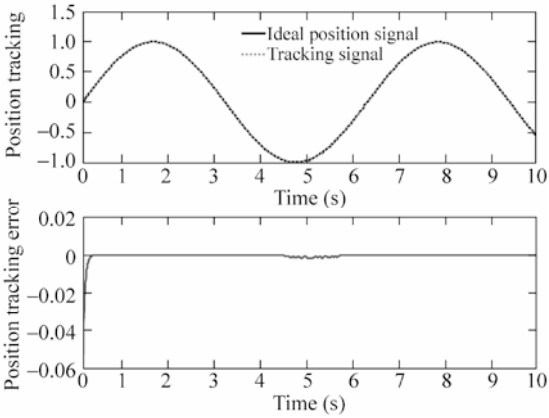


Figure 9.4 Position tracking by equivalent sliding mode control ($\mu = 1$)

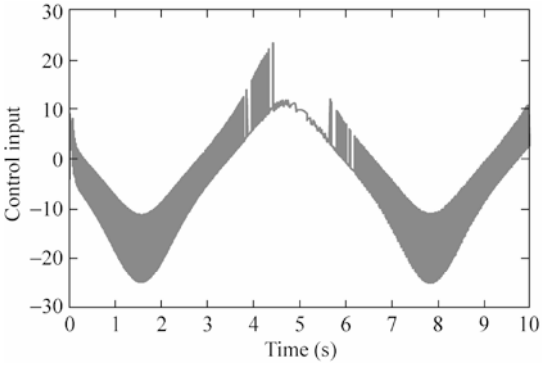


Figure 9.5 Control input ($\mu = 1$)

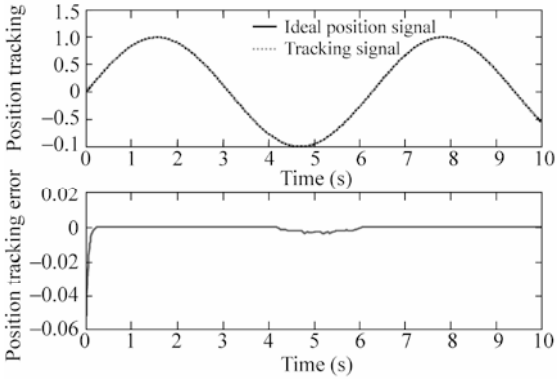


Figure 9.6 Position tracking by fuzzy sliding mode control ($\mu \neq 1$)

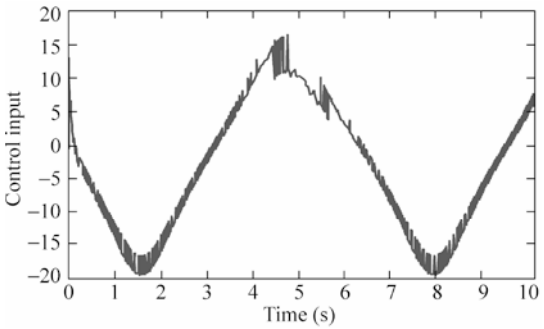


Figure 9.7 Control input $(\mu \neq 1)$

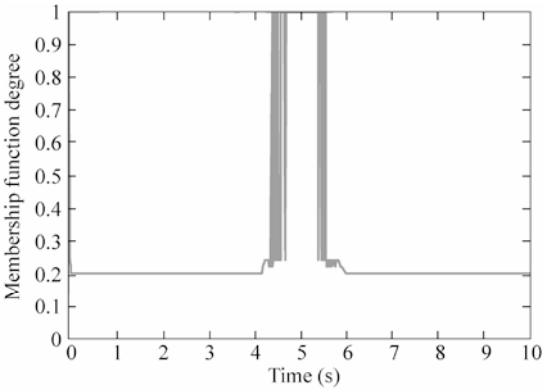


Figure 9.8 Membership function $\mu_{NZ}(s)$ $(\mu \neq 1)$

phenomenon can be reduced effectively when the fuzzy sliding mode control based on an equivalent control is adopted.

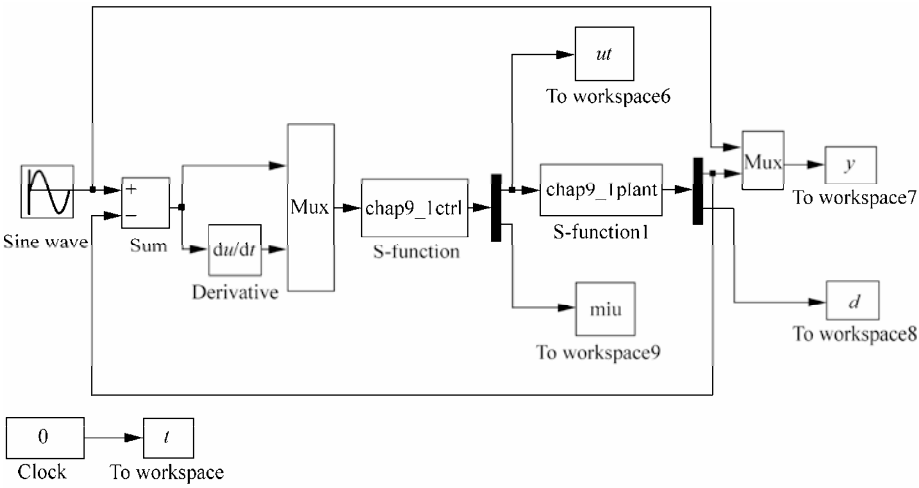
Simulation programs:

(1) Fuzzy logic system program: chap9_1fuzz.m

```
close all
clear all;

a=newfis('fuzz_smc');

a=addvar(a,'input','s',1/25*[-25,25]);
a=addmf(a,'input',1,'N','trapmf',1/25*[-25,-25,-3,0]);
a=addmf(a,'input',1,'Z','trimf',1/25*[-3,0,3]);
a=addmf(a,'input',1,'P','trapmf',1/25*[0,3,25,25]);

% a=addvar(a,'output','Mu',20*[-5,5]);
% a=addmf(a,'output',1,'N','trapmf',20*[-5,-5,-3,0]);
% a=addmf(a,'output',1,'Z','trimf',20*[-3,0,3]);
% a=addmf(a,'output',1,'P','trapmf',20*[0,3,5,5]);
```

```
a=addvar(a,'output','Mu',[-1,1]);
a=addmf(a,'output',1,'N','trapmf',[-1,-1,-0.5,0]);
a=addmf(a,'output',1,'Z','trapmf',[0,0.1,0.2,0.3]);
a=addmf(a,'output',1,'P','trapmf',[0,0.5,1,1]);

rulelist=[1 3 1 1;
          2 2 1 1;
          3 3 1 1];

a=addrule(a,rulelist);
showrule(a)                      %Show fuzzy rule base

a1=setfis(a,'DefuzzMethod','centroid');  %Defuzzy
a1=setfis(a,'DefuzzMethod','lom');  %Defuzzy
writefis(a1,'fsmc');              %Save fuzzy system as "fsmc.fis"
a2=readfis('fsmc');
ruleview(a2);

figure(1);
plotmf(a,'input',1);
figure(2);
plotmf(a,'output',1);
```

(2) Simulink main program: chap9_1sim.mdl



(3) S-function of 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 {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      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[];
function sys=mdlOutputs(t,x,u)
persistent a2

if t==0
   a2=readfis('fsmc.fis');
end
xd=sin(t);
dxd=cos(t);
ddxd=-sin(t);

e=u(1);
de=u(2);

c=25;
s=c*e+de;

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

g=9.8;mc=1.0;m=0.1;l=0.5;
S=l*(4/3-m*(cos(x1))^2/(mc+m));
fx=g*sin(x1)-m*l*x2^2*cos(x1)*sin(x1)/(mc+m);
fx=fx/S;
gx=cos(x1)/(mc+m);
gx=gx/S;

ueq=1/gx*(c*de+ddxd-fx);
D=5;
xite=D+0.15;
us=1/gx*xite*sign(s);

M=2;
if M==1      % Using conventional equavalent sliding mode control
   Mu=1.0;
elseif M==2
    Mu=evalfis([s],a2);  % Using fuzzy equavalent sliding mode control
```

```
end
ut=ueq+Mu*us;

sys(1)=ut;
sys(2)=Mu;
```

(4) S-function of the plant: chap9_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 = 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;
%%%%%%%%
bi=0.50;ci=5;
dt=5*exp(-(t-ci)^2/(2*bi^2));  %rbf_func.m
%%%%%%%%

sys(1)=x(2);
sys(2)=fx+gx*u+dt;
function sys=mdlOutputs(t,x,u)
bi=0.50;ci=5;
dt=5*exp(-(t-ci)^2/(2*bi^2));  %rbf_func.m

sys(1)=x(1);
sys(2)=dt;
```

(5) Plot program: chap9_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','tracking signal');
subplot(212);
plot(t,y(:,1)-y(:,2),'linewidth',2);
xlabel('time(s)');ylabel('Position tracking error');

