第3章 控制输入受限或输出受限条件下的滑模控制

在运动控制领域，由于执行机构的限制，只能提供有限的控制力矩，很容易产生控制输入受限问题，过大的控制律值难以实现，该问题在某种程度上将影响控制系统的稳定性和控制性能，甚至使得整个控制系统不稳定。如何在控制输入限制的条件下实现有效的控制算法设计，是一个很有意义的命题，这就是“控制输入饱和”问题。

3.1 控制输入受限条件下的滑模控制分析

3.1.1 基本原理

假设被控对象为

$$\begin{cases}\dot{x}_1 = x_2 \\ \dot{x}_2 = f(x,t) + bu + dt\end{cases} \tag{3.1}$$

其中，$b \neq 0$；dt 为干扰，$|dt| \leqslant D$；u 为受限的控制量。

取最大控制输入值为 $u_{\max}$，$u_{\max} > 0$，$\Delta u = u - v$，$u = \mathrm{sat}(v)$，控制输入饱和函数 $\mathrm{sat}(v)$ 表示为

$$\mathrm{sat}(v) = \begin{cases} u_{\max}, & v > u_{\max} \\ v, & |v| \leqslant u_{\max} \\ -u_{\max}, & v < -u_{\max} \end{cases} \tag{3.2}$$

控制输入饱和函数示意图如图 3.1 所示。

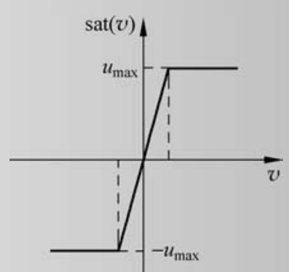


图 3.1 控制输入受限函数示意图

3.1.2 控制器设计与分析

通过定义辅助分析系统，采用输入饱和误差动态放大的方法，可实现一种基于控制输入抗饱和的滑模控制，闭环控制系统示意图如图 3.2 所示。

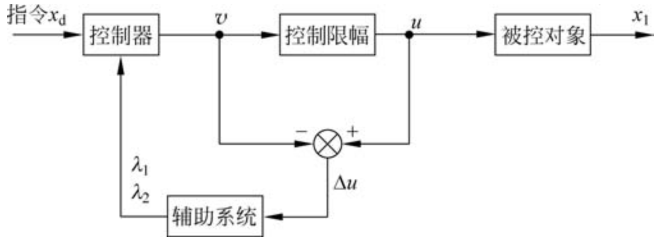


图 3.2　基于控制受限下的闭环控制系统

通过设计一个稳定的自适应辅助系统，可实现控制饱和的补偿[1]。设计辅助系统为

$$\begin{cases}\dot{\lambda}_1=-c_1\lambda_1+\lambda_2\\ \dot{\lambda}_2=-c_2\lambda_2+b\Delta u\end{cases} \tag{3.3}$$

式(3.3)可写成

$$\dot{\boldsymbol{\lambda}}=\boldsymbol{A\lambda}+\boldsymbol{B}\Delta u$$

其中，$\boldsymbol{A}=\begin{bmatrix}-c_1 & 1\\ 0 & -c_2\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0\\ b\end{bmatrix}$。

为了保证 $t\rightarrow\infty$时，$\lambda_i\rightarrow 0$，需要 $\boldsymbol{A}$ 为 Hurwitz，即需要 $c_1>0$，$c_2>0$，并且为了防止 Δu 过大造成系统式(3.3)不稳定，需要保证 c_1 和 c_2 足够大。

取控制目标为 $x_1\rightarrow x_d$，x_d 为角度指令信号。定义角度误差为

$$e=x_1-x_d-\lambda_1 \tag{3.4}$$

则 $\dot{e}=\dot{x}_1-\dot{x}_d-\dot{\lambda}_1=x_2-\dot{x}_d-\lambda_2+c_1\lambda_1$。

滑模函数为

$$s=ce+\dot{e} \tag{3.5}$$

其中，$c>0$。

于是

$$\begin{aligned}\dot{s}&=c\dot{e}+\ddot{e}=c\dot{e}+\ddot{x}_1-\ddot{x}_d-\ddot{\lambda}_1\\ &=c\dot{e}+f+bu+dt-\ddot{x}_d-(-c_1\dot{\lambda}_1+\dot{\lambda}_2)\\ &=c\dot{e}+f+bu+dt-\ddot{x}_d+c_1(-c_1\lambda_1+\lambda_2)-(-c_2\lambda_2+b\Delta u)\\ &=c\dot{e}+f+bv+dt-\ddot{x}_d+c_1(-c_1\lambda_1+\lambda_2)+c_2\lambda_2\end{aligned}$$

设计控制器为

$$v=\frac{1}{b}(-c\dot{e}-f+\ddot{x}_d-c_1(-c_1\lambda_1+\lambda_2)-c_2\lambda_2-\eta\,\mathrm{sgn}(s)) \tag{3.6}$$

其中，$\eta\geqslant D$。

于是

$$\dot{s}=dt-\eta\,\mathrm{sgn}(s)$$

定义 Lyapunov 函数

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

则

$$\dot{V}=s\dot{s}=s(dt-\eta\mathrm{sgn}(s))=dt\cdot s-\eta|s|\leqslant 0$$

为了保证 $x_1\to x_{\mathrm{d}}$，$\dot{x}_1\to\dot{x}_{\mathrm{d}}$，需要 $\lambda_1\to 0$，$\lambda_2\to 0$，因此，本算法的有效性取决于 $\lambda_i\to 0$ 是否成立，即 Δu 的有界性。在初始条件下，V 有界从而 Δu 有界，针对式(3.3)，通过 c_i 的设定可保证 $\lambda_i\to 0$，从而使 e 和 $\dot{e}$ 都趋近于零，从而实现 $t\to\infty$ 时，V 有界，即 Δu 有界。

3.1.3 仿真实例

被控对象取下式

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=-25x_2+133u+10\sin t\end{cases}$$

则 $f(x,t)=-25x_2$，$b=133$，$D=10$。理想角度指令为 $x_{\mathrm{d}}=\sin t$，系统初始状态为 $[5,0]$，取 $c=1.5$，$c_1=c_2=10$，$\eta=D+0.5$。采用辅助系统式(3.3)和控制律式(3.6)。滑模控制中，采用饱和函数代替切换函数，取边界层厚度 Δ 为 0.02。仿真结果如图 3.3 和图 3.4 所示。

