# 第 5 章 RBF 神经网络滑模控制

CHAPTER 5

早在 20 世纪 50 年代，苏联学者 Emelyanov 等就提出了滑模控制(Sliding Mode Control，SMC)方法。在以后几十年中，滑模控制设计引起了国内外学者的广泛关注。

滑模控制为含有不确定性的非线性系统鲁棒控制提供了有效的控制设计方法。滑模变结构控制的原理是根据系统所期望的动态特性来设计系统的切换超平面，通过滑动模态控制器使系统状态从超平面之外向切换超平面运动。系统一旦到达切换超平面，控制作用将保证系统状态沿切换超平面到达系统原点，这一沿切换超平面向原点的滑动过程称为滑模运动。由于系统的特性和参数只取决于设计的切换超平面而与外界干扰没有关系，所以滑模变结构控制具有很强的鲁棒性。

近年来，一些学者[1-2]将滑模控制结合神经网络用于非线性系统的控制中。滑模控制存在稳定性分析难，达到条件难以满足及抖振等问题[3]。如果系统的数学模型已知，滑模控制器可以使系统输出直接跟踪期望指令。但较大的外界扰动需要较大的切换增益，这就造成抖振，抖振是滑模控制中难以避免的问题。采用神经网络对滑模控制进行补偿，为解决这一问题提供了有效的途径。

## 5.1 经典滑模控制器设计

针对线性系统

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}u, \quad \boldsymbol{x} \in \mathbf{R}^n, \quad u \in \mathbf{R} \tag{5.1}$$

滑模面设计为

$$s(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = \sum_{i=1}^{n} c_i x_i = \sum_{i=1}^{n-1} c_i x_i + x_n \tag{5.2}$$

其中，$\boldsymbol{x}$ 为状态向量，$\boldsymbol{c} = [c_1 \quad \cdots \quad c_{n-1} \quad 1]^{\mathrm{T}}$。

在滑模控制中，参数 $c_1, c_2, \cdots, c_{n-1}$ 应满足多项式 $p^{n-1} + c_{n-1}p^{n-2} + \cdots + c_2 p + c_1$ 为 Hurwitz，其中 $p$ 为拉氏算子。

例如，$n=2$ 时，假设 $\dot{x}_1 = x_2$，则可设计 $s(\boldsymbol{x}) = c_1 x_1 + x_2$，为了保证多项式 $p + c_1$ 为 Hurwitz，需要多项式 $p + c_1 = 0$ 的特征根实数部分为负，即 $c_1 > 0$。例如，取 $c_1 = 10$，可得滑模面为 $s(\boldsymbol{x}) = 10x_1 + x_2$。

又例如，当 $n=3$ 时，设 $\dot{x}_1 = x_2, \dot{x}_2 = x_3$，则可设计 $s(\boldsymbol{x}) = c_1 x_1 + c_2 x_2 + x_3$，针对 $n=2$，

为了保证多项式 $p^2+c_2p+c_1$ 为 Hurwitz，需要多项式 $p^2+c_2p+c_1=0$ 特征根实数部分为负。可取 $p^2+2\lambda p+\lambda^2=0$，即 $(p+\lambda)^2=0$，取 $\lambda>0$，即可满足多项式 $p^2+2\lambda p+\lambda^2=0$ 的特征根实数部分为负，从而可得 $c_2=2\lambda$，$c_1=\lambda^2$，例如，令 $\lambda=5$，可得 $c_1=25$，$c_2=10$，则 $s(\boldsymbol{x})=25x_1+10x_2+x_3$。

对于一个二阶系统，滑模控制设计可分为两步：首先设计滑模面，以便使系统能够按照预定的“滑动模态”轨迹运动，然后设计控制器，使系统的状态沿滑模面运动。上述设计的基础是构建 Lyapunov 函数。