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

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

figure(4);
plot(t,miu(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Membership function degree');
```

9.2 Sliding Mode Control Based on Fuzzy Switch-Gain Regulation

Fuzzy rule is adopted and the switch gain is estimated effectively according to the attaining condition of sliding mode. Thus, the disturbances and chattering phenomenon are reduced using switch gain.

9.2.1 System Description

Consider the uncertain system as follows:

$$\ddot{\theta} = f(\theta,\dot{\theta}) + b\,(u(t) + E(t)) \tag{9.5}$$

where $u(t) \in \mathbf{R}$ is control input $f(\theta,\dot{\theta})$ is known, $b > 0$, $E(t)$ is the unknown disturbance.

9.2.2 Design of Sliding Mode Controller

Select sliding mode variable as

$$s = \dot{e} + ce, \quad c > 0 \tag{9.6}$$

where e is the tracking error and $e = \theta_d - \theta$, θ_d is the desired trajectory.

Sliding mode controller is designed as

$$u = \frac{1}{b}(-f(\theta) + \ddot{\theta}_d + c\dot{e} + K(t)\,\mathrm{sgn}(s)) \tag{9.7}$$

and select

$$K(t) = \max |E(t)| + \eta \tag{9.8}$$

where $\eta > 0$.

Analysis of stability:

Select the Lyapunov function as

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

Therefore, we have

$$\dot{V} = s\dot{s} = s(\ddot{e} + c\dot{e}) = s(\ddot{\theta}_d - \ddot{\theta} + c\dot{e}) = s(\ddot{\theta}_d - f(\theta) - bu - E(t) + c\dot{e})$$

From Eq. (9.7), we have

$$\dot{V} = s(-K(t)\mathrm{sgn}(s) - E(t)) = -K(t)|s| - E(t)s \leqslant -\eta |s|$$

In Eq. (9.7), $K(t)$ is used to compensate the uncertainty $E(t)$ and guarantee the existing condition of the sliding mode. The switch gain $K(t)$ brings out chattering phenomenon. If $E(t)$ is time variant, then, in order to bring out the chattering phenomenon, $K(t)$ should be the time variant.

Using fuzzy rule and designing the fuzzy system, the estimation of $K(t)$ can be obtained.

9.2.3 Design of Fuzzy System

Existing condition of the sliding mode:

$$s\dot{s} < 0 \tag{9.9}$$

If Eq. (9.9) is satisfied, then the system states are on the sliding surface when the system states attain on the sliding surface. The selection of $K(t)$ must remove the effect of uncertainties in order to make system state attain sliding surface.

In order to guarantee the existence condition of sliding mode, fuzzy rule is given as follows:

If $s\dot{s} > 0$ then $K(t)$ should be increased;

If $s\dot{s} < 0$ then $K(t)$ should be decreased.

From the rule above, the fuzzy system of the relation between $s\dot{s}$ and $\Delta K(t)$ can be designed. In this system, $s\dot{s}$ is the input, and $\Delta K(t)$ is the output. The fuzzy sets of the input and output are defined respectively as follows:

$$s\dot{s} = \{NB \quad NM \quad ZO \quad PM \quad PB\}$$
$$\Delta K = \{NB \quad NM \quad ZO \quad PM \quad PB\}$$

where NB is negative and large, NM is negative and mid, ZO is zero, PM is the right mid, PB is positive and large.

The membership functions of the input and output of fuzzy system are shown in Fig. 9.9 and Fig. 9.10 respectively.

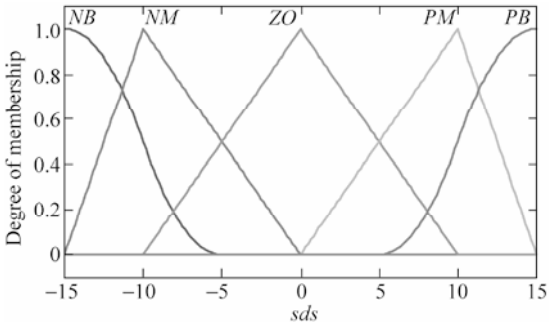


Figure 9.9 The membership function of fuzzy input

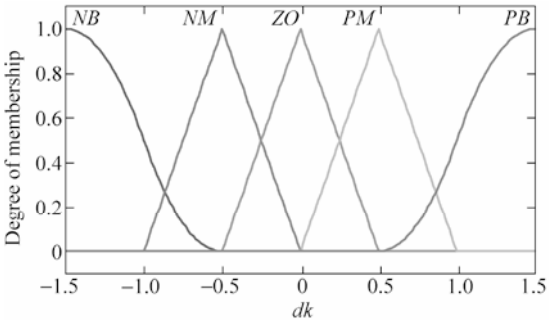


Figure 9.10 The membership function of fuzzy output

Fuzzy rule is selected as:

R1: IF $s\dot{s}$ is PB THEN ΔK is PB

R2: IF $s\dot{s}$ is PM THEN ΔK is PM

R3: IF $s\dot{s}$ is ZO THEN ΔK is ZO

R4: IF $s\dot{s}$ is NM THEN ΔK is NM

R5: IF $s\dot{s}$ is NB THEN ΔK is NB

Using integral method, the supper bound of $\hat{K}(t)$ is estimated:

$$\hat{K}(t) = G\int_0^t \Delta K \mathrm{d}t \tag{9.10}$$

where G is proportionality coefficient and is decided according to the experiences.

$\hat{K}(t)$ is used in Eq. (9.7) instead of $K(t)$, therefore, the controller is designed as:

$$u = \frac{1}{b}(-f(\theta) + \ddot{\theta}_{\mathrm{d}} + c\dot{e} + \hat{K}(t)\mathrm{sgn}(s)) \tag{9.11}$$

Replace Eq. (9.7) with Eq. (9.11), we have

$$\dot{V} = s(-K(t)\mathrm{sgn}(s) - E(t)) = -K(t)\,|\,s\,| - E(t)s \leqslant -\eta\,|\,s\,|$$

9.2.4 Simulation Example

Consider the system as follows:

$$\ddot{\theta} = f(\theta) + b(u(t) + E(t))$$

where $f(\theta) = -25\dot{\theta},\ b = 133$. The uncertainty $E(t)$ is the form of Gauss function:

$$E(t) = 200\exp\left(-\frac{(t-c_i)^2}{2b_i^2}\right)$$

Select $b_i = 0.50,\ c_i = 5.0,\ \eta = 1.0$. Therefore, the switch gain of the controller is $\hat{K}(t) = \max(|\,E(t)\,|) + \eta = 201$. $E(t)$ is shown in Fig.9.11.

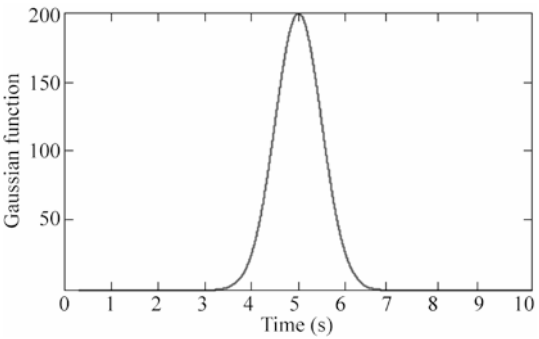


Figure 9.11 The uncertainty $E(t)$ of Gaussian function

Program of Gaussian function: chap4_3func.m

Desired position trajectory is $\theta_{\mathrm{d}} = \sin(2\pi t)$. Fuzzy system is established using S-function program chap4_3rule.m, and the membership functions are shown in Fig. 9.9 and Fig. 9.10.

Firstly, $M=2$ is selected, the controller is given in Eq. (9.11), and $G=400$, $c=150$. The simulation results are shown in Fig. 9.12 – Fig. 9.14. $M=1$ is selected, the controller is given in Eq. (9.7), $D=200$, and $c=150$. The simulation results are shown in Fig. 9.15 and Fig. 9.16.

From the simulation, we can find that the fuzzy sliding mode control method based fuzzy rule can reduce effectively disturbances by switch gain.

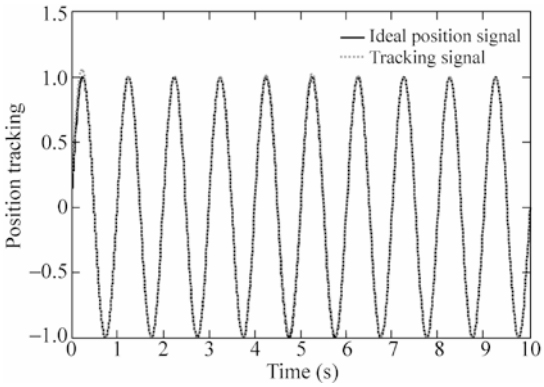


Figure 9.12 Position tracking using controller with Eq. (9.11)

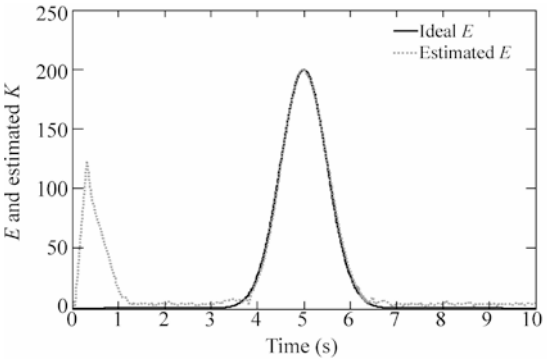


Figure 9.13 $E(t)$ and its $\hat{K}(t)$

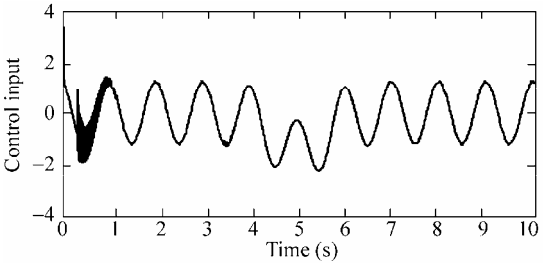


Figure 9.14 Control input with Eq. (9.11)

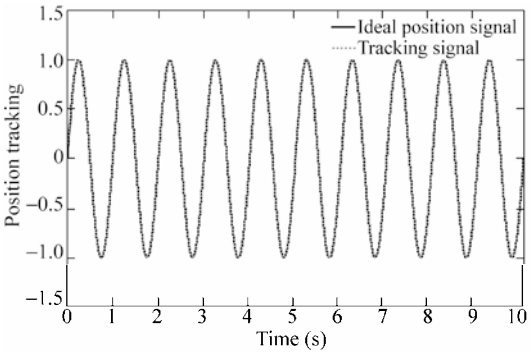


Figure 9.15 Position tracking using controller with Eq. (9.7)

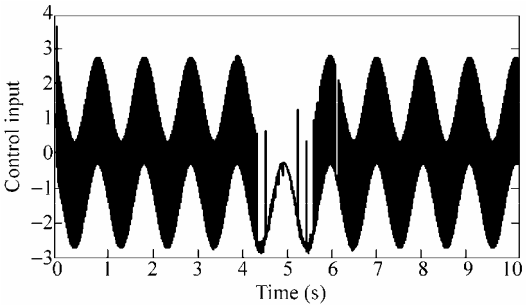
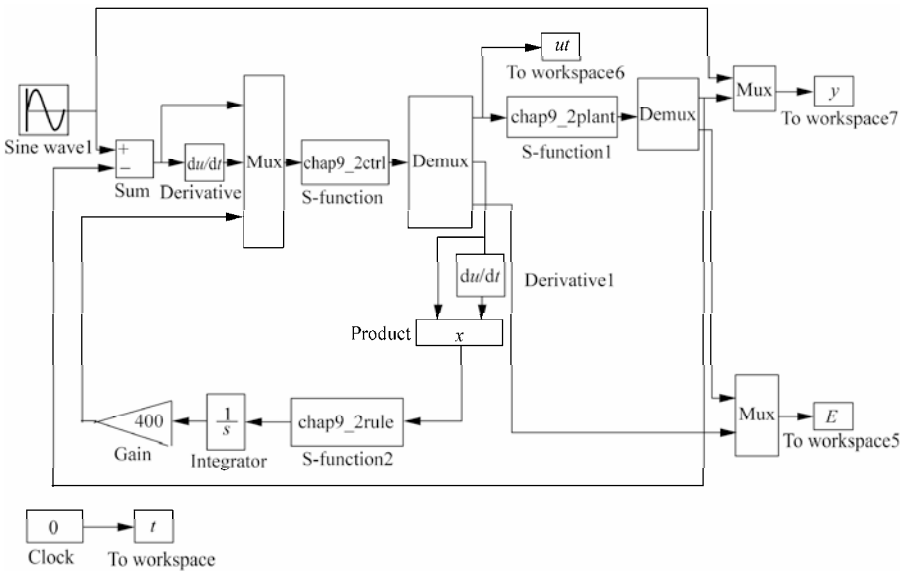


Figure 9.16 Control input with Eq. (9.7)

Simulation programs:

(1) Simulink main program: chap9_2sim.mdl



(2) S-function of 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     = 3;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[];
function sys=mdlOutputs(t,x,u)
persistent s0
e=u(1);
de=u(2);

c=150;
thd=sin(2*pi*t);
dthd=2*pi*cos(2*pi*t);
ddthd=-(2*pi)^2*sin(2*pi*t);

x1=thd-e;
x2=dthd-de;

fx=-25*x2;b=133;

s=c*e+de;

D=200;xite=1.0;

M=2;
if M==1
  K=D+xite;
elseif M==2   %Estimation for K with fuzzy
  K=abs(u(3))+xite;
end
```

```
ut=1/b*(-fx+ddthd+c*de+K*sign(s));

sys(1)=ut;
sys(2)=s;
sys(3)=K;
```