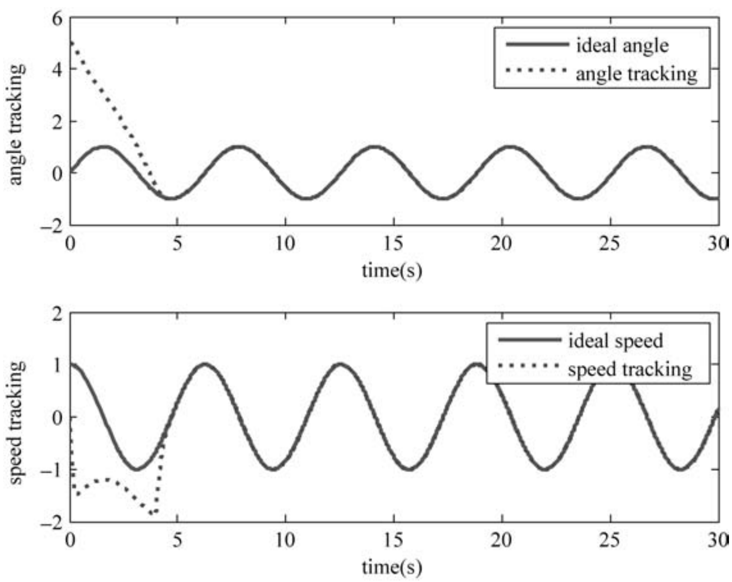


图 3.3　位置和速度跟踪

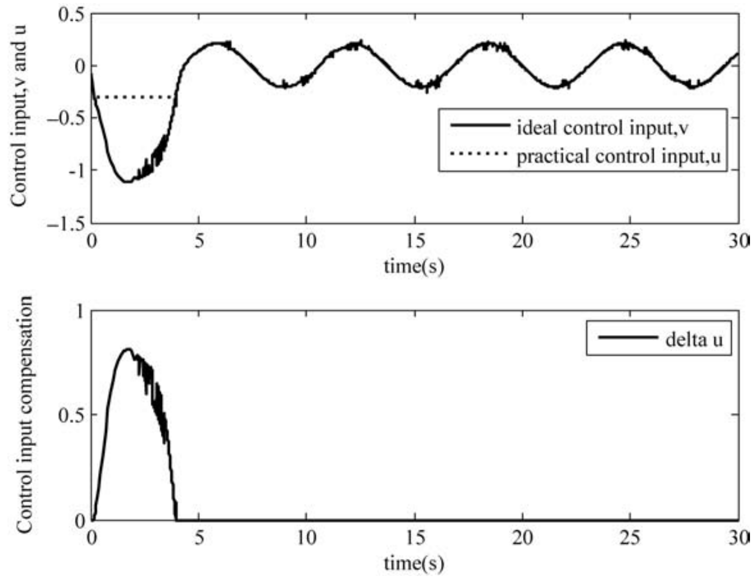
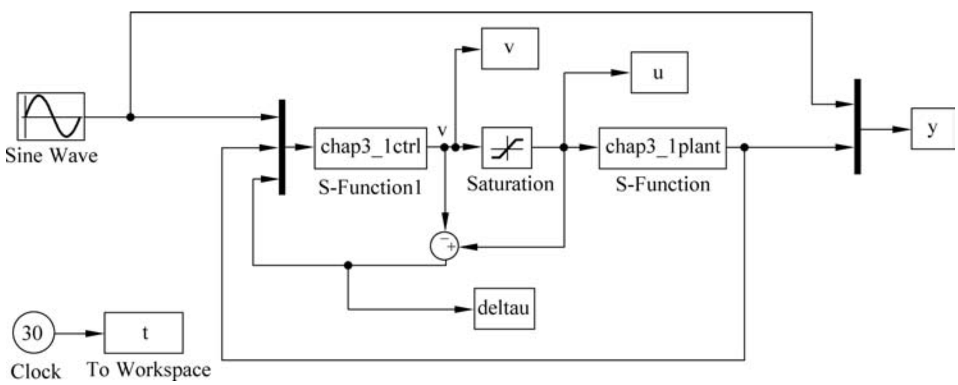


图 3.4　受限前后控制输入 v 与 u 及 Δu

通过仿真结果图 3.4 可见，控制输入 v 有界，因此 Δu 有界，因此辅助系统式(3.3)是稳定的，通过 $\boldsymbol{A}$ 为 Hurwitz 条件的设计，可以保证 $t\to\infty$时，$\lambda_i\to 0$。

仿真程序：

(1) Simulink 主程序：chap3_1sim.mdl。



(2) 控制器子程序：chap3_1ctrl.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 {1,2,4,9}
    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 = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0 0];
str = [];
ts = [-1 0];
function sys = mdlDerivatives(t,x,u)
c1 = 10;c2 = 10;
deltau = u(4);
lamda1 = x(1);
lamda2 = x(2);

b = 133;
sys(1) = -c1 * lamda1 + lamda2;
sys(2) = -c2 * lamda2 + b * deltau;
function sys = mdlOutputs(t,x,u)
c1 = 10;c2 = 10;
xd = sin(t);
dxd = cos(t);
ddxd = -sin(t);

x1 = u(2);
x2 = u(3);
deltau = u(4);
f = -25 * x2;
b = 133;
D = 10;

lamda1 = x(1);
lamda2 = x(2);
dlamda1 = lamda2 - c1 * lamda1;
dlamda2 = -c2 * lamda2 + b * deltau;

e = x1 - xd - lamda1;
de = x2 - dxd - dlamda1;

c = 1.5;
s = c * e + de;
xite = D + 0.5;

fai = 0.02;
if abs(s)<= fai
   sat = s/fai;
```

```
else
   sat = sign(s);
end
vt = -1/b*(c*x2+c2*lamda2+(c1-c)*dlamda1+f-c*dxd-ddxd)-1/b*xite*sat;

sys(1) = vt;
```

(3) 被控对象子程序：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 = [5 0];
str = [];
ts = [-1 0];
function sys = mdlDerivatives(t,x,u)
ut = u(1);
f = -25*x(2);
b = 133;
dt = 10*sin(t);
sys(1) = x(2);
sys(2) = f+b*ut+dt;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 做图子程序：chap3_1plot.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);
subplot(211);
plot(t,v(:,1),'k',t,u(:,1),'k:','linewidth',2);
xlabel('time(s)');ylabel('Control input,v and u');
legend('ideal control input,v','practical control input,u');
subplot(212);
plot(t,deltau(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('Control input compensation');
legend('delta u');
```

3.2 基于 RBF 网络补偿的控制输入受限滑模控制

RBF 神经网络能在一个紧凑集和任意精度下，逼近任何非线性函数。采用 RBF 神经网络可实现控制输入受限的有效补偿。

3.2.1 系统描述

被控对象为

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=f(x,t)+bu+dt\end{cases}\tag{3.7}$$

其中，$b>0$；dt 为干扰，$|dt|\leqslant D$；u 为受限的控制量。

取最大控制输入值为 $u_{\max}$，$\delta=u-v$，$u=\mathrm{sat}(v)$，控制输入受限函数 $\mathrm{sat}(v)$ 表示为

$$\mathrm{sat}(v)=\begin{cases}u_{\max}, & v>u_{\max}\\ v, & |v|\leqslant u_{\max}\\ -u_{\max}, & v<-u_{\max}\end{cases}\tag{3.8}$$

控制输入受限函数示意图如图 3.1 所示。在实际工程中，若执行器幅值未知，会造成 δ 未知。通过设计 RBF 网络，采用 RBF 网络逼近 δ 的方法，可实现一种基于控制输入受限下的滑模控制方法。闭环控制系统示意图如图 3.5 所示。

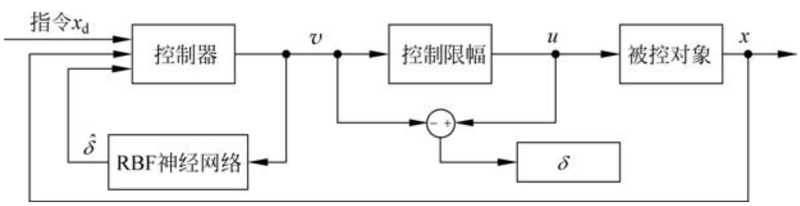


图 3.5 基于控制受限下的闭环控制系统

3.2.2 基于RBF网络控制受限逼近的滑模控制

RBF网络输入输出算法为

$$h_j = \exp\left(\frac{\| \boldsymbol{x} - \boldsymbol{c}_j \|^2}{2b_j^2}\right)$$

$$\delta = \boldsymbol{W}^{*\mathrm{T}} \boldsymbol{h}(\boldsymbol{x}) + \varepsilon \tag{3.9}$$

其中，$\boldsymbol{x}$ 为网络输入；i 表示网络输入层第 i 个的输入；j 为网络隐含层第 j 个网络输入；$\boldsymbol{h} = [h_j]^{\mathrm{T}}$ 为高斯基函数的输出；$\boldsymbol{W}^*$ 为网络的理想权值；ε 为理想神经网络逼近 δ 的误差，$\varepsilon \leqslant \varepsilon_{\max}$。

根据图3.5，网络输入取 $\boldsymbol{x} = v$，则网络输出为

$$\hat{\delta} = \hat{\boldsymbol{W}}^{\mathrm{T}} \boldsymbol{h} \tag{3.10}$$

其中，$\hat{\delta}$ 为网络输出，$\hat{\boldsymbol{W}}$ 为神经网络的估计权值。

取 $\widetilde{\boldsymbol{W}} = \hat{\boldsymbol{W}} - \boldsymbol{W}^*$，则 $\delta - \hat{\delta} = \boldsymbol{W}^{*\mathrm{T}} \boldsymbol{h} + \varepsilon - \hat{\boldsymbol{W}}^{\mathrm{T}} \boldsymbol{h} = (\boldsymbol{W}^{*\mathrm{T}} - \hat{\boldsymbol{W}}^{\mathrm{T}}) \boldsymbol{h} + \varepsilon = -\widetilde{\boldsymbol{W}}^{\mathrm{T}} \boldsymbol{h} + \varepsilon$。