考虑如下被控对象：

$$J\ddot{\theta}(t)=u(t)+dt \tag{5.3}$$

其中，$J$ 为转动惯量；$\theta(t)$ 为角度；$u(t)$ 为控制输入；$dt$ 为外界扰动且满足 $|dt|\leqslant D$。

设计滑模函数为

$$s(t)=ce(t)+\dot{e}(t) \tag{5.4}$$

其中，$c$ 必须满足 Hurwitz 条件，即 $c>0$。

定义跟踪误差，并求导可得

$$e(t)=\theta(t)-\theta_{\mathrm{d}}(t),\quad \dot{e}(t)=\dot{\theta}(t)-\dot{\theta}_{\mathrm{d}}(t)$$

其中，$\theta_{\mathrm{d}}(t)$ 为理想跟踪指令。

设计 Lyapunov 函数为

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

则

$$\dot{s}(t)=c\dot{e}(t)+\ddot{e}(t)=c\dot{e}(t)+\ddot{\theta}(t)-\ddot{\theta}_{\mathrm{d}}(t)=c\dot{e}(t)+\frac{1}{J}(u+dt)-\ddot{\theta}_{\mathrm{d}}(t) \tag{5.5}$$

且

$$s\dot{s}=s\left[c\dot{e}+\frac{1}{J}(u+dt)-\ddot{\theta}_{\mathrm{d}}\right]$$

为了保证 $s\dot{s}<0$，设计滑模控制律为

$$u(t)=J[-c\dot{e}+\ddot{\theta}_{\mathrm{d}}-\eta\mathrm{sgn}(s)]-D\mathrm{sgn}(s) \tag{5.6}$$

则

$$s\dot{s}=s\left[c\dot{e}+\frac{1}{J}(u+dt)-\ddot{\theta}_{\mathrm{d}}\right]$$

$$s\dot{s}=-\eta|s|-\frac{D}{J}|s|<0$$

从而

$$\dot{V}\leqslant 0\quad(\dot{V}=0,\text{当 } s=0)$$

从控制律的表达式可知，滑模控制器具有很好的鲁棒性。然而，当干扰 $dt$ 较大时，为了保证鲁棒性，必须保证足够大的干扰上界 $D$，而较大的干扰上界 $D$ 会导致切换增益过大，从而造成抖振。

另外，在控制律式(5.6)中，建模信息 $J$ 必须是精确已知的，这在实际工程中是难以实现的，采用 RBF 神经网络逼近方法可有效地解决这一难题。

## 5.2　基于 RBF 神经网络的二阶 SISO 系统的滑模控制

### 5.2.1　系统描述

考虑如下二阶被控对象：

$$\ddot{\theta}=f(\theta,\dot{\theta})+g(\theta,\dot{\theta})u+d(t) \tag{5.7}$$

其中，$f(\cdot)$和 $g(\cdot)$为非线性函数；$u\in\mathbf{R}$ 和 $y\in\mathbf{R}$ 分别为控制输入和系统输出；$d(t)$为外界干扰，且满足 $|d(t)|\leqslant D$。

理想跟踪指令为 $\theta_{\mathrm{d}}$，定义跟踪误差为

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

设计滑模面为

$$s=\dot{e}+ce \tag{5.8}$$

其中，$c>0$，则

$$\dot{s}=\ddot{e}+c\dot{e}=\ddot{\theta}_{\mathrm{d}}-\ddot{\theta}+c\dot{e}=\ddot{\theta}_{\mathrm{d}}-f-gu-d(t)+c\dot{e} \tag{5.9}$$

如果 $f$ 和 $g$ 是已知的，可设计控制律为

$$u=\frac{1}{g}[-f+\ddot{\theta}_{\mathrm{d}}+c\dot{e}+\eta\mathrm{sgn}(s)] \tag{5.10}$$

将式(5.10)代入式(5.9)，可得

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

如果选择 $\eta\geqslant D$，可得

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

如果 $f(\cdot)$未知，可通过逼近 $f(\cdot)$来实现稳定控制设计。下面介绍 RBF 神经网络对未知项 $f(\cdot)$的逼近算法。

### 5.2.2　基于 RBF 神经网络逼近 $f(\cdot)$的滑模控制

采用 RBF 神经网络逼近 $f(\cdot)$，RBF 神经网络算法为

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

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

其中，$\boldsymbol{x}$ 为网络的输入；$i$ 为网络的输入个数；$j$ 表示网络隐含层第 $j$ 个节点；$\boldsymbol{h}=[h_j]^{\mathrm{T}}$ 为高斯函数的输出；$\boldsymbol{W}^*$ 为网络的理想权值；$\varepsilon$ 为网络的逼近误差；且 $|\varepsilon|\leqslant\varepsilon_{\mathrm{N}}$。