(3) S-function of 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  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.15,0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
%bi=0.05;ci=5;
bi=0.5;ci=5;
dt=200*exp(-(t-ci)^2/(2*bi^2));   %rbf_func.m
%dt=0;

sys(1)=x(2);
sys(2)=-25*x(2)+133*u+dt;
function sys=mdlOutputs(t,x,u)
%bi=0.05;ci=5;
bi=0.5;ci=5;
dt=200*exp(-(t-ci)^2/(2*bi^2));   %rbf_func.m
%dt=0;

sys(1)=x(1);
sys(2)=dt;
```

(4) S-function of fuzzy system: chap9_2rule.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      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[];
function sys=mdlOutputs(t,x,u)
warning off;
persistent a1
if t==0
  a1=readfis('smc_fuzz');
end

sys(1)=evalfis([u(1)],a1);
```

(5) Plot program: chap9_2plot.m

```
close all;

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

figure(2);
plot(t,E(:,1),'k',t,E(:,2),'r:','linewidth',2);
xlabel('time(s)');ylabel('E and estimated K');
legend('Ideal E','estimated E');

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

9.3 Sliding Mode Control Based on Fuzzy System Approximation

9.3.1 Problem Statement

Consider a second-order nonlinear system as follows:

$$\ddot{\theta} = f(\theta,\dot{\theta}) + g(\theta,\dot{\theta})u + d(t) \tag{9.12}$$

where f and g are all nonlinear functions, $u \in \mathbf{R}$ and is the input control, $d(t)$ is the outer disturbance, and $|d(t)| \leqslant D$.

Let the desired output be θ_{d}, and denote

$$e = \theta_{\mathrm{d}} - \theta$$

Design the sliding mode function as

$$s = \dot{e} + ce \tag{9.13}$$

where $c > 0$, then

$$\dot{s} = \ddot{e} + c\dot{e} = \ddot{\theta}_{\mathrm{d}} - \ddot{\theta} + c\dot{e} = \ddot{\theta}_{\mathrm{d}} - f - gu - d(t) + c\dot{e} \tag{9.14}$$

If f and g are known, we can design control law as

$$u = \frac{1}{g}[-f + \ddot{\theta}_{\mathrm{d}} + c\dot{e} + \eta\,\mathrm{sgn}(s)] \tag{9.15}$$

Then Eq. (9.14) becomes

$$\dot{s} = \ddot{e} + c\dot{e} = \ddot{\theta}_{\mathrm{d}} - \ddot{\theta} + c\dot{e} = \ddot{\theta}_{\mathrm{d}} - f - gu - d(t) + c\dot{e} = -\eta\,\mathrm{sgn}(s) - d(t)$$

Therefore, if $\eta \geqslant D$, we have

$$s\dot{s} = -\eta\,|s| - s \cdot d(t) \leqslant 0$$

If $f(x)$ is unknown, we should estimate $f(x)$ by some algorithms. In the following, we will simply recall fuzzy systems approximate uncertain item $f(x)$.

9.3.2 Controller Design Based on Fuzzy System

9.3.2.1 Uncertainty Approximation Using Fuzzy System

If $f(x)$ is unknown, we can replace $f(x)$ with the fuzzy estimation $\hat{f}(x)$ to realize feedback control[1]. The universal approximation theorem is described as

Step one: For x_i ($i=1,2,\cdots,n$), define the fuzzy sets $A_i^{l_i}$, $l_i=1,2,\cdots,p_i$.

Step two: Adopt $\prod_{i=1}^{n} p_i$ fuzzy rules to construct fuzzy system $\hat{f}(x\,|\,\theta_f)$:

$$R^{(j)}\text{: If } x_1 \text{ is } A_1^{l_1} \text{ and } \ldots \text{ and } x_n \text{ is } A_1^{l_n} \text{ then } \hat{f} \text{ is } E^{l_1,\cdots,l_n} \tag{9.16}$$

where $l_i=1,2,\cdots,p_i$, $i=1,2,\cdots,n$.

Therefore, the output of fuzzy system is

$$\hat{f}(\boldsymbol{x}\,|\,\theta_f)=\frac{\sum_{l_1=1}^{p_1}\cdots\sum_{l_n=1}^{p_n}\overline{y}_f^{l_1,\cdots,l_n}\left(\prod_{i=1}^{n}\mu_{A_i^{l_i}}(x_i)\right)}{\sum_{l_1=1}^{p_1}\cdots\sum_{l_n=1}^{p_n}\left(\prod_{i=1}^{n}\mu_{A_i^{l_i}}(x_i)\right)} \tag{9.17}$$

where $\mu_{A_i^j}(x_i)$ is the membership function of x_i. All the states are required known. Moreover, if there are noises in the measurement output $y=x_1$, the computation of $\mu_{A_i^j}(x_i)$ is affected seriously, therefore, fuzzy system is contaminated.

Let $\overline{y}_f^{l_1,\cdots,l_n}$ be a free parameter and be put in the set $\hat{\boldsymbol{\theta}}_f \in \mathbf{R}^{\prod_{i=1}^{n} p_i}$. Column vector $\boldsymbol{\xi}(\boldsymbol{x})$ is introduced and Eq. (9.17) can be written as:

$$\hat{f}(\boldsymbol{x}\,|\,\boldsymbol{\theta}_f)=\hat{\boldsymbol{\theta}}_f^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}) \tag{9.18}$$

where $\boldsymbol{\xi}(\boldsymbol{x})$ is the $\prod_{i=1}^{n} p_i$-dimensional column vector, and $l_1,\cdots,l_n$ elements are respectively

$$\xi_{l_1,\cdots,l_n}(x)=\frac{\prod_{i=1}^{n}\mu_{A_i^{l_i}}(x_i)}{\sum_{l_1=1}^{p_1}\cdots\sum_{l_n=1}^{p_n}\left(\prod_{i=1}^{n}\mu_{A_i^{l_i}}(x_i)\right)} \tag{9.19}$$

The membership functions are needed to be selected according to experiences. Moreover, all the states must be known.

9.3.2.2 Design of Adaptive Fuzzy Sliding Mode Controller

Suppose the optimal parameter as

$$\theta_f^{*}=\arg\min_{\theta_f\in\Omega_f}\left(\sup_{x\in\mathrm{R}^n}|\,\hat{f}(\boldsymbol{x}\,|\,\theta_f)-f(\boldsymbol{x})\,|\right)$$

where Ω_f is the set of θ_f, i.e. $\theta_f\in\Omega_f$.

The term f can be expressed as

$$f = \boldsymbol{\theta}_f^{*\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) + \varepsilon \tag{9.20}$$

where x is the input signal of the fuzzy system, where $\boldsymbol{\xi}(\boldsymbol{x})$ is the fuzzy vector, ε is approximation error of fuzzy system, and $\varepsilon \leqslant \varepsilon_{\mathrm{N}}$.

The fuzzy system is used to approximate f. The fuzzy system input is selected as $\boldsymbol{x} = [e \quad \dot{e}]^{\mathrm{T}}$, and the output of the fuzzy system is

$$\hat{f}(\boldsymbol{x} \mid \boldsymbol{\theta}_f) = \hat{\boldsymbol{\theta}}_f^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) \tag{9.21}$$

The control input Eq. (9.15) is written as

$$u = \frac{1}{g}(-\hat{f} + \ddot{\theta}_{\mathrm{d}} + c\dot{e} + \eta \operatorname{sgn}(s)) \tag{9.22}$$

Submitting Eqs. (9.22) to (9.14), we have

$$\begin{aligned} \dot{s} &= \ddot{\theta}_{\mathrm{d}} - f - gu - d(t) + c\dot{e} = \ddot{\theta}_{\mathrm{d}} - f - (-\hat{f} + \ddot{\theta}_{\mathrm{d}} + c\dot{e} + \eta \operatorname{sgn}(s)) - d(t) + c\dot{e} \\ &= -f + \hat{f} - \eta \operatorname{sgn}(s) - d(t) = -\tilde{f} - d(t) - \eta \operatorname{sgn}(s) \end{aligned} \tag{9.23}$$

Since

$$\tilde{f} = f - \hat{f} = f = \boldsymbol{\theta}_f^{*\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) + \varepsilon - \hat{\boldsymbol{\theta}}_f^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) = \tilde{\boldsymbol{\theta}}_f^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) + \varepsilon \tag{9.24}$$

where $\tilde{\boldsymbol{\theta}}_f = \boldsymbol{\theta}_f^* - \hat{\boldsymbol{\theta}}_f$.

Define the Lyapunov function as

$$L = \frac{1}{2} s^2 + \frac{1}{2} \gamma \tilde{\boldsymbol{\theta}}_f^{\mathrm{T}} \tilde{\boldsymbol{\theta}}_f$$

where $\gamma > 0$.

Derivative L, and from Eqs. (9.23) and (9.24), we have

$$\begin{aligned} \dot{L} &= s\dot{s} + \gamma \tilde{\boldsymbol{\theta}}_f^{\mathrm{T}} \dot{\tilde{\boldsymbol{\theta}}}_f = s(-\tilde{f} - d(t) - \eta \operatorname{sgn}(s)) - \gamma \tilde{\boldsymbol{\theta}}_f^{\mathrm{T}} \dot{\hat{\boldsymbol{\theta}}}_f \\ &= s(-\tilde{\boldsymbol{\theta}}_f^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) - \varepsilon - d(t) - \eta \operatorname{sgn}(s)) - \gamma \tilde{\boldsymbol{\theta}}_f^{\mathrm{T}} \dot{\hat{\boldsymbol{\theta}}}_f \\ &= -\tilde{\boldsymbol{\theta}}_f^{\mathrm{T}} (s\boldsymbol{\xi}(\boldsymbol{x}) + \gamma \dot{\hat{\boldsymbol{\theta}}}_f) - s(\varepsilon + d(t) + \eta \operatorname{sgn}(s)) \end{aligned}$$

Let the adaptive rule be

$$\dot{\hat{\boldsymbol{\theta}}}_f = -\frac{1}{\gamma} s\boldsymbol{\xi}(\boldsymbol{x}) \tag{9.25}$$

Then

$$\dot{L} = -s(\varepsilon + d(t) + \eta \operatorname{sgn}(s)) = -s(\varepsilon + d(t)) - \eta \mid s \mid$$

Due to the approximation error ε is sufficiently small, design $\eta \geqslant \varepsilon_{\mathrm{N}} + D$, we can obtain approximately $\dot{L} \leqslant 0$.

9.3.3 Simulation Example

Consider the following inverted pendulum:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{g \sin x_1 - mlx_2^2 \cos x_1 \sin x_1 / (m_{\mathrm{c}} + m)}{l(4/3 - m\cos^2 x_1 / (m_{\mathrm{c}} + m))} + \dfrac{\cos x_1 / (m_{\mathrm{c}} + m)}{l(4/3 - m\cos^2 x_1 / (m_{\mathrm{c}} + m))} u \end{cases}$$

where x_1 and x_2 are the swing angle and swing rate respectively. $g = 9.8\ \mathrm{m/s^2}$, $m_{\mathrm{c}} = 1\ \mathrm{kg}$ is the vehicle mass, $m = 0.10$ is the mass of the pendulum. $l = 0.50$ is one half of the pendulum length, and u is the control input.

W select the following five membership functions as:

$$\mu_{\mathrm{NM}}(x_i) = \exp[-((x_i + \pi/6)/(\pi/24))^2], \mu_{\mathrm{NS}}(x_i) = \exp[-((x_i + \pi/12)/(\pi/24))^2],$$
$$\mu_{\mathrm{Z}}(x_i) = \exp[-(x_i/(\pi/24))^2], \mu_{\mathrm{PS}}(x_i) = \exp[-((x_i - \pi/12)/(\pi/24))^2],$$
$$\mu_{\mathrm{PM}}(x_i) = \exp[-((x_i - \pi/6)/(\pi/24))^2]$$

The membership functions curves are shown in Fig. 9.17.

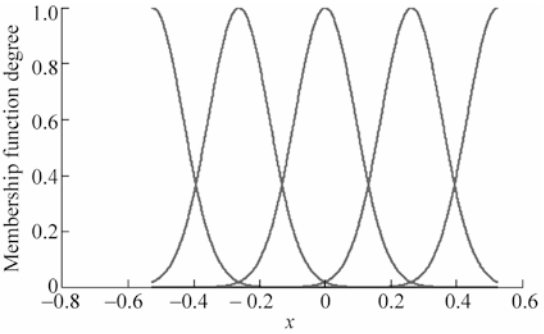


Figure 9.17 The membership function of x_i

Choosing $x_1 = \theta$, The desired trajectory is $\theta_{\mathrm{d}}(t) = 0.1 \sin t$. The initial state is $[\pi/60 \quad 0]$, $\theta_f(0) = 0.1$. We adapt control law as Eq. (9.22) and adaptive law as Eq. (9.25), choosing $\eta = 0.1$, $k_1 = 20$, $k_2 = 10$ and adaptive parameter $\gamma = 0.05$.

The curves of position tracking and uncertainty approximation are shown in Fig. 9.18 – Fig. 9.20.

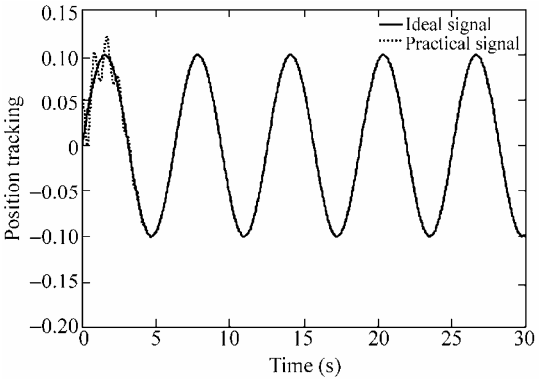


Figure 9.18 Position tracking

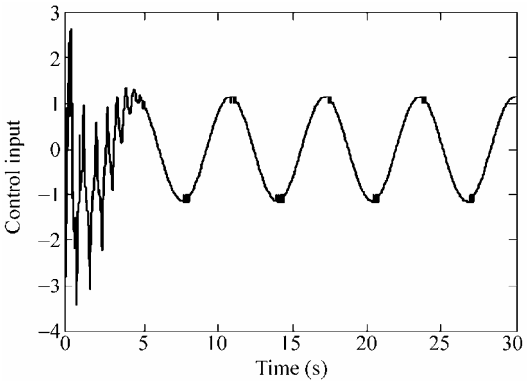


Figure 9.19 Control input

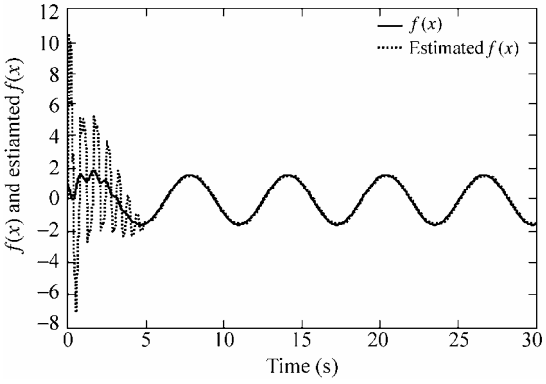
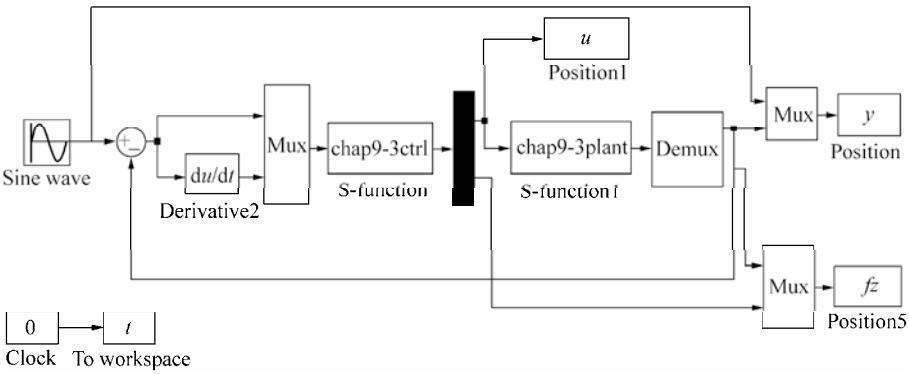


Figure 9.20 $f(x)$ and $\hat{f}(x)$

Simulation programs:

(1) Main Simulink program: chap9_3sim.mdl



(2) Control law program: chap9_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
sizes = simsizes;
sizes.NumContStates  = 25;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [zeros(25,1)];
str = [];
ts  = [];
function sys=mdlDerivatives(t,x,u)
gama=0.005;
r=0.1*sin(t);
dr=0.1*cos(t);
ddr=-0.1*sin(t);
```

```
e=u(1);
de=u(2);
n=25;
s=n*e+de;

x1=e;
x2=de;

for i=1:1:25
    thtaf(i,1)=x(i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FS1=0;
for l1=1:1:5
   gs1=-[(x1+pi/6-(l1-1)*pi/12)/(pi/24)]^2;
    u1(l1)=exp(gs1);
end

for l2=1:1:5
   gs2=-[(x2+pi/6-(l2-1)*pi/12)/(pi/24)]^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;
for i=1:1:25
    sys(i)=-1/gama*s*FS(i);
end
function sys=mdlOutputs(t,x,u)
r=0.1*sin(t);
dr=0.1*cos(t);
ddr=-0.1*sin(t);

e=u(1);
de=u(2);
n=25;
s=n*e+de;

x1=e;
x2=de;
for i=1:1:25
    thtaf(i,1)=x(i);
end
```

```
FS1=0;
for l1=1:1:5
   gs1=-[(x1+pi/6-(l1-1)*pi/12)/(pi/24)]^2;
    u1(l1)=exp(gs1);
end

for l2=1:1:5
   gs2=-[(x2+pi/6-(l2-1)*pi/12)/(pi/24)]^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;

fxp=thtaf'*FS';

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;

if t<=1.0
    xite=1.0;
else
    xite=0.10;
end

ut=1/gx*(-fxp+ddr+n*de+xite*sign(s));
sys(1)=ut;
sys(2)=fxp;
```

(3) Membership function program: chap9_3mf.m

```
clear all;
close all;

L1=-pi/6;
L2=pi/6;
L=L2-L1;

T=L*1/1000;

x=L1:T:L2;
figure(1);
for i=1:1:5
```

```
   gs=-[(x+pi/6-(i-1)*pi/12)/(pi/24)].^2;
    u=exp(gs);
    hold on;
    plot(x,u);
end
xlabel('x');ylabel('Membership function degree');
```

(4) Plant program: chap9_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*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)=fx;
```

(5) Plot program: chap9_3plot.m

```
close all;

figure(1);
plot(t,y(:,1),'k',t,y(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking');
legend('ideal signal','practical signal');