取控制目标为 $x_1 \to x_{\mathrm{d}}$，x_{d} 为角度指令信号。定义角度误差为 $e = x_1 - x_{\mathrm{d}}$，则 $\dot{e} = \dot{x}_1 - \dot{x}_{\mathrm{d}}$，滑模函数为 $s = ce + \dot{e}$，$c > 0$，则

$$\begin{aligned}\dot{s} &= c\dot{e} + \ddot{e} = c\dot{e} + \ddot{x}_1 - \ddot{x}_{\mathrm{d}} \\ &= c\dot{e} + f + bu + dt - \ddot{x}_{\mathrm{d}} \\ &= c\dot{e} + f + b(v + \delta) + dt - \ddot{x}_{\mathrm{d}}\end{aligned}$$

设计控制律为

$$v = \frac{1}{b}(-c\dot{e} - f + \ddot{x}_{\mathrm{d}} - \eta\,\mathrm{sgn}(s)) - \hat{\delta} \tag{3.11}$$

其中，$\eta \geqslant D + b\varepsilon_{\max}$。

于是

$$\dot{s} = -\eta\,\mathrm{sgn}(s) + b(\delta - \hat{\delta}) + dt = -\eta\,\mathrm{sgn}(s) - b\widetilde{\boldsymbol{W}}^{\mathrm{T}} \boldsymbol{h} + b\varepsilon + dt$$

定义 Lyapunov 函数

$$V = \frac{1}{2}s^2 + \frac{1}{2}\gamma \widetilde{\boldsymbol{W}}^{\mathrm{T}} \widetilde{\boldsymbol{W}}$$

其中，$\gamma > 0$。

于是

$$\begin{aligned}\dot{V} &= s\dot{s} + \gamma \widetilde{\boldsymbol{W}}^{\mathrm{T}} \dot{\widetilde{\boldsymbol{W}}} \\ &= -\eta|s| + s \cdot dt + sb(-\widetilde{\boldsymbol{W}}^{\mathrm{T}} \boldsymbol{h} + \varepsilon) + \gamma \widetilde{\boldsymbol{W}}^{\mathrm{T}} \dot{\hat{\boldsymbol{W}}} \\ &= -\eta|s| + s \cdot dt + sb\varepsilon + \widetilde{\boldsymbol{W}}^{\mathrm{T}}(-sb\boldsymbol{h} + \gamma \dot{\hat{\boldsymbol{W}}})\end{aligned}$$

取自适应律为

$$\dot{\hat{\boldsymbol{W}}}=\frac{1}{\gamma}sb\boldsymbol{h} \tag{3.12}$$

则

$$\dot{V}=-\eta|s|+(dt+b\varepsilon)s\leqslant 0$$

由于当且仅当 $s=0$ 时,$\dot{V}=0$,则 $t\to\infty$时,$s\to 0$,且 $\widetilde{W}$ 有界。

神经网络只能逼近有界的函数,这就要求 δ 有界,本算法的稳定性取决于 δ 的有界性,即 v 的有界性。

3.2.3 仿真实例

被控对象取

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=-25x_2+133u+10\sin t\end{cases}$$

则 $f(x,t)=-25x_2$,$b=133$,$D=10$。理想角度指令为 $x_{\mathrm{d}}=\sin t$。为了表明控制系统补偿控制输入受限的能力,采用较大的初始误差,系统初始状态为[10,0]。RBF 网络结构取 1—5—1,网络输入取 $x=v$,根据网络输入实际范围来设计高斯基函数的参数,取 $c_i=6\times[-1.0\quad -0.5\quad 0\quad 0.5\quad 1.0]$ 和 $b_j=5.0$,网络权值的初始值为 0。采用控制律式(3.11)和自适应律式(3.12),取 $c=5$,$\eta=D+0.5$,$\gamma=10$。在滑模控制中,采用饱和函数代替切换函数,取边界层厚度 Δ 为 0.02。仿真结果如图 3.6~图 3.8 所示。