网络的输入取 $\boldsymbol{x}=[e\quad\dot{e}]^{\mathrm{T}}$，则 RBF 神经网络的输出为

$$\hat{f}(x)=\hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \tag{5.11}$$

其中，$\boldsymbol{h}(\boldsymbol{x})$为 RBF 神经网络的高斯函数。

则控制输入式(5.10)可写为

$$u=\frac{1}{g}[-\hat{f}+\ddot{\theta}_{\mathrm{d}}+c\dot{e}+\eta\mathrm{sgn}(s)] \tag{5.12}$$

将控制律式(5.12)代入式(5.9)，可得

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

$$=-f+\hat{f}-\eta\mathrm{sgn}(s)-d(t)=-\tilde{f}-d(t)-\eta\mathrm{sgn}(s) \tag{5.13}$$

其中，

$$\tilde{f}=f-\hat{f}=\boldsymbol{W}^{*\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})+\varepsilon-\hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})=\tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})+\varepsilon \tag{5.14}$$

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

定义 Lyapunov 函数为

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

其中，$\gamma>0$。

对 Lyapunov 函数 $L$ 求导，综合式(5.12)和式(5.13)，可得

$$\dot{L}=s\dot{s}+\gamma\tilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\tilde{\boldsymbol{W}}}=s[-\tilde{f}-d(t)-\eta\mathrm{sgn}(s)]-\gamma\tilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}}$$

$$=s[-\tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})-\varepsilon-d(t)-\eta\mathrm{sgn}(s)]-\gamma\tilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\hat{\boldsymbol{W}}}$$

$$=-\tilde{\boldsymbol{W}}^{\mathrm{T}}[s\boldsymbol{h}(\boldsymbol{x})+\gamma\dot{\hat{\boldsymbol{W}}}]-s[\varepsilon+d(t)+\eta\mathrm{sgn}(s)]$$

设计自适应律为

$$\dot{\hat{\boldsymbol{W}}}=-\frac{1}{\gamma}s\boldsymbol{h}(\boldsymbol{x}) \tag{5.15}$$

则

$$\dot{L}=-s[\varepsilon+d(t)+\eta\mathrm{sgn}(s)]=-s[\varepsilon+d(t)]-\eta|s|$$

由于逼近误差 $\varepsilon$ 可以限制得足够小，取 $\eta\geqslant\varepsilon_{\mathrm{N}}+D$，可得 $\dot{L}\leqslant 0$。

存在 $\eta_0>0$，$\eta\geqslant\eta_0+\varepsilon_{\mathrm{N}}+D$，使得

$$\dot{L}\leqslant-\eta_0|s|\leqslant 0$$

由于 $L\geqslant 0$，$\dot{L}\leqslant 0$，从而 $s$ 和 $\tilde{\boldsymbol{W}}$ 有界。当 $\dot{L}\equiv 0$ 时，$s=0$，根据 LaSalle 不变性原理[8,9]，闭环系统为渐近稳定，当 $t\to\infty$ 时，$s\to 0$，从而 $e\to 0$，$\dot{e}\to 0$。

### 5.2.3 仿真实例

考虑单级倒立摆动力学方程

$$\dot{x}_1=x_2$$

$$\dot{x}_2=\frac{g\sin x_1-mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l[4/3-m\cos^2 x_1/(m_c+m)]}+\frac{\cos x_1/(m_c+m)}{l[4/3-m\cos^2 x_1/(m_c+m)]}u$$

其中，$x_1$ 和 $x_2$ 分别为摆角和摆速；$g=9.8\mathrm{m/s}^2$ 为重力加速度；$m_c=1\mathrm{kg}$ 为小车质量；$m=0.1\mathrm{kg}$ 为摆的质量；$l=0.5\mathrm{m}$ 为摆长的一半；$u$ 为控制输入。