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

figure(3);
plot(t,fx(:,1),'k',t,fx(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('fx and estiamted fx');
legend('fx','estiamted fx');
```

9.4 Adaptive Fuzzy Control Based on Fuzzy Compensation for Manipulator

9.4.1 System Description

Dynamic equation of manipulator:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \boldsymbol{\tau} \tag{9.26}$$

where $\boldsymbol{H}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the matrix resulting from Coriolis and centrifugal forces, $\boldsymbol{G}(\boldsymbol{q})$ is the gravity. $\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ is the uncertainty generated by $\boldsymbol{F}_{\mathrm{r}}$, $\boldsymbol{\tau}$ is the control input, $\boldsymbol{\tau}_{\mathrm{d}}$ is the disturbance adding on the $\boldsymbol{\tau}$.

9.4.2 Control Based on Fuzzy Compensation

Suppose $\boldsymbol{H}(\boldsymbol{q})$, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ and $\boldsymbol{G}(\boldsymbol{q})$ are known, and all the states are measured.

Select sliding variable as:

$$\boldsymbol{s} = \dot{\tilde{\boldsymbol{q}}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{q}} \tag{9.27}$$

where $\boldsymbol{\Lambda}$ is positive-definite, $\tilde{\boldsymbol{q}}(t)$ is the tracking error.

Denote:

$$\dot{\boldsymbol{q}}_{\mathrm{r}}(t) = \dot{\boldsymbol{q}}_{\mathrm{d}}(t) - \boldsymbol{\Lambda}\tilde{\boldsymbol{q}}(t) \tag{9.28}$$

and select the Lyapunov function as

$$V(t) = \frac{1}{2}\left(\boldsymbol{s}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{s} + \sum_{i=1}^{n} \tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\Gamma}_i \tilde{\boldsymbol{\Theta}}_i \right) \tag{9.29}$$

where $\tilde{\boldsymbol{\Theta}}_i = \boldsymbol{\Theta}_i^* - \boldsymbol{\Theta}_i$, $\boldsymbol{\Theta}_i^*$ is the desired parameter, $\boldsymbol{\Gamma}_i > 0$.

Because $\boldsymbol{s} = \dot{\tilde{\boldsymbol{q}}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{q}} = \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{q}} = \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\mathrm{r}}$, we have

$$\begin{aligned} \boldsymbol{s} &= \dot{\tilde{\boldsymbol{q}}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{q}} = \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\mathrm{d}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{q}} = \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\mathrm{r}} \\ \boldsymbol{H}\dot{\boldsymbol{s}} &= \boldsymbol{H}\ddot{\boldsymbol{q}} - \boldsymbol{H}\ddot{\boldsymbol{q}}_{\mathrm{r}} = \boldsymbol{\tau} - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G} - \boldsymbol{F} - \boldsymbol{H}\ddot{\boldsymbol{q}}_{\mathrm{r}} \end{aligned}$$

Therefore,

$$\begin{aligned} \dot{V}(t) &= \boldsymbol{s}^{\mathrm{T}}\boldsymbol{H}\dot{\boldsymbol{s}} + \frac{1}{2}\boldsymbol{s}^{\mathrm{T}}\dot{\boldsymbol{H}}\boldsymbol{s} + \sum_{i=1}^{n} \tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\Gamma}_i \dot{\tilde{\boldsymbol{\Theta}}}_i \\ &= -\boldsymbol{s}^{\mathrm{T}}(-\boldsymbol{\tau} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{G} + \boldsymbol{F} + \boldsymbol{H}\ddot{\boldsymbol{q}}_{\mathrm{r}} - \boldsymbol{C}\boldsymbol{s}) + \sum_{i=1}^{n} \tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\Gamma}_i \dot{\tilde{\boldsymbol{\Theta}}}_i \\ &= -\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{H}\ddot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{C}\dot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{G} + \boldsymbol{F} - \boldsymbol{\tau}) + \sum_{i=1}^{n} \tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\Gamma}_i \dot{\tilde{\boldsymbol{\Theta}}}_i \end{aligned} \tag{9.30}$$

where $\boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is unknown nonlinear function. MIMO fuzzy system $\hat{\boldsymbol{F}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} | \boldsymbol{\Theta})$ is adopted to approximate to $\boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$.

Fuzzy adaptive sliding mode controller is designed as:

$$\boldsymbol{\tau} = \boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{G}(\boldsymbol{q}) + \hat{\boldsymbol{F}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} | \boldsymbol{\Theta}) - \boldsymbol{K}_{\mathrm{D}}\boldsymbol{s} - \boldsymbol{W}\,\mathrm{sgn}(\boldsymbol{s}) \tag{9.31}$$

where $\boldsymbol{W} = \mathrm{diag}\,[w_{\mathrm{m}_1}, w_{\mathrm{m}_2}, \cdots, w_{\mathrm{m}_n}]$, $w_{\mathrm{m}_i} \geqslant |\omega_i|$, $i = 1, 2, \cdots, n$, $\boldsymbol{K}_{\mathrm{D}} = \mathrm{diag}(K_i)$, $K_i > 0$, $i = 1, 2, \cdots, n$, and

$$\hat{\boldsymbol{F}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} | \boldsymbol{\Theta}) = \begin{bmatrix} \hat{F}_1(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} | \boldsymbol{\Theta}_1) \\ \hat{F}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} | \boldsymbol{\Theta}_2) \\ \vdots \\ \hat{F}_n(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} | \boldsymbol{\Theta}_n) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Theta}_1^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \\ \boldsymbol{\Theta}_2^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \\ \vdots \\ \boldsymbol{\Theta}_n^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \end{bmatrix} \tag{9.32}$$

Fuzzy approximating error is

$$\boldsymbol{\omega} = \boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) - \hat{\boldsymbol{F}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \mid \boldsymbol{\Theta}^*) \tag{9.33}$$

From Eqs. (9.31) and (9.30), we have

$$\begin{aligned}
\dot{V}(t) &= -\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) - \hat{\boldsymbol{F}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \mid \boldsymbol{\Theta}) + \boldsymbol{K}_{\mathrm{D}}\boldsymbol{s} + \boldsymbol{W}\,\mathrm{sgn}(\boldsymbol{s})) + \sum_{i=1}^{n} \tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\Gamma}_i \dot{\tilde{\boldsymbol{\Theta}}}_i \\
&= -\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) - \hat{\boldsymbol{F}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \mid \boldsymbol{\Theta}) + \hat{\boldsymbol{F}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \mid \boldsymbol{\Theta}^*) \\
&\quad - \hat{\boldsymbol{F}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \mid \boldsymbol{\Theta}^*) + \boldsymbol{K}_{\mathrm{D}}\boldsymbol{s} + \boldsymbol{W}\,\mathrm{sgn}(\boldsymbol{s})) + \sum_{i=1}^{n} \tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\Gamma}_i \dot{\tilde{\boldsymbol{\Theta}}}_i \\