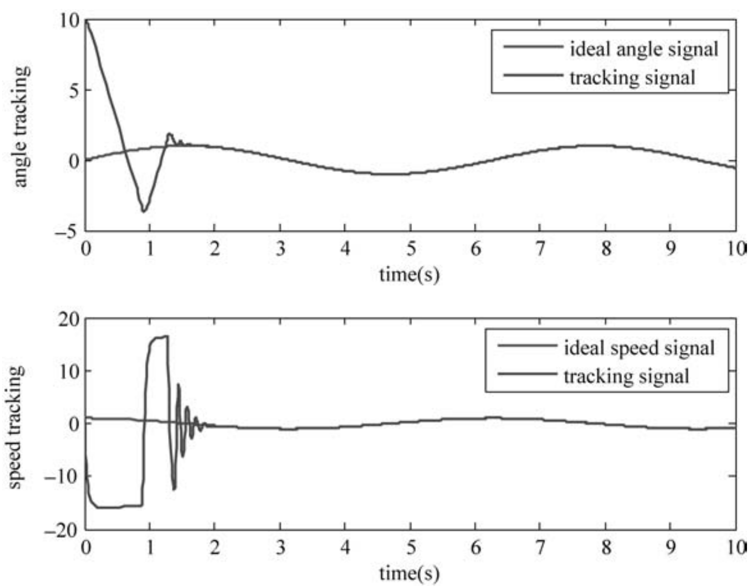


图 3.6 位置和速度跟踪

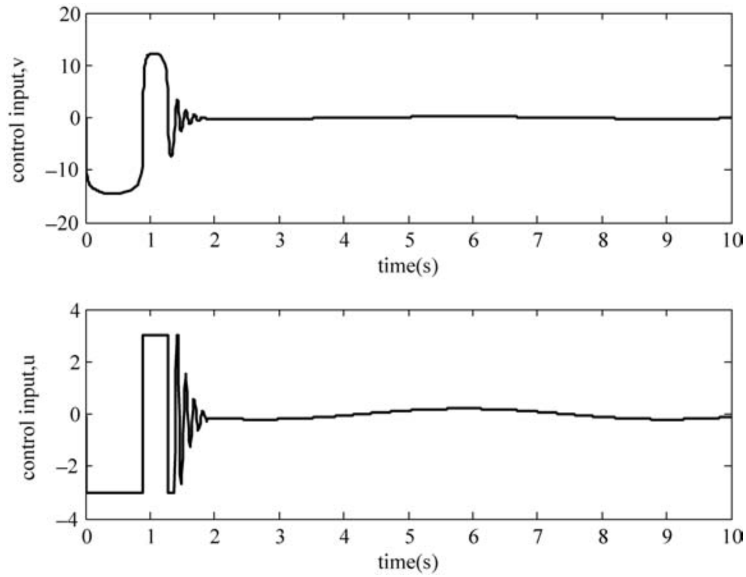


图 3.7　受限前后控制输入 v 与 u

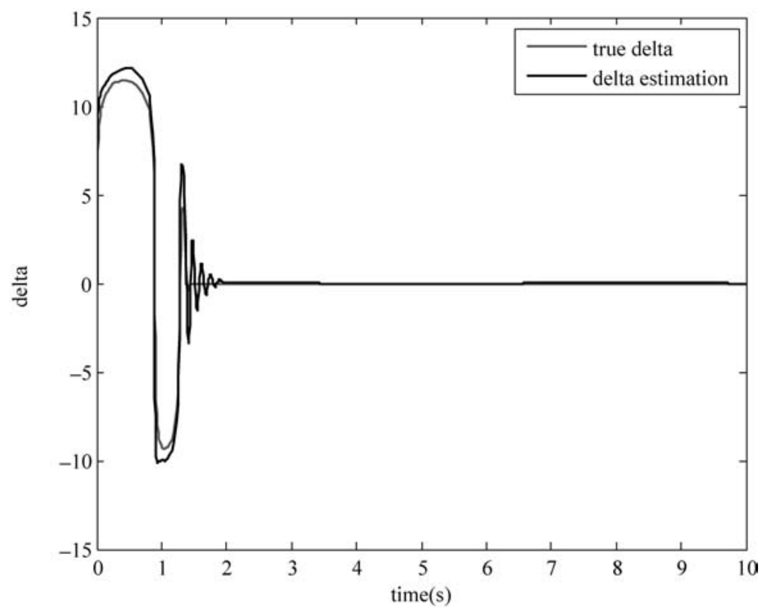
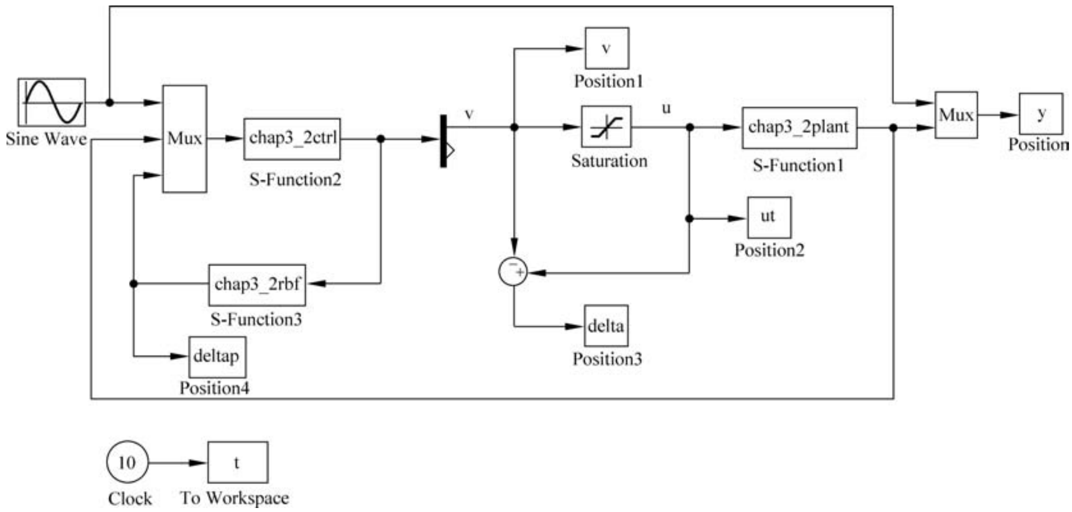


图 3.8　控制输入受限值 δ 及逼近

仿真程序：

（1）Simulink 主程序：chap3_2sim. mdl。



（2）控制器子程序：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 {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 = 4;
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(4);

e = x1 - x1d;
de = x2 - dx1d;
c = 5;
s = c * e + de;

D = 10;
xite = D + 0.50;
fai = 0.02;
if abs(s)<= fai
   sat = s/fai;
else
   sat = sign(s);
end
b = 133;
f = -25 * x2;
v = 1/b * ( - c * de - f + ddx1d - xite * sat) - deltap;
sys(1) = v;
sys(2) = s;
```

(3) 神经网络逼近子程序：chap3_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 = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = 0 * ones(1,5);
str = [];
ts = [];
cij = 6 * [-1 -0.5 0 0.5 1];
bj = 5;
```

```
function sys = mdlDerivatives(t,x,u)
global cij bj
v = u(1);
s = u(2);

b = 133;

xi = v;
h = zeros(5,1);
for j = 1:1:5
    h(j) = exp( - norm(xi - cij(:,j))^2/(2 * bj^2));
end
gama = 10;
for i = 1:1:5
    sys(i) = 1/gama * s * b * h(i);
end
function sys = mdlOutputs(t,x,u)
global cij bj
v = u(1);
xi = v;

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）被控对象子程序：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 = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
```

```
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [10 0];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
sys(1) = x(2);
sys(2) = - 25 * x(2) + 133 * u + 10 * sin(t);
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

（5）做图子程序：chap3_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 signal','tracking signal');
subplot(212);
plot(t,cos(t),'r',t,y(:,3),'b','linewidth',2);
xlabel('time(s)');ylabel('speed tracking');
legend('ideal speed signal','tracking signal');

figure(2);
subplot(211);
plot(t,v(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('control input,v');
subplot(212);
plot(t,ut(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('control input,u');

figure(3);
plot(t,delta(:,1),'r',t,deltap(:,1),'k','linewidth',2);
xlabel('time(s)');ylabel('delta');
legend('true delta','delta estimation');
```

3.3 一种按设定误差性能指标函数收敛的滑模控制

3.3.1 问题描述

考虑如下被控对象：

$$\begin{cases}\dot{x}_1 = x_2 \\ \dot{x}_2 = f(x) + g(x)(u + d(t))\end{cases} \tag{3.13}$$

其中，x_1 为实际输出；u 为控制输入；$f(x)$和 $g(x)$为已知函数；$d(t)$为未知扰动，

$|d(t)| \leqslant D$。

取 x_1 的理想指令为 x_d，跟踪误差为 $e=x_1-x_d$，则 $\dot{e}=x_2-\dot{x}_d$，$\ddot{e}=\dot{x}_2-\ddot{x}_d=f(x)+g(x)(u+d(t))-\ddot{x}_d$。取误差性能指标函数为

$$\lambda(t)=(\lambda(0)-\lambda_\infty)\exp(-lt)+\lambda_\infty \tag{3.14}$$

其中，$l>0$，$0<|e(0)|<\lambda(0)$，$\lambda_\infty>0$，$\lambda_\infty<\lambda(0)$。

则 $\lambda(t)>0$ 且为按指数快速递减到 λ_∞ 的值。

3.3.2 跟踪误差性能函数设计

定理[1-3]：为了保证跟踪误差快速收敛，并达到一定的收敛精度，跟踪误差按下式进行设定

$$e(t)=\lambda(t)S(\varepsilon) \tag{3.15}$$

则

$$S(\varepsilon)=\frac{e(t)}{\lambda(t)} \tag{3.16}$$

函数 $S(\varepsilon)$需要满足如下要求：

(1) $S(\varepsilon)$为光滑连续的单调递增函数；

(2) $-1<S(\varepsilon)<1$；

(3) $\lim\limits_{\varepsilon\to+\infty} S(\varepsilon)=1$，$\lim\limits_{\varepsilon\to-\infty} S(\varepsilon)=-1$。

根据上述要求，设计误差性能函数 $S(\varepsilon)$为双曲正切函数，表示如下：

$$S(\varepsilon)=\frac{\exp(\varepsilon)-\exp(-\varepsilon)}{\exp(\varepsilon)+\exp(-\varepsilon)} \tag{3.17}$$

由于 $-1<S(\varepsilon)<1$，则根据 $\lambda(t)$的定义，可得 $-\lambda(t)<\lambda(t)S(\varepsilon)<\lambda(t)$，即

$$-\lambda(t)<e(t)<\lambda(t) \tag{3.18}$$

从而跟踪误差的收敛集合为

$$\Xi=\{e\in R: |e(t)|\leqslant\lambda_\infty\}$$

通过对跟踪误差的限定，可实现理想的输出，并实现输出值范围的限定。

3.3.3 收敛性分析

根据双曲正切函数性质，函数 $S(\varepsilon)$的反函数为

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

从而

$$\dot{\varepsilon}=\frac{1}{2}\left(\frac{\dot{\lambda}+\dot{e}}{\lambda+e}-\frac{\dot{\lambda}-\dot{e}}{\lambda-e}\right)$$

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

取 $M_1=\dfrac{\ddot{\lambda}(\lambda+e)-(\dot{\lambda}+\dot{e})^2}{2(\lambda+e)^2}$，$M_2=-\dfrac{\ddot{\lambda}(\lambda-e)-(\dot{\lambda}-\dot{e})^2}{2(\lambda-e)^2}$，$M_3=\dfrac{\lambda+e}{2(\lambda+e)^2}+\dfrac{\lambda-e}{2(\lambda-e)^2}$

则

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

取滑模函数为 $\sigma=\dot{\varepsilon}+c\varepsilon$，$c>0$，则

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

则

$$\dot{\sigma}=M_1+M_2+M_3f(x)+u_1+M_3g(x)d(t)-M_3\ddot{x}_{\mathrm{d}}+c\dot{\varepsilon}$$

其中，$u_1=M_3g(x)u$。

设计

$$u_1=-k\sigma-\eta\,\mathrm{sgn}(\sigma)-M_1-M_2-M_3f(x)+M_3\ddot{x}_{\mathrm{d}}-c\dot{\varepsilon}$$