取 $x_1=\theta$，期望轨迹为 $\theta_{\mathrm{d}}(t)=0.1\sin(t)$，系统的初始状态为 $[\pi/60,0]$。采用控制律式(5.12)和自适应律式(5.15)，控制参数取 $c=15$，$\eta=0.1$ 和自适应参数取 $\gamma=0.05$。

神经网络的结构取为 2-5-1，$\boldsymbol{c}_i$ 和 $b_i$ 分别设置为 $[-1.0\quad -0.5\quad 0\quad 0.5\quad 1.0]$ 和 $b_j=0.50$，网络的初始权值为 0.10。仿真结果如图 5.1～图 5.3 所示。

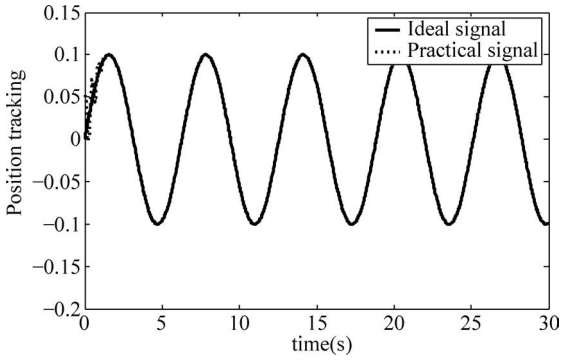



图 5.1　摆角跟踪

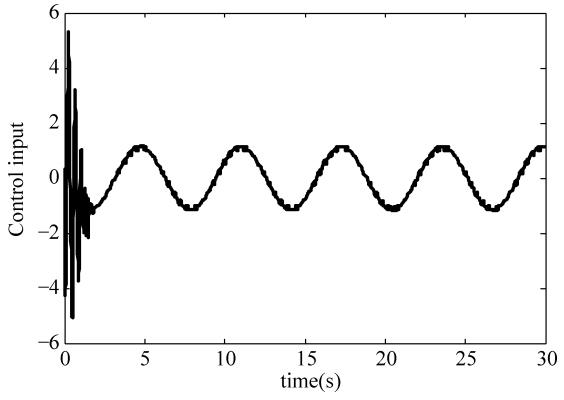



图 5.2　控制输入

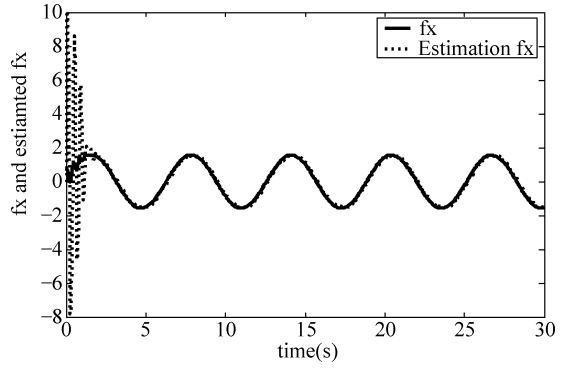



图 5.3　$f(x)$和$\hat{f}(x)$

仿真主程序为 chap5_1sim. mdl，详见附录。

## 5.3　基于 RBF 逼近未知函数 $f(\cdot)$和 $g(\cdot)$的滑模控制

### 5.3.1　引言

考虑二阶非线性系统式(5.7)，假设 $f(\cdot)$和 $g(\cdot)$都是未知的非线性函数，$u\in\mathbf{R}$ 和 $y\in\mathbf{R}$ 分别是控制输入和系统的输出，$d(t)$为外界干扰，且满足 $|d(t)|\leqslant D$。

类似5.2节，设理想跟踪指令为$\theta_d$，定义跟踪误差为$e=\theta_d-\theta$，设计滑模面为$s=\dot{e}+ce$，其中$c>0$。