&= -\boldsymbol{s}^{\mathrm{T}}(\tilde{\boldsymbol{\Theta}}^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) + \boldsymbol{\omega} + \boldsymbol{K}_{\mathrm{D}}\boldsymbol{s} + \boldsymbol{W}\,\mathrm{sgn}(\boldsymbol{s})) + \sum_{i=1}^{n} \tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\Gamma}_i \dot{\tilde{\boldsymbol{\Theta}}}_i \\
&= -\boldsymbol{s}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{D}}\boldsymbol{s} - \boldsymbol{s}^{\mathrm{T}}\boldsymbol{\omega} - \boldsymbol{W} \parallel \boldsymbol{s} \parallel + \sum_{i=1}^{n} (\tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\Gamma}_i \dot{\tilde{\boldsymbol{\Theta}}}_i - s_i \tilde{\boldsymbol{\Theta}}_i^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}))
\end{aligned}$$

where $\tilde{\boldsymbol{\Theta}} = \boldsymbol{\Theta}^* - \boldsymbol{\Theta}$, $\boldsymbol{\xi}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ is the fuzzy system.

The adaptive rule is

$$\dot{\boldsymbol{\Theta}}_i = -\boldsymbol{\Gamma}_i^{-1} s_i \boldsymbol{\xi}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}), \quad i = 1,2,\cdots,n \tag{9.34}$$

Therefore, we have

$$\dot{V}(t) = -\boldsymbol{s}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{D}}\boldsymbol{s} - \boldsymbol{s}^{\mathrm{T}}\boldsymbol{\omega} - \boldsymbol{W} \parallel \boldsymbol{s} \parallel \leqslant -\boldsymbol{s}^{\mathrm{T}}\boldsymbol{K}_{\mathrm{D}}\boldsymbol{s} \leqslant 0$$

Suppose the joint number of manipulator is n, and if MIMO fuzzy system $\hat{\boldsymbol{F}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \mid \boldsymbol{\Theta})$ is adopted to approximate to $\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$, then for each joint, the number of input variables is 3. If k membership functions are designed for each input variable, then the whole number of rules is k^{3n} [4].

For instance, the joint number of the manipulator is 2, the number of input variable is 3, there are 5 membership functions, then the whole rule number is $5^{3\times 2} = 5^6 = 15625$. The two many rules will bring out excessive computation. In order to decrease the number of fuzzy rules, independent design should be adopted with respect to $\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},t)$.

9.4.3 Control Based on Friction Compensation

When $\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ only consists of $\boldsymbol{F}_{\mathrm{r}}$, we can consider the case of fuzzy compensation with respect to friction. Because the friction is relative to velocity, the fuzzy system which approximates friction can be written as $\hat{\boldsymbol{F}}(\dot{\boldsymbol{q}} \mid \boldsymbol{\theta})$.

The method based on traditional fuzzy compensation, i.e., Eqs. (9.31) and (9.34), is adopted to design the controller. The fuzzy adaptive sliding mode controller is designed as:

$$\boldsymbol{\tau} = \boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_{\mathrm{r}} + \boldsymbol{G}(\boldsymbol{q}) + \hat{\boldsymbol{F}}(\dot{\boldsymbol{q}} \mid \boldsymbol{\theta}) - \boldsymbol{K}_{\mathrm{D}}\boldsymbol{s} - \boldsymbol{W}\,\mathrm{sgn}(\boldsymbol{s}) \tag{9.35}$$

The adaptive rule is

$$\dot{\boldsymbol{\theta}}_i = -\boldsymbol{\Gamma}_i^{-1} s_i \boldsymbol{\xi}(\dot{\boldsymbol{q}}),\ \ i = 1,2,\cdots,n \tag{9.36}$$

and the fuzzy system is

$$\hat{\boldsymbol{F}}(\dot{\boldsymbol{q}} \mid \boldsymbol{\theta}) = \begin{bmatrix} \hat{F}_1(\dot{\boldsymbol{q}}_1) \\ \hat{F}_2(\dot{\boldsymbol{q}}_2) \\ \vdots \\ \hat{F}_n(\dot{\boldsymbol{q}}_n) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_1^{\mathrm{T}}\boldsymbol{\xi}^1(\dot{\boldsymbol{q}}_1) \\ \boldsymbol{\theta}_2^{\mathrm{T}}\boldsymbol{\xi}^2(\dot{\boldsymbol{q}}_2) \\ \vdots \\ \boldsymbol{\theta}_n^{\mathrm{T}}\boldsymbol{\xi}^n(\dot{\boldsymbol{q}}_n) \end{bmatrix}$$

9.4.4 Simulation Example

The kinetic equation of dual-joint rigid manipulator is:

$$\begin{pmatrix} H_{11}(q_2) & H_{12}(q_2) \\ H_{21}(q_2) & H_{22}(q_2) \end{pmatrix}\begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + \begin{pmatrix} -C_{12}(q_2)\dot{q}_2 & -C_{12}(q_2)(\dot{q}_1 + \dot{q}_2) \\ C_{12}(q_2)\dot{q}_1 & 0 \end{pmatrix} \cdot \begin{pmatrix} g_1(q_1 + q_2)g \\ g_2(q_1 + q_2)g \end{pmatrix} + F(q,\dot{q},\ddot{q}) = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix}$$

where

$$\begin{aligned} H_{11}(q_2) &= (m_1 + m_2)r_1^2 + m_2 r_2^2 + 2m_2 r_1 r_2 \cos(q_2) \\ H_{12}(q_2) &= H_{21}(q_2) = m_2 r_2^2 + m_2 r_1 r_2 \cos(q_2) \\ H_{22}(q_2) &= m_2 r_2^2 \\ C_{12}(q_2) &= m_2 r_1 r_2 \sin(q_2) \end{aligned}$$

where m_1 and m_2 are the mass of link1 and link2, and r_1 and r_2 are the lengths of link1 and link2.

Let $\boldsymbol{y} = [q_1 \quad q_2]^{\mathrm{T}}$, $\boldsymbol{\tau} = [\tau_1 \quad \tau_2]^{\mathrm{T}}$, $\boldsymbol{x} = [q_1 \quad \dot{q}_1 \quad q_2 \quad \dot{q}_2]^{\mathrm{T}}$. The parameters: $r_1 = 1\,\mathrm{m}$, $r_2 = 0.8\,\mathrm{m}$, $m_1 = 1\,\mathrm{kg}$, $m_2 = 1.5\,\mathrm{kg}$.

The control object is to make the outputs q_1, q_2 track the desired trajectories $y_{\mathrm{d1}} = 0.3\sin t$ and $y_{\mathrm{d2}} = 0.3\sin t$ respectively. The membership function is defined as:

$$\mu_{A_i^l}(x_i) = \exp\left(-\left(\frac{x_i - \overline{x}_i^l}{\pi/24}\right)^2\right)$$

where $\overline{x}_i^l$ are $-\pi/6$, $-\pi/12$, 0, $\pi/12$, and $\pi/6$, respectively, $i=1,2,\cdots,5$, A_i^l is the fuzzy set including NB, NS, ZO, PS, PB belong to lth fuzzy rule.

The control based on friction compensation is used for the case with friction, and the controller parameters are: $\lambda_1=10$, $\lambda_2=10$, $\boldsymbol{K}_{\mathrm{D}}=20\boldsymbol{I}$, $\Gamma_1=\Gamma_2=0.0001$. The initial states are: $q_1(0)=q_2(0)=\dot{q}_1(0)=\dot{q}_2(0)=0$. The friction is $\boldsymbol{F}(\dot{\boldsymbol{q}})=\begin{bmatrix}15\dot{q}_1+6\mathrm{sgn}(\dot{q}_1)\\15\dot{q}_2+6\mathrm{sgn}(\dot{q}_2)\end{bmatrix}$, $\boldsymbol{W}=\mathrm{diag}[2,2]$. The fuzzy sliding mode controller is given in Eq. (9.35), and the adaptive rule is given in Eq. (9.36). The simulation results are shown in Fig. 9.21 – Fig. 9.23.

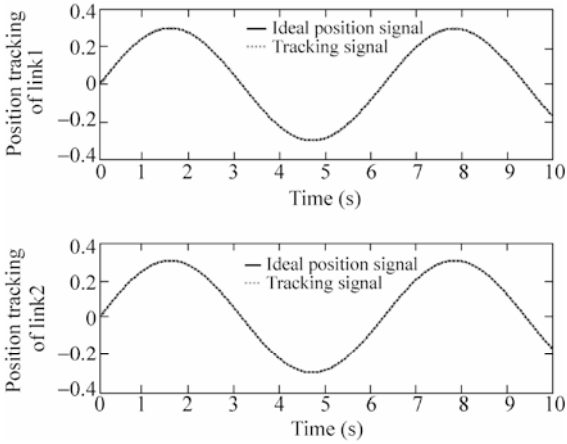


Figure 9.21 Position tracking of dual joints

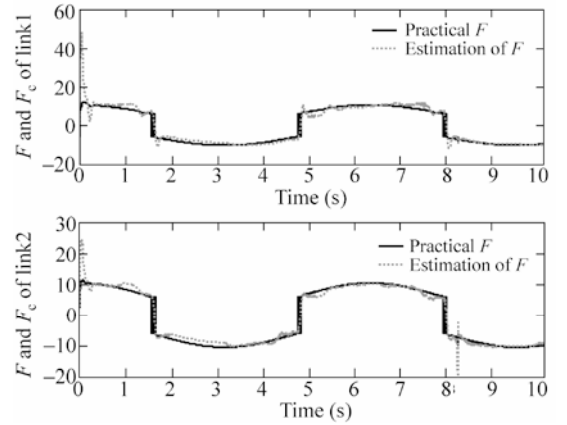


Figure 9.22 Friction and the compensation of dual joints

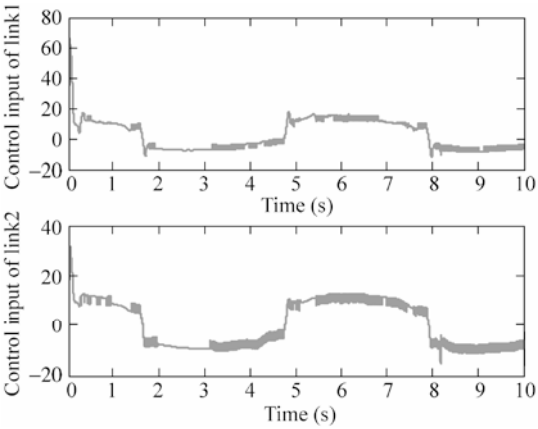
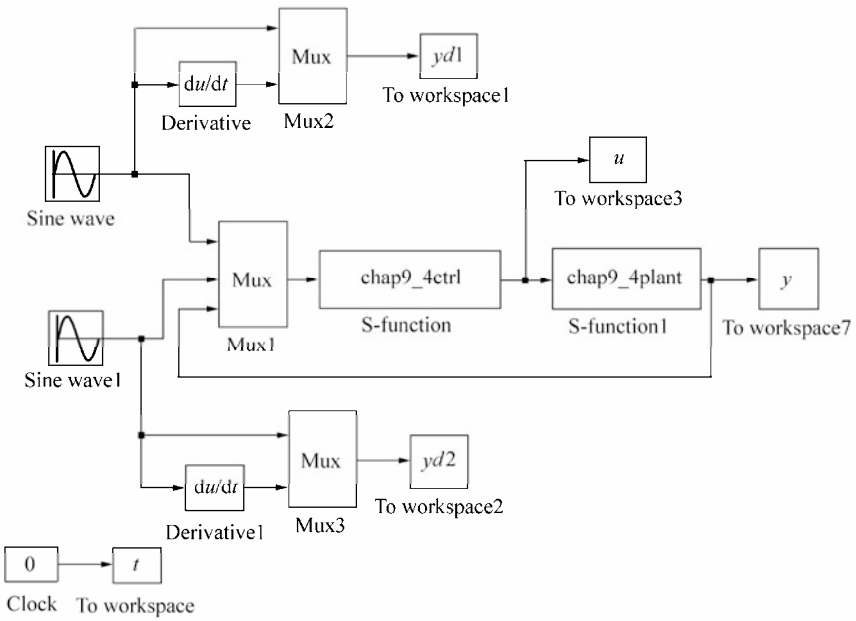


Figure 9.23 Control input of dual joints

Simulation program:
Control based on friction fuzzy compensation for manipulator

(1) Simulink main program: chap9_4sim.mdl



(2) S-function of controller: chap9_4ctrl.m