其中 $k>0$。

则控制律为

$$u=\frac{u_1}{M_3g(x)}\tag{3.19}$$

则

$$\dot{\sigma}=-k\sigma-\eta\,\mathrm{sgn}(\sigma)+M_3g(x)d(t)$$

定义 Lyapunov 函数如下：

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

从而有

$$\begin{aligned}
\dot{V}&=\sigma\dot{\sigma}=\sigma(-k\sigma-\eta\,\mathrm{sgn}(\sigma)+M_3g(x)d(t))\\
&=-k\sigma^2-\eta|\sigma|+M_3g(x)d(t)\sigma\leqslant-k\sigma^2=-2kV
\end{aligned}$$

其中，$\eta\geqslant|M_3g(x)|D$。

求解 $\dot{V}\leqslant-2kV$，可以得到如下收敛效果：

$$V(t)\leqslant\exp(-2k(t-t_0))V(t_0)$$

可见，为 $V(t)$ 指数收敛于零，则 σ 指数收敛于零，ε 和 $\dot{\varepsilon}$ 指数收敛于零，收敛速度取决于控制律中的 k 值。

由于双曲正切函数为单调递增函数，函数 $S(\varepsilon)$ 有界且单调收敛于零，由式(3.15)可知，则跟踪误差 $e(t)$ 单调收敛于零，且 $e(t)$ 的收敛范围取决于式(3.18)。

根据双曲正切函数导数的性质，有 $\dot{S}(\varepsilon)=(1-S^2(\varepsilon))\dot{\varepsilon}$，则 $\dot{S}(\varepsilon)$ 指数收敛于零，又

由于 $\dot{\lambda}$ 指数收敛于零，λ 指数收敛于 λ_{∞}，由于 $\dot{\lambda}S$ 和 $\lambda\dot{S}$ 都不是单调的指数收敛，则 $\dot{e}(t)=\dot{\lambda}S+\lambda\dot{S}$ 为单调收敛于零。

3.3.4 仿真实例

针对被控对象，取 $f(x)=-25x_2$，$g(x)=133$，$d(t)=3\sin t$，初始状态为 $[0.50\quad 0]$，理想指令为 $x_{\mathrm{d}}=\sin t$，则 $e(0)=x_1(0)-x_{\mathrm{d}}(0)=0.50$。误差指标函数取式(3.14)，取 $l=5.0$，$\lambda(0)=0.51$，$\lambda_{\infty}=0.001$，采用控制律式(3.19)，取 $c=50$，$D=3.0$，$\eta=|M_3 g(x)|D+0.10$，$k=10$，仿真结果如图 3.9～图 3.12 所示。