在控制系统设计中，采用两个RBF神经网络分别逼近$f(\cdot)$和$g(\cdot)$，图5.4为基于神经网络的闭环自适应控制系统。

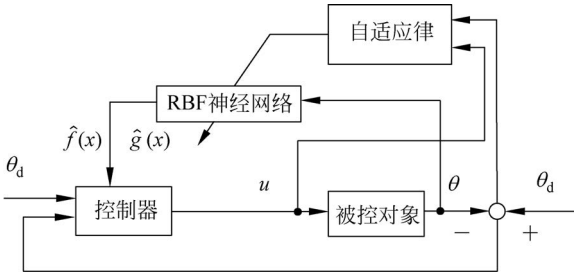



图5.4　基于神经网络的自适应控制系统

采用RBF神经网络逼近$f(x)$，网络算法为

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

$$f(\cdot)=\boldsymbol{W}^{*\mathrm{T}}\boldsymbol{h}_f(x)+\varepsilon_f,\quad g(\cdot)=\boldsymbol{V}^{*\mathrm{T}}\boldsymbol{h}_g(x)+\varepsilon_g$$

其中，$\boldsymbol{x}$为网络的输入；$i$为网络的输入个数；$j$表示网络隐含层第$j$个节点；$\boldsymbol{h}=[h_j]^{\mathrm{T}}$为高斯函数的输出；$\boldsymbol{W}^*$和$\boldsymbol{V}^*$为网络的理想权值；$\varepsilon_f$和$\varepsilon_g$为网络的逼近误差，且$|\varepsilon_f|\leqslant\varepsilon_{\mathrm{Mf}}$，$|\varepsilon_g|\leqslant\varepsilon_{\mathrm{Mg}}$，$f(\cdot)$和$g(\cdot)$分别为理想RBF神经网络的输出。

定义网络的输入为$\boldsymbol{x}=[x_1\quad x_2]^{\mathrm{T}}$，则RBF神经网络的输出为

$$\hat{f}(x)=\hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}_f(x),\quad \hat{g}(x)=\hat{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{h}_g(x) \tag{5.16}$$

其中，$\boldsymbol{h}_f(x)$和$\boldsymbol{h}_g(x)$为RBF神经网络的高斯函数。

则控制律式(5.10)可写为

$$u=\frac{1}{\hat{g}(x)}[-\hat{f}(x)+\ddot{\theta}_d+c\dot{e}+\eta\mathrm{sgn}(s)] \tag{5.17}$$

其中，$\eta$待设定。

将式(5.17)代入式(5.9)，可得

$$\begin{aligned}\dot{s}&=\ddot{e}+c\dot{e}=\ddot{\theta}_d-\ddot{\theta}+c\dot{e}=\ddot{\theta}_d-f-gu-d(t)+c\dot{e}\\&=\ddot{\theta}_d-f-\hat{g}u+(\hat{g}-g)u-d(t)+c\dot{e}\\&=\ddot{\theta}_d-f-\hat{g}\frac{1}{\hat{g}(x)}[-\hat{f}(x)+\ddot{\theta}_d+c\dot{e}+\eta\mathrm{sgn}(s)]+(\hat{g}-g)u-d(t)+c\dot{e}\\&=(\hat{f}-f)-\eta\mathrm{sgn}(s)+(\hat{g}-g)u-d(t)=\tilde{f}-\eta\mathrm{sgn}(s)+\tilde{g}u-d(t)\\&=\tilde{\boldsymbol{W}}^{\mathrm{T}}\varphi_f(\boldsymbol{x})-\varepsilon_f-\eta\mathrm{sgn}(s)+(\tilde{\boldsymbol{V}}^{\mathrm{T}}\varphi_g(\boldsymbol{x})-\varepsilon_g)u-d(t)\end{aligned} \tag{5.18}$$

其中，$\tilde{\boldsymbol{W}}=\boldsymbol{W}^*-\hat{\boldsymbol{W}}$，$\tilde{\boldsymbol{V}}=\boldsymbol{V}^*-\hat{\boldsymbol{V}}$，且

$$\begin{aligned}\tilde{f}&=\hat{f}-f=\hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}_f(\boldsymbol{x})-\boldsymbol{W}^{*\mathrm{T}}\boldsymbol{h}_f(\boldsymbol{x})-\varepsilon_f=\tilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}_f(\boldsymbol{x})-\varepsilon_f\\\tilde{g}&=\hat{g}-g=\hat{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{h}_g(\boldsymbol{x})-\boldsymbol{V}^{*\mathrm{T}}\boldsymbol{h}_g(\boldsymbol{x})-\varepsilon_g=\tilde{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{h}_g(\boldsymbol{x})-\varepsilon_g\end{aligned} \tag{5.19}$$