```
function [sys,x0,str,ts] = MIMO_Tong_s(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 nmn1 nmn2 Fai
nmn1=10;nmn2=10;
Fai=[nmn1 0;0 nmn2];
sizes = simsizes;
sizes.NumContStates  = 10;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 8;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [0.1*ones(10,1)];
str = [];
ts  = [];
function sys=mdlDerivatives(t,x,u)
global nmn1 nmn2 Fai
qd1=u(1);
qd2=u(2);
dqd1=0.3*cos(t);
dqd2=0.3*cos(t);
dqd=[dqd1 dqd2]';

ddqd1=-0.3*sin(t);
ddqd2=-0.3*sin(t);
ddqd=[ddqd1 ddqd2]';

q1=u(3);dq1=u(4);
q2=u(5);dq2=u(6);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fsd1=0;
for l1=1:1:5
    gs1=-[(dq1+pi/6-(l1-1)*pi/12)/(pi/24)]^2;
    u1(l1)=exp(gs1);
end
fsd2=0;
for l2=1:1:5
    gs2=-[(dq2+pi/6-(l2-1)*pi/12)/(pi/24)]^2;
    u2(l2)=exp(gs2);
end
```

```
for l1=1:1:5
    fsu1(l1)=u1(l1);
    fsd1=fsd1+u1(l1);
end
for l2=1:1:5
    fsu2(l2)=u2(l2);
    fsd2=fsd2+u2(l2);
end
fs1=fsu1/(fsd1+0.001);
fs2=fsu2/(fsd2+0.001);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e1=q1-qd1;
e2=q2-qd2;
e=[e1 e2]';
de1=dq1-dqd1;
de2=dq2-dqd2;
de=[de1 de2]';

s=de+Fai*e;
Gama1=0.0001;Gama2=0.0001;

S1=-1/Gama1*s(1)*fs1;
S2=-1/Gama2*s(2)*fs2;
for i=1:1:5
    sys(i)=S1(i);
end
for j=6:1:10
    sys(j)=S2(j-5);
end

function sys=mdlOutputs(t,x,u)
global nmn1 nmn2 Fai
q1=u(3);dq1=u(4);
q2=u(5);dq2=u(6);

r1=1;r2=0.8;
m1=1;m2=1.5;

H11=(m1+m2)*r1^2+m2*r2^2+2*m2*r1*r2*cos(q2);
H22=m2*r2^2;
H21=m2*r2^2+m2*r1*r2*cos(q2);
H12=H21;
H=[H11 H12;H21 H22];