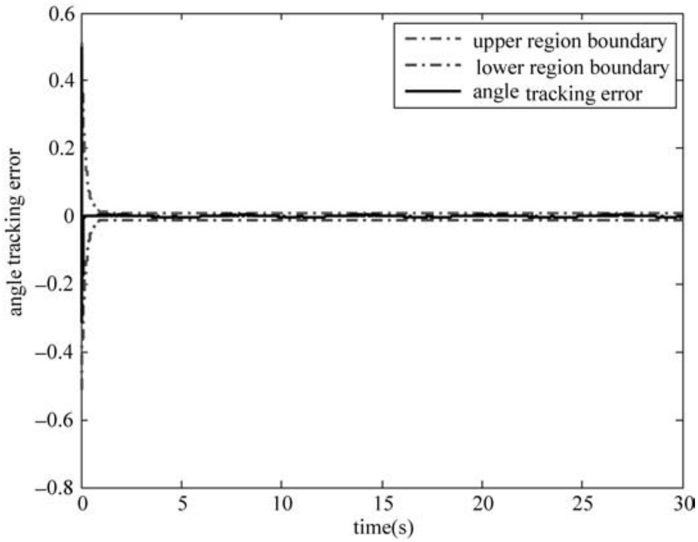


图 3.9　跟踪误差的收敛过程

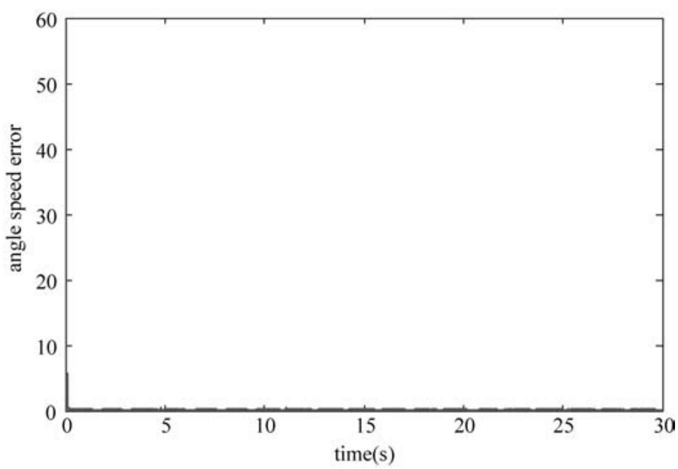


图 3.10　跟踪误差速度的收敛过程

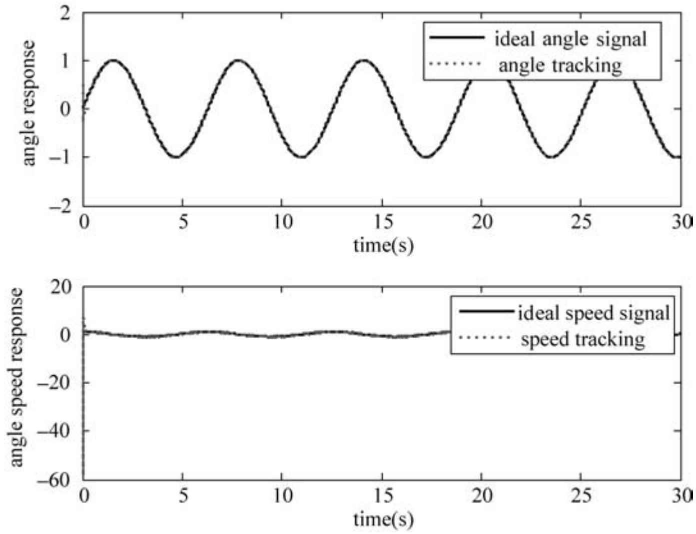


图 3.11　角度和角速度跟踪

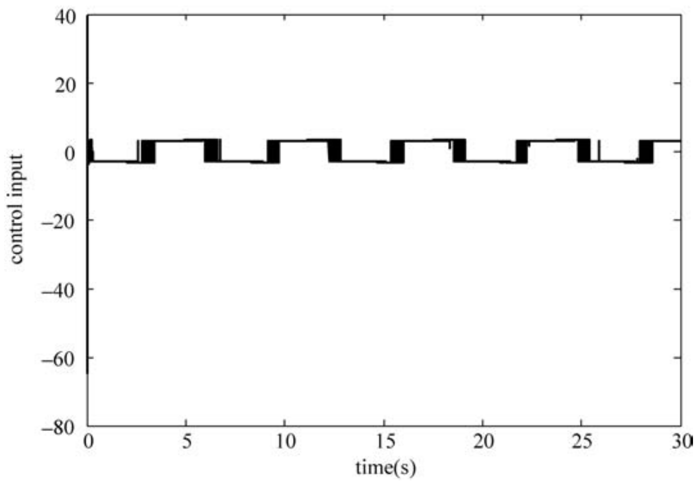


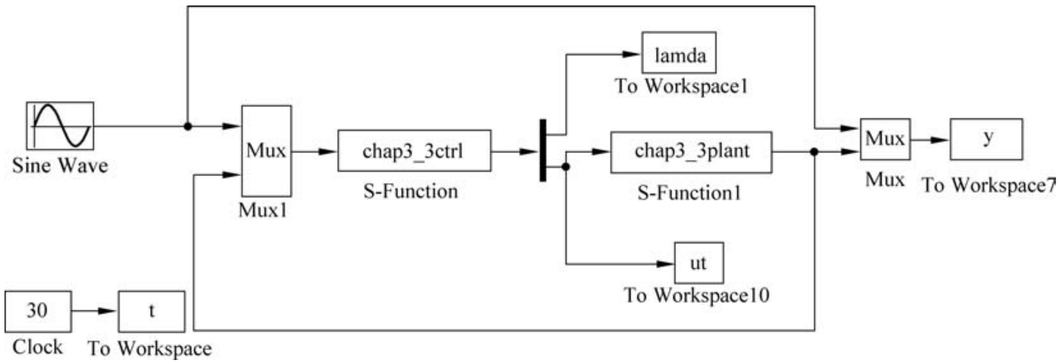
图 3.12　控制输入

需要说明的是，仿真过程中，两种情况下，$\lambda(t)$接近 $e(t)$：(1)$t=0$ 时，当$\lambda(0)$越接近 $e(0)$时；(2)$t\to\infty$时，当 λ_∞ 取值很小时，$e(t)$越接近 λ_∞。在该两种情况下，根据 $S(\varepsilon)=\dfrac{e(t)}{\lambda(t)}$，函数 $S(\varepsilon)$接近于 1.0，此时函数 $S(\varepsilon)$的反函数 $\varepsilon=\dfrac{1}{2}\ln\dfrac{1+S}{1-S}$中越容易产生奇异或$\dfrac{1+S}{1-S}$为负。

避免的方法为：$\lambda(0)$不能过于接近 $e(0)$，λ_∞ 值也不能过小。如果 $\lambda(0)$接近 $e(0)$或 λ_∞ 取值很小时，会造成 M_3 产生奇异，从而 u 产生奇异，为了避免产生奇异，需要相应地改变 Simulink 环境下数值分析求解的方法。本仿真采用定点求解方法，间隔时间取 0.001。

仿真程序：

(1) Simulink 主程序：chap3_3sim.mdl。



(2) 控制器 S 函数：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 {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_inf) * 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）被控对象 S 函数：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 = 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）做图程序：chap3_3plot.m。

```
close all;

figure(1);
plot(t,lamda,'-.r',t,-lamda,'-.b',t,y(:,2)-y(:,1),'k','linewidth',2);
legend('upper region boundary','lower region boundary','angle tracking error');
xlabel('time(s)');ylabel('angle 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 angle signal','angle 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');
```

3.4 基于双曲正切函数的输入受限滑模控制

本节给出一种通过双曲正切函数直接设计有界控制输入的方法。

3.4.1 系统描述

针对如下系统

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= u + d(t)\end{aligned} \tag{3.20}$$

其中，x_1 和 x_2 为状态，控制输入为 u，$|d(t)|\leqslant D$ 为扰动。

控制目标为：在受限的控制输入下，实现 $t\to\infty$ 时，$x_1\to 0$，$x_2\to 0$。

3.4.2 双曲正切光滑函数特点

双曲正切函数定义为

$$\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$$

双曲正切函数具有如下 2 个性质：(1)$|\tanh(x)|\leqslant 1$；(2)$x\tanh(x)\geqslant 0$。采用双曲正切光滑函数可实现控制输入的有界。

3.4.3 控制器的设计及分析

针对模型式(3.20)，设计滑模函数为

$$s=cx_1+x_2$$

其中，$c>0$。

定义 Lyapunov 函数为

$$V=\frac{1}{2}s^2 \tag{3.21}$$

则

$$\dot{V}=s\dot{s}=s(u+d(t))$$

设计控制律为

$$u=-\eta\tanh(\eta s)-\lambda\,\mathrm{sgn}s \tag{3.22}$$

其中 $\eta>0,\lambda\geqslant D+\lambda_0,\lambda_0>0$。

则

$$\begin{aligned}\dot{V}&=s(-\eta\tanh(\eta s)-\lambda\,\mathrm{sgn}s+d)\\&=s(-\eta\tanh(\eta s)-\lambda\,\mathrm{sgn}s+d)\\&=-\eta s\tanh(\eta s)+ds-\lambda\mid s\mid\end{aligned}$$

由于 $\eta s\tanh(\eta s)>0$，则

$$\dot{V}\leqslant-\lambda_0\mid s\mid$$

从而 V 渐进收敛，即当 $t\to\infty$ 时，$s\to0$，$x_1\to0$，$x_2\to0$ 且渐进收敛。

由于 $\tanh(x)=\dfrac{e^x-e^{-x}}{e^x+e^{-x}}\in[-1\quad+1]$，则由式(3.22)可得控制输入幅值为

$$\mid u\mid\leqslant\eta+\lambda$$

因此，针对模型式(3.20)的结构，按式(3.22)设计控制律，通过设计 η 和 λ，便可实现控制输入的受限。

3.4.4 仿真实例

考虑被控对象为式(3.20)，$d(t)=1.5\sin t$，初始状态为 $[0.1\quad0]$。取 $c=30$，按式(3.22)设计控制律，采用饱和函数方法，取边界层厚度 Δ 为 0.10。取 $\eta=3.0$，$\lambda=1.6$，则 $|u|\leqslant4.6$。

仿真结果如图 3.13 和图 3.14 所示。

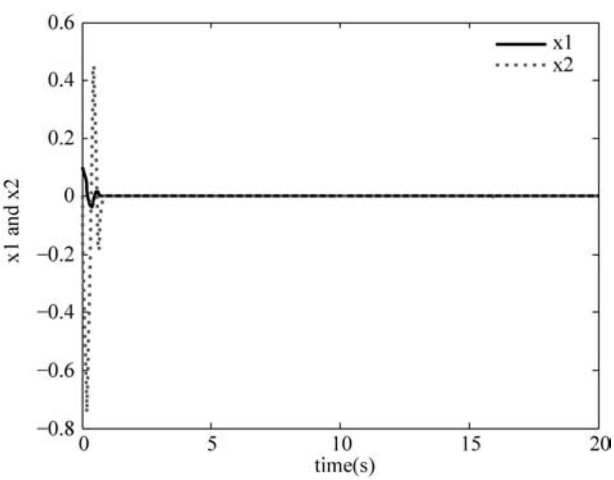


图 3.13　状态响应

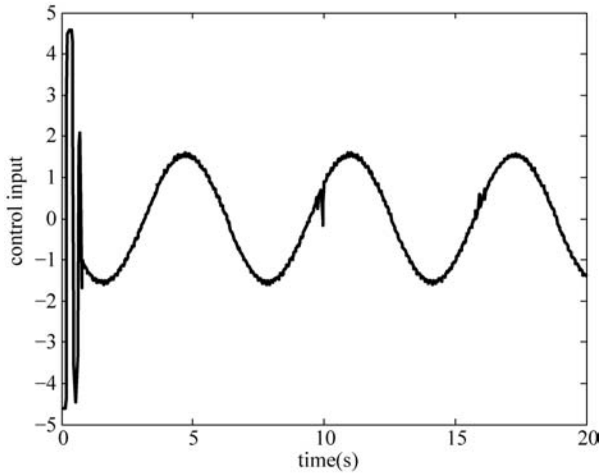
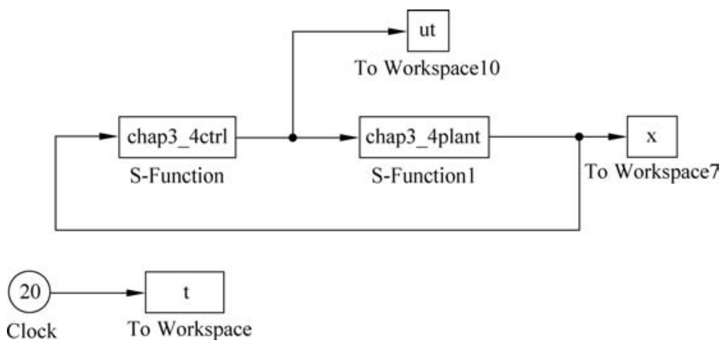


图 3.14　控制输入

仿真程序：

(1) Simulink 主程序：chap3_4sim.mdl。



(2) 控制器程序：chap3_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 {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      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
```