定义闭环系统 Lyapunov 函数为

$$L=\frac{1}{2}s^2+\frac{1}{2\gamma_1}\widetilde{\boldsymbol{W}}^{\mathrm{T}}\widetilde{\boldsymbol{W}}+\frac{1}{2\gamma_2}\widetilde{\boldsymbol{V}}^{\mathrm{T}}\widetilde{\boldsymbol{V}}$$

其中，$\gamma_1>0$，$\gamma_2>0$。

对 $L$ 求导，综合式(5.18)，可得

$$\begin{aligned}\dot{L}&=s\dot{s}+\frac{1}{\gamma_1}\widetilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\widetilde{\boldsymbol{W}}}+\frac{1}{\gamma_2}\widetilde{\boldsymbol{V}}^{\mathrm{T}}\dot{\widetilde{\boldsymbol{V}}}\\&=s\{\widetilde{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}_f(\boldsymbol{x})-\varepsilon_f-\eta\mathrm{sgn}(s)+[\widetilde{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{h}_g(\boldsymbol{x})-\varepsilon_g]u-d(t)\}-\frac{1}{\gamma_1}\widetilde{\boldsymbol{W}}^{\mathrm{T}}\dot{\widetilde{\boldsymbol{W}}}-\frac{1}{\gamma_2}\widetilde{\boldsymbol{V}}^{\mathrm{T}}\dot{\widetilde{\boldsymbol{V}}}\\&=\widetilde{\boldsymbol{W}}^{\mathrm{T}}\left[s\boldsymbol{h}_f(\boldsymbol{x})-\frac{1}{\gamma_1}\dot{\widetilde{\boldsymbol{W}}}\right]+\widetilde{\boldsymbol{V}}^{\mathrm{T}}\left[s\boldsymbol{h}_g(\boldsymbol{x})u-\frac{1}{\gamma_2}\dot{\widetilde{\boldsymbol{V}}}\right]+s[-\varepsilon_f-\eta\mathrm{sgn}(s)-\varepsilon_g u-d(t)]\end{aligned}$$

设计自适应律为

$$\dot{\widetilde{\boldsymbol{W}}}=-\gamma_1 s\boldsymbol{h}_f(\boldsymbol{x}) \tag{5.20}$$

$$\dot{\widetilde{\boldsymbol{V}}}=-\gamma_2 s\boldsymbol{h}_g(\boldsymbol{x})u \tag{5.21}$$

则

$$\begin{aligned}\dot{L}&=s[-\varepsilon_f-\eta\mathrm{sgn}(s)-\varepsilon_g u-d(t)]\\&=[-\varepsilon_f-\varepsilon_g u-d(t)]s-\eta|s|\end{aligned}$$

由于逼近误差 $\varepsilon_f$ 和 $\varepsilon_g$ 可以限制得足够小，如取 $\eta\geqslant|\varepsilon_f+\varepsilon_g u+d(t)|$，则可得 $\dot{L}\leqslant 0$。

存在 $\eta_0>0$，$\eta\geqslant\eta_0+|\varepsilon_{Mf}|+|\varepsilon_{Mg}u|+D$，使得

$$\dot{L}\leqslant-\eta_0|s|\leqslant 0$$

由于 $L\geqslant 0$，$\dot{L}\leqslant 0$，从而 $s$、$\widetilde{\boldsymbol{W}}$ 和 $\widetilde{\boldsymbol{V}}$ 有界。当 $\dot{L}\equiv 0$ 时，$s=0$，根据 LaSalle 不变性原理，闭环系统为渐近稳定，当 $t\to\infty$ 时，$s\to 0$，从而 $e\to 0$，$\dot{e}\to 0$。$\dot{L}\leqslant 0$ 只能保证 $\widetilde{\boldsymbol{W}}$ 和 $\widetilde{\boldsymbol{V}}$ 有界，但无法得到 $\widetilde{\boldsymbol{W}}$ 和 $\widetilde{\boldsymbol{V}}$ 收