C12=m2*r1*sin(q2);
C=[-C12*dq2 -C12*(dq1+dq2);C12*q1 0];

g1=(m1+m2)*r1*cos(q2)+m2*r2*cos(q1+q2);
g2=m2*r2*cos(q1+q2);
```

```
G=[g1;g2];

qd1=u(1);
qd2=u(2);
dqd1=0.3*cos(t);
dqd2=0.3*cos(t);
dqd=[dqd1 dqd2]';

ddqd1=-0.3*sin(t);
ddqd2=-0.3*sin(t);
ddqd=[ddqd1 ddqd2]';

e1=q1-qd1;
e2=q2-qd2;
e=[e1 e2]';
de1=dq1-dqd1;
de2=dq2-dqd2;
de=[de1 de2]';

s=de+Fai*e;

dqr=dqd-Fai*e;
ddqr=ddqd-Fai*de;

for i=1:1:5
    thta1(i,1)=x(i);
end
for i=1:1:5
    thta2(i,1)=x(i+5);
end

fsd1=0;
for l1=1:1:5
    gs1=-[(dq1+pi/6-(l1-1)*pi/12)/(pi/24)]^2;
    u1(l1)=exp(gs1);
end
fsd2=0;
for l2=1:1:5
    gs2=-[(dq2+pi/6-(l2-1)*pi/12)/(pi/24)]^2;
    u2(l2)=exp(gs2);
end

for l1=1:1:5
    fsu1(l1)=u1(l1);
    fsd1=fsd1+u1(l1);
end
for l2=1:1:5
    fsu2(l2)=u2(l2);
    fsd2=fsd2+u2(l2);
```

```
end
fs1=fsu1/(fsd1+0.001);
fs2=fsu2/(fsd2+0.001);

Fp(1)=thta1'*fs1';
Fp(2)=thta2'*fs2';

KD=20*eye(2);
W=[1.5 0;0 1.5];

tol=H*ddqr+C*dqr+G+1*Fp'-KD*s-W*sign(s);   %(4.134)

sys(1)=tol(1);
sys(2)=tol(2);
sys(3)=Fp(1);
sys(4)=Fp(2);
```

(3) Membership function program: chap9_4mf.m

```
clear all;
close all;

L1=-pi/6;
L2=pi/6;
L=L2-L1;

T=L*1/1000;

x=L1:T:L2;
figure(1);
for i=1:1:5
    gs=-[(x+pi/6-(i-1)*pi/12)/(pi/24)].^2;
    u=exp(gs);
    hold on;
    plot(x,u);
end
xlabel('x');ylabel('Membership function degree');
```

(4) S-function of the plant: chap9_4plant.m

```
function [sys,x0,str,ts]=MIMO_Tong_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  = 4;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 6;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0 0 0 0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
r1=1;r2=0.8;
m1=1;m2=1.5;

H11=(m1+m2)*r1^2+m2*r2^2+2*m2*r1*r2*cos(x(3));
H22=m2*r2^2;
H21=m2*r2^2+m2*r1*r2*cos(x(3));
H12=H21;
H=[H11 H12;H21 H22];

C12=m2*r1*sin(x(3));
C=[-C12*x(4) -C12*(x(2)+x(4));C12*x(1) 0];

g1=(m1+m2)*r1*cos(x(3))+m2*r2*cos(x(1)+x(3));
g2=m2*r2*cos(x(1)+x(3));
G=[g1;g2];

Fr=[15*x(2)+6*sign(x(2));15*x(4)+6*sign(x(4))];

tol=[u(1) u(2)]';
S=inv(H)*(tol-C*[x(2);x(4)]-G-Fr);

sys(1)=x(2);
sys(2)=S(1);
sys(3)=x(4);
sys(4)=S(2);
function sys=mdlOutputs(t,x,u)
Fr=[15*x(2)+6*sign(x(2));15*x(4)+6*sign(x(4))];

sys(1)=x(1);
sys(2)=x(2);
sys(3)=x(3);
sys(4)=x(4);
sys(5)=Fr(1);
sys(6)=Fr(2);
```

(5) Plot program: chap9_4plot.m

```
close all;

figure(1);
subplot(211);
plot(t,yd1(:,1),'k',t,y(:,1),'r:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking of link1');
legend('Ideal position signal','tracking signal');
subplot(212);
plot(t,yd2(:,1),'k',t,y(:,3),'r:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking of link2');
legend('Ideal position signal','tracking signal');

figure(2);
subplot(211);
plot(t,y(:,5),'k',t,u(:,3),'r:','linewidth',2);
xlabel('time(s)');ylabel('F and Fc of link1');
legend('Practical F','Estimation of F');
subplot(212);
plot(t,y(:,6),'k',t,u(:,4),'r:','linewidth',2);
xlabel('time(s)');ylabel('F and Fc of link2');
legend('Practical F','Estimation of F');