```
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];
function sys = mdlOutputs(t,x,u)
x1 = u(1);x2 = u(2);
c = 30;
s = c * x1 + x2;

xite = 3.0;
lamda = 1.6;

fai = 0.01;
if abs(s)<= fai
   s_sat = s/fai;
else
   s_sat = sign(s);
end

% ut = - xite * tanh(xite * s) - lamda * sign(s);
ut = - xite * tanh(xite * s) - lamda * s_sat;
sys(1) = ut;
```

(3) 被控对象程序：chap3_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 = [0.1 0];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
ut = u(1);
dt = 1.5 * sin(t);
sys(1) = x(2);
sys(2) = ut + dt;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
```

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

(4) 做图程序：chap3_4plot.m。

```
close all;

figure(1);
plot(t,x(:,1),'k',t,x(:,2),'r:','linewidth',2);
xlabel('time(s)');ylabel('x1 and x2');
legend('x1','x2');

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

3.5 基于 RBF 网络的输入受限滑模控制

3.5.1 系统描述

针对如下系统

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= u + f_0(x) + d(t)\end{aligned} \tag{3.23}$$

其中，x_1 和 x_2 为状态，控制输入为 u，$f_0(x)$为未知函数，$|d(t)|\leqslant D$ 为扰动。

控制目标为：在受限的控制输入下，实现 $t\to\infty$时，$x_1\to 0$，$x_2\to 0$。

3.5.2 RBF 神经网络逼近

采用 RBF 网络可实现未知函数 $f(\boldsymbol{x})$的逼近，RBF 网络算法为

$$f = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \varepsilon$$

其中，$\boldsymbol{x}$ 为网络的输入，$\boldsymbol{h}(\boldsymbol{x})$为高斯函数的输出，$\boldsymbol{W}$ 为网络的理想权值，ε 为网络的逼近误差，$|\varepsilon|\leqslant\varepsilon_{\mathrm{N}}$，$\boldsymbol{W}=[W_1\quad\cdots\quad W_n]^{\mathrm{T}}$，$W_{\min}\leqslant|W_j|\leqslant W_{\max}$，$j=1,2,\cdots,n$。

采用 RBF 逼近未知函数 f，网络的输入取 $\boldsymbol{x}=[x_1\quad x_2]^{\mathrm{T}}$，则 RBF 网络的输出为

$$\hat{f}(\boldsymbol{x}) = \hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \tag{3.24}$$

则

$$\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \varepsilon - \hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) = \tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \varepsilon$$

并定义 $\tilde{\boldsymbol{W}}=\boldsymbol{W}-\hat{\boldsymbol{W}}$。

由于未知函数 $f(\boldsymbol{x})$有界，则 $\hat{f}(\boldsymbol{x})$有界，可令$|\hat{f}(\boldsymbol{x})|\leqslant\hat{f}_{\max}$。

3.5.3 控制器的设计及分析

取滑模函数为

$$s = cx_1 + x_2$$

其中，$c>0$。

定义 Lyapunov 函数为

$$V = \frac{1}{2}s^2 + \frac{1}{2\gamma}\tilde{\boldsymbol{W}}^{\mathrm{T}}\tilde{\boldsymbol{W}} \tag{3.25}$$

其中，$\gamma>0$。

则

$$\begin{aligned}\dot{V} &= s\dot{s} - \frac{1}{\gamma}\tilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}} \\ &= s(cx_2 + u + f_0(x) + d(t)) - \frac{1}{\gamma}\tilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}} \\ &= s(u + f(x) + d(t)) - \frac{1}{\gamma}\tilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}}\end{aligned}$$

其中，$f(x)=cx_2+f_0(x)$。

设计控制律为

$$u = -\eta\tanh(\eta s) - \hat{f}(x) - \lambda\,\mathrm{sgn}s \tag{3.26}$$

其中，$\eta>0$，$\lambda\geqslant D+\varepsilon_N+\lambda_0$，$\lambda_0>0$。

则

$$\begin{aligned}\dot{V} &= s(-\eta\tanh(\eta s) - \hat{f}(x) + f(x) + d(t)) - \frac{1}{\gamma}\tilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}} \\ &= s(-\eta\tanh(\eta s) + \tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \varepsilon + d - \lambda\,\mathrm{sgn}s) - \frac{1}{\gamma}\tilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}} \\ &= -\eta s\tanh(\eta s) + \tilde{\boldsymbol{W}}^{\mathrm{T}}\left(\boldsymbol{h}(\boldsymbol{x})s - \frac{1}{\gamma}\dot{\hat{\boldsymbol{W}}}\right) + \varepsilon s + ds - \lambda\,|s|\end{aligned}$$

如果按照传统方法，设计自适应律为

$$\dot{\hat{\boldsymbol{W}}} = \gamma\boldsymbol{h}(\boldsymbol{x})s \tag{3.27}$$

按上式设计自适应律，$\hat{\boldsymbol{W}}$ 的界无法设定，从而无法得到 $\hat{f}(x)$的界。

由于 $f(x)$有界，为了实现该函数的有界逼近，采用神经网络有界映射自适应律[4]。首先设计投影映射算子，考虑 $\tilde{\boldsymbol{W}}=\boldsymbol{W}-\hat{\boldsymbol{W}}$，定义$\boldsymbol{\xi}=\boldsymbol{h}(\boldsymbol{x})s$，为了保证 $\tilde{\boldsymbol{W}}^{\mathrm{T}}\left(\boldsymbol{h}(\boldsymbol{x})s-\frac{1}{\gamma}\dot{\hat{\boldsymbol{W}}}\right)\leqslant 0$，且 $\boldsymbol{W}_{\min}\leqslant|\hat{\boldsymbol{W}}_j|\leqslant\boldsymbol{W}_{\max}$，取

$$\dot{\hat{\boldsymbol{W}}}_j = \gamma\,\mathrm{proj}_{\hat{W}}(\xi_j)$$

其中

$$\mathrm{proj}_{\hat{W}}(\xi_j) = \begin{cases} 0, & \hat{\boldsymbol{W}}_j \geqslant \boldsymbol{W}_{\max}, \xi_j > 0 \\ 0, & \hat{\boldsymbol{W}}_j \leqslant \boldsymbol{W}_{\min}, \xi_j < 0 \\ \xi_j, & \text{其他} \end{cases} \tag{3.28}$$

取 $\mathrm{proj}_{\hat{W}}(\xi)=\{\mathrm{proj}_{\hat{W}}(\xi_j)\}$，则有界映射自适应律为

$$\dot{\hat{\boldsymbol{W}}}=\gamma\mathrm{proj}_{\hat{W}}(\xi) \tag{3.29}$$

采用自适应律式(3.29)，可保证

$$\tilde{\boldsymbol{W}}^{\mathrm{T}}\left(\boldsymbol{h}(\boldsymbol{x})s-\frac{1}{\gamma}\dot{\hat{\boldsymbol{W}}}\right)\leqslant 0$$

由于 $\eta s\tanh(\eta s)>0$，$\varepsilon s+ds-\lambda|s|\leqslant\lambda_0|s|$则

$$\dot{V}\leqslant-\lambda_0\ |\ s\ |$$

从而实现 V 的渐进收敛，当 $t\to\infty$时，$s\to 0$，即 $x_1\to 0$，$x_2\to 0$ 且渐进收敛。

由于 $\tanh(x)=\dfrac{\mathrm{e}^x-\mathrm{e}^{-x}}{\mathrm{e}^x+\mathrm{e}^{-x}}\in[-1\quad +1]$，则由式(3.26)可得控制输入幅值为

$$|\ u\ |\leqslant\eta+\hat{f}_{\max}+\lambda$$

因此，针对模型式(3.23)的结构，按式(3.26)设计控制律，并按式(3.29)设计自适应律，通过设计 η、λ、RBF 网络隐含层节点和 $W_{\max}$，便可实现控制输入的受限。

3.5.4 仿真实例

考虑被控对象为式(3.23)，初始状态为[0.5 0]。RBF 网络采用 5 个隐含层节点，即 $n=5$，则 $\hat{f}_{\max}(\boldsymbol{x})=\hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})\leqslant\|\hat{\boldsymbol{W}}\|\ \|\boldsymbol{h}(\boldsymbol{x})\|\leqslant 5W_{\max}$。根据网络输入 x_2 的实际范围来设计高斯基函数的参数，参数 $\boldsymbol{c}_i$ 和 b_i 取值分别为[−1 −0.5 0 0.5 1]和 3.0。网络权值中各个元素的初始值取 0.10。

按式(3.26)设计控制律中，采用饱和函数方法，取边界层厚度 Δ 为 0.10。按式(3.29)设计自适应律，取 $\hat{\boldsymbol{W}}$ 的初始值为 0.1，取 $\boldsymbol{W}_{\min}=-1.0$，$\boldsymbol{W}_{\max}=1.0$，$\gamma=0.10$，$\lambda=1.0$，$\eta=10$，则

$$|\ u\ |\leqslant\eta+\hat{f}_{\max}+\lambda\leqslant 10+5+1=16$$

仿真结果如图 3.15～图 3.17 所示。

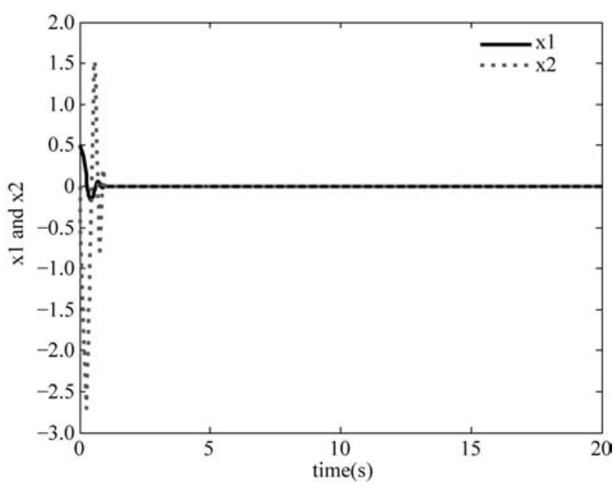


图 3.15 状态响应

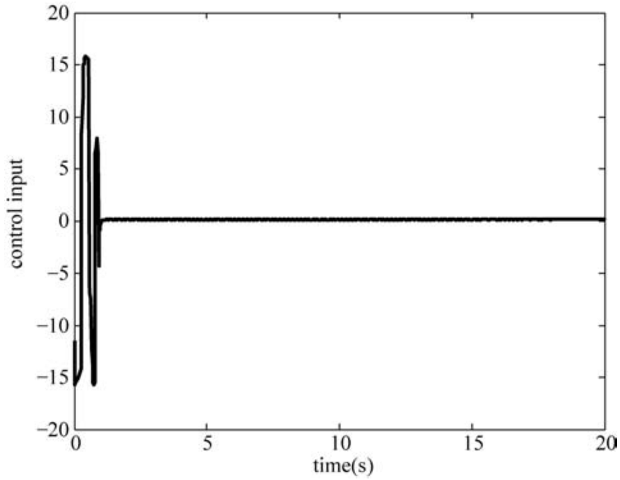


图 3.16 控制输入

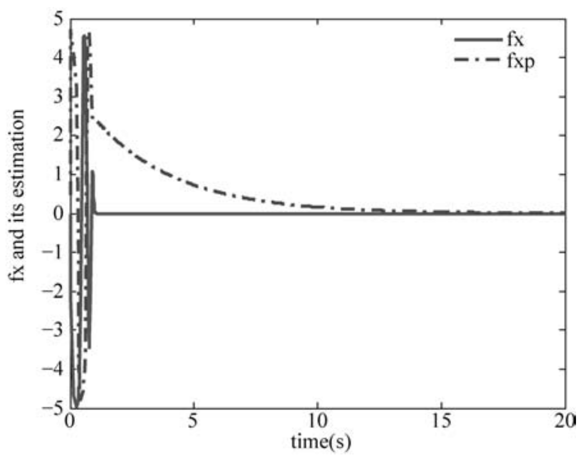
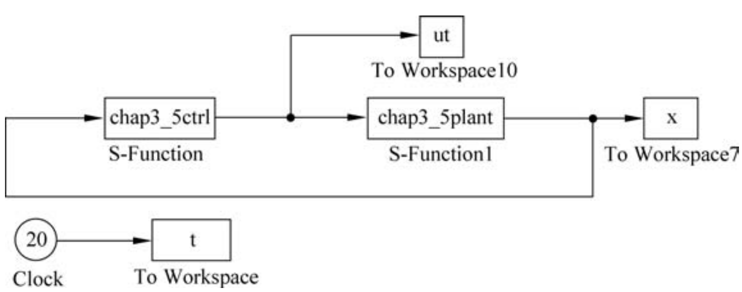


图 3.17 $f(x)$及其逼近效果

仿真程序：

(1) Simulink 主程序：chap3_5sim.mdl。



(2) 控制器程序：chap3_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
global bj cij
cij = [ -1  -0.5 0 0.5 1;
      -1  -0.5 0 0.5 1];
bj = 3.0;

sizes = simsizes;
sizes.NumContStates   = 5;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [0.1 0.1 0.1 0.1 0.1];
str = [];
ts  = [];
function sys = mdlDerivatives(t,x,u)
global bj cij

x1 = u(1);x2 = u(2);
c = 30;
s = c * x1 + x2;

xi = [x1;x2];
h = zeros(5,1);
for j = 1:1:5
    h(j) = exp( - norm(xi - cij(:,j))^2/(2 * bj^2));
end

gama = 10;
W = [x(1) x(2) x(3) x(4) x(5)];
Wmin = - 1;Wmax = 1;

for j = 1:1:5
% dw(j) = gama * h(j) * s;
% sys(j) = dw(j);

Kesi(j) = h(j) * s;
if W(j)> = Wmax&Kesi(j)> 0
Proj_w(j) = 0;
elseif W(j)< = Wmin&Kesi(j)< 0
Proj_w(j) = 0;
else
```

```
Proj_w(j) = Kesi(j);
end
dw(j) = gama * Proj_w(j);
sys(j) = dw(j);
end

function sys = mdlOutputs(t,x,u)
global bj cij
x1 = u(1);x2 = u(2);
c = 30;
s = c * x1 + x2;

xi = [x1;x2];
h = zeros(5,1);
for j = 1:1:5
    h(j) = exp( - norm(xi - cij(:,j))^2/(2 * bj^2));
end

W = [x(1) x(2) x(3) x(4) x(5)];
fp = W * h;

xite = 10;
lamda = 1.0;

fai = 0.02;
if abs(s)< = fai
   s_sat = s/fai;
else
   s_sat = sign(s);
end

% ut = - xite * tanh(xite * s) - fp - lamda * sign(s);
ut = - xite * tanh(xite * s) - fp - lamda * s_sat;

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

(3) 被控对象程序：chap3_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     = 3;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.5 0];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
ut = u(1);
fx = 5 * tanh(x(2));
sys(1) = x(2);
sys(2) = ut - fx;
function sys = mdlOutputs(t,x,u)
fx = 5 * tanh(x(2));

sys(1) = x(1);
sys(2) = x(2);
sys(3) = fx;
```