figure(3);
subplot(211);
plot(t,u(:,1),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input of Link1');
subplot(212);
plot(t,u(:,2),'r','linewidth',2);
xlabel('time(s)');ylabel('Control input of Link2');
```

9.5 Adaptive Sliding Mode Control Based on Switching Fuzzy

Using the adaptive fuzzy control method, the switching item in the sliding mode controller is approximated and the switching item is continued. Therefore, the chattering phenomenon can be reduced sufficiently[5].

9.5.1 Plant Description

Considering the following *n*-order SISO nonlinear system:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_n &= f(\boldsymbol{x},t) + g(\boldsymbol{x},t)u(t) + d(t) \end{aligned} \tag{9.37}$$

$$y = x_1$$

where f and g are the known nonlinear function, $\boldsymbol{x} \in \mathbf{R}^n$, $u \in \mathbf{R}$, $y \in \mathbf{R}$, $d(t)$ is the unknown disturbance, $|d(t)| \leqslant D$, $g(\boldsymbol{x},t) > 0$.

9.5.2 Design of Adaptive Fuzzy Sliding Mode Controller

The switching function is defined as

$$s = -(k_1 e + k_2 \dot{e} + \cdots + k_{n-1} e^{(n-1)} + e^{(n-1)}) = -k\boldsymbol{e} \tag{9.38}$$

where $\boldsymbol{e} = \boldsymbol{x}_\mathrm{d} - \boldsymbol{x} = [e \quad \dot{e} \quad \cdots \quad e^{(n-1)}]^\mathrm{T}$, $k_1, k_2, \cdots, k_{n-1}$ is satisfied with Hurwitzian stability condition.

The sliding mode controller is designed as

$$u(t) = \frac{1}{g(\boldsymbol{x},t)}\left(-f(\boldsymbol{x},t) + \sum_{i=1}^{n-1} k_i e^{(i)} + x_\mathrm{d}^{(n)} - u_\mathrm{sw}\right) \tag{9.39}$$

where $u_\mathrm{sw} = \eta \operatorname{sgn}(s)$, $\eta > D$.

From Eqs. (9.37) and (9.38), we get

$$\begin{aligned} \dot{s} &= -\sum_{i=1}^{n-1} k_i e^{(i)} + x^{(n)} - x_\mathrm{d}^{(n)} \\ &= -\sum_{i=1}^{n-1} k_i e^{(i)} + f(\boldsymbol{x},t) + g(\boldsymbol{x},t)u(t) + d(t) - x_\mathrm{d}^{(n)} \end{aligned} \tag{9.40}$$

And from Eq. (9.39), we have

$$\dot{s} = d(t) - \eta \operatorname{sgn}(s)$$

i.e.

$$s\dot{s} = d(t)s - \eta |s| \leqslant 0 \tag{9.41}$$

When d is relative large, the switching item η in controller (9.39) is large. This results in serious chattering phenomenon. Fuzzy system $\hat{h}$ is used to approximate $\eta \operatorname{sgn}(s)$. Therefore, the switching signal is weakened and the chattering phenomenon can be reduced.

Using product deduce, single-value fuzzy and center average fuzzy, the fuzzy output is $\hat{h}$. From Eq. (9.39), the controller is written as[5]

$$u(t) = \frac{1}{g(\boldsymbol{x},t)}\left(-f(\boldsymbol{x},t) + \sum_{i=1}^{n-1} k_i e^{(i)} + x_\mathrm{d}^{(n)} - \hat{h}(s)\right) \tag{9.42}$$

$$\hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}) = \boldsymbol{\theta}_{\mathrm{h}}^{\mathrm{T}} \boldsymbol{\phi}(s) \tag{9.43}$$

where $\hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}})$ is the fuzzy output of the universal approximation Eq. (9.18), $\boldsymbol{\phi}(s)$ is the fuzzy vector, vector $\boldsymbol{\theta}_{\mathrm{h}}^{\mathrm{T}}$ varies according to the adaptive rule. The ideal $\hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}})$ is

$$\hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}^{*}) = \eta \operatorname{sgn}(s) \tag{9.44}$$

where $\eta > D$.

The adaptive rule is:

$$\dot{\boldsymbol{\theta}}_{\mathrm{h}} = \gamma s \boldsymbol{\phi}(s) \tag{9.45}$$

where $\gamma > 0$.

Proof:

The optimization parameter is defined as:

$$\boldsymbol{\theta}_{\mathrm{h}}^{*} = \arg \min_{\boldsymbol{\theta}_{\mathrm{h}} \in \Omega_{\mathrm{h}}} [\sup \mid \hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}) - \eta \operatorname{sgn}(s) \mid] \tag{9.46}$$

where Ω_{h} is the set of θ_{h}.

Therefore, we have

$$\begin{aligned}
\dot{s} &= -\sum_{i=1}^{n-1} k_i e^{(i)} + x^{(n)} - x_{\mathrm{d}}^{(n)} \\
&= -\sum_{i=1}^{n-1} k_i e^{(i)} + f(x,t) + g(x,t)u(t) + d(t) - x_{\mathrm{d}}^{(n)} \\
&= -\hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}) + d(t) \\
&= -\hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}) + d(t) + \hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}^{*}) - \hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}^{*}) \\
&= \tilde{\boldsymbol{\theta}}_{\mathrm{h}}^{\mathrm{T}} \phi(s) + d(t) - \hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}^{*})
\end{aligned} \tag{9.47}$$

where $\tilde{\boldsymbol{\theta}}_{\mathrm{h}} = \boldsymbol{\theta}_{\mathrm{h}}^{*} - \boldsymbol{\theta}_{\mathrm{h}}$.

The Lyapunov function is selected as

$$V = \frac{1}{2}\left(s^2 + \frac{1}{\gamma} \tilde{\boldsymbol{\theta}}_{\mathrm{h}}^{\mathrm{T}} \tilde{\boldsymbol{\theta}}_{\mathrm{h}} \right) \tag{9.48}$$

Therefore,

$$\begin{aligned}
\dot{V} &= s\dot{s} + \frac{1}{\gamma} \tilde{\boldsymbol{\theta}}_{\mathrm{h}}^{\mathrm{T}} \dot{\tilde{\boldsymbol{\theta}}}_{\mathrm{h}} \\
&= s(\tilde{\boldsymbol{\theta}}_{\mathrm{h}}^{\mathrm{T}} \boldsymbol{\phi}(s) + d(t) - \hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}^{*})) + \frac{1}{\gamma} \tilde{\boldsymbol{\theta}}_{\mathrm{h}}^{\mathrm{T}} \dot{\tilde{\boldsymbol{\theta}}}_{\mathrm{h}}
\end{aligned}$$

$$= s\tilde{\boldsymbol{\theta}}_{\mathrm{h}}^{\mathrm{T}}\boldsymbol{\phi}(s) + \frac{1}{\gamma}\tilde{\boldsymbol{\theta}}_{\mathrm{h}}^{\mathrm{T}}\dot{\tilde{\boldsymbol{\theta}}}_{\mathrm{h}} + s(d(t) - \hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}^{*})) \tag{9.49}$$

Because

$$\hat{h}(s \mid \boldsymbol{\theta}_{\mathrm{h}}^{*}) = \eta\,\mathrm{sgn}(s)$$

we have

$$\dot{V} = \frac{1}{\gamma}\tilde{\boldsymbol{\theta}}_{\mathrm{h}}^{\mathrm{T}}(\gamma s\boldsymbol{\phi}(s) - \dot{\boldsymbol{\theta}}_{\mathrm{h}}) + sd(t) - \eta \mid s \mid \tag{9.50}$$

where $\dot{\tilde{\boldsymbol{\theta}}}_{\mathrm{h}} = -\dot{\boldsymbol{\theta}}_{\mathrm{h}}$.

From Eqs. (9.45) and (9.50), we get

$$\dot{V} = sd(t) - \eta \mid s \mid < 0$$

9.5.3 Simulation Example

The kinetic equation of the inverted pendulum is:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{g\sin x_1 - mlx_2^2\cos x_1 \sin x_1/(m_{\mathrm{c}}+m)}{l(4/3 - m\cos^2 x_1/(m_{\mathrm{c}}+m))} + \dfrac{\cos x_1/(m_{\mathrm{c}}+m)}{l(4/3 - m\cos^2 x_1/(m_{\mathrm{c}}+m))}u + d(t) \end{cases}$$

where x_1 and x_2 are the rolling angle and the rolling rate respectively. $g = 9.8\ \mathrm{m/s^2}$, m_c is the mass of the vehicle, $m_c = 1\,\mathrm{kg}$, m is the mass of the rolling pole, $m = 0.1\,\mathrm{kg}$, l is one half of the rolling pole, $l = 0.5\,\mathrm{m}$, u is the controller, and $d(t) = 10\sin t$.

The desired trajectory is $x_{\mathrm{d}}(t) = 0.1\sin t$, the switching function is $s = -k_1 e - \dot{e}$, $k_1 = 30$. The membership function of the switching function is defined as $\mu_N(s) = \dfrac{1}{1+\exp(5(s+3))}$, $\mu_Z(s) = \exp(-s^2)$, $\mu_P(s) = \dfrac{1}{1+\exp(5(s-3))}$.

Let $\boldsymbol{\theta}_{\mathrm{h}}^{\mathrm{T}}$ is a 3×1 vector, and the initial value of each argument in the vector is 0.10. Controller (9.42) and adaptive rule (9.45) are adopted. The initial state of the inverted pendulum is $[-\pi/60 \quad 0]$. The adaptive parameter is selected as $\gamma = 150$. In the program, *fsd*, *fsu* and *fs* denote the numerator, denominator and itself of $\phi(s)$ respectively. The simulation results are shown in Figs. 9.24 – 9.26.

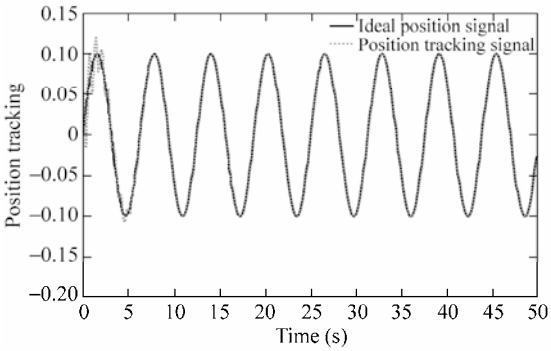


Figure 9.24 Tracking for sine position

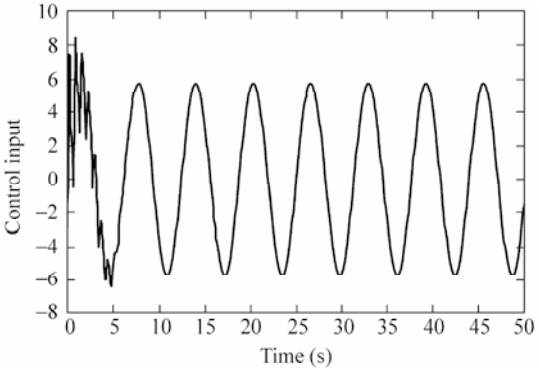


Figure 9.25 Control input

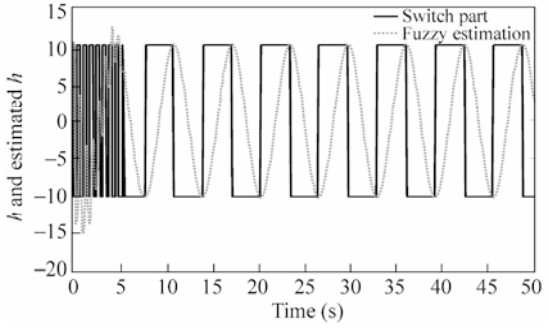
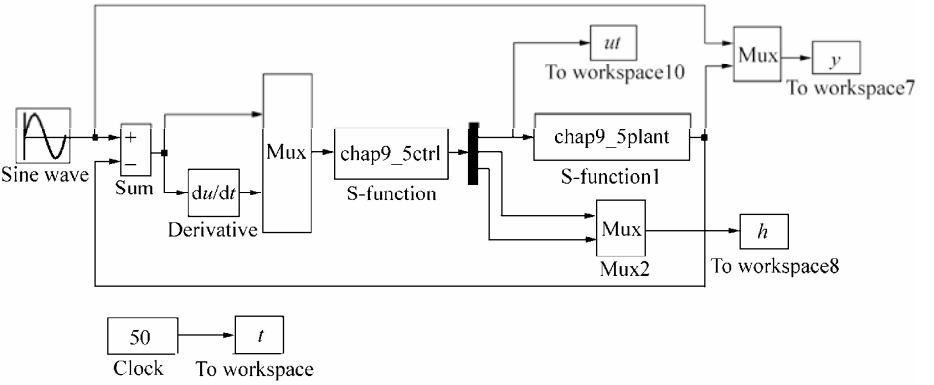


Figure 9.26 Control switching part and its fuzzy approximation

Simulation programs:

(1) Simulink main program: chap9_5sim.mdl



(2) S-function of controller: chap9_5ctrl.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      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [0.1*ones(3,1)];
str = [];
ts  = [];
function sys=mdlDerivatives(t,x,u)
xd=0.1*sin(t);
dxd=0.1*cos(t);
ddxd=-0.1*sin(t);

e=u(1);
```

```
de=u(2);
x1=xd-e;
x2=de-dxd;

k1=30;
s=-(k1*e+de);

for i=1:1:3
    thtah(i,1)=x(i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fsd=0;
gs=5*(s+3);
uh(1)=1/(1+exp(gs));

uh(2)=exp(-s^2);

gs=5*(s-3);
uh(3)=1/(1+exp(gs));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fsu=uh;
for i=1:1:3
   fsd=fsd+uh(i);
end
fs=fsu/(fsd+0.001);

gama=150;
S=gama*s*fs;
for j=1:1:3
    sys(j)=S(j);
end

function sys=mdlOutputs(t,x,u)
xd=0.1*sin(t);
dxd=0.1*cos(t);
ddxd=-0.1*sin(t);

e=u(1);
de=u(2);
x1=xd-e;
x2=de-dxd;

k1=30;
s=-(k1*e+de);

for i=1:1:3
    thtah(i,1)=x(i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fsd=0;
```

```
gs=5*(s+3);
uh(1)=1/(1+exp(gs));

uh(2)=exp(-s^2);

gs=5*(s-3);
uh(3)=1/(1+exp(gs));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fsu=uh;
for i=1:1:3
   fsd=fsd+uh(i);
end
fs=fsu/(fsd+0.001);
h=thtah'*fs';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g=9.8;mc=1.0;m=0.1;l=0.5;
S=l*(4/3-m*(cos(x1))^2/(mc+m));
fx=g*sin(x1)-m*l*x2^2*cos(x1)*sin(x1)/(mc+m);
fx=fx/S;
gx=cos(x1)/(mc+m);
gx=gx/S;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ut=1/gx*(-fx+ddxd-1*h+k1*de);

xite=10+0.01;
sys(1)=ut;
sys(2)=xite*sign(s);
sys(3)=h;
```

(3) S-function of the plant: chap9_5plant.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     = 1;
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=10*sin(t);
sys(1)=x(2);
sys(2)=fx+gx*u-dt;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
```

(4) Plot program: chap9_5plot.m

```
close all;

figure(1);
plot(t,y(:,1),'k',t,y(:,2),'r:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking');
legend('ideal position signal','position tracking signal');

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

figure(3);
plot(t,h(:,1),'k',t,h(:,2),'r:','linewidth',2);
xlabel('time(s)');ylabel('h and estiamted h');
legend('Switch part','fuzzy estination');
```

References

[1] Wang LX. A Course in Fuzzy System and Control, Prentice Hall, 1997

[2] Wang LX. Stable adaptive fuzzy control of nonlinear systems, IEEE Transactions on Fuzzy Systems, 1993, 1(2): 146－155

[3] Chen JY. Expert SMC-based fuzzy control with genetic algorithms, Journal of the Franklin Institute, 1999, 336: 589－610

[4] Yoo BK, Ham WC. Adaptive Control of Robot Manipulator Using Fuzzy Compensator. IEEE Transactions on Fuzzy Systems, 2000, 8(2): 186－199

[5] Wang J, Rad AB, Chan PT. Indirect adaptive fuzzy sliding mode control: Part I: fuzzy switching, Fuzzy Sets and systems, 2001, 122, 21－30