(4) 做图程序：chap3_5plot. m。

```
close all;

figure(1);
plot(t,x(:,1),'k',t,x(:,2),'r:','linewidth',2);
xlabel('time(s)');ylabel('x1 and x2');
legend('x1','x2');

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

figure(3);
plot(t,x(:,3),'r',t,ut(:,2),'-.b','linewidth',2);
xlabel('time(s)');ylabel('fx and its estimation');
legend('fx','fxp');
```

参考文献

[1] Bechlioulis C P，Rovithakis G A. Robust Adaptive Control of Feedback Linearizable MIMO Nonlinear Systems with Prescribed Performance[J]. IEEE Transactions on Automatic Control，2008，53(9)：2090-2099.

[2] Bechlioulis C P，Rovithakis G A. Adaptive Control with Guaranteed Transient and Steady State Tracking Error Bounds for Strict Feedback Systems，Automatica[J]. 2009，45(2)：532-538.

[3] Bechlioulis C P，Rovithakis G A. Prescribed Performance Adaptive Control for Multi-Input Multi-Output Affine in the Control Nonlinear Systems[J]. IEEE Transactions on Automatic Control，2010，55(5)：1220-1226.

[4] Gong J Q，Yao B. Neural Network Adaptive Robust Control of Nonlinear Systems[C]// Semi-strict Feedback Form，Automatica，2001，37(8)：1149-1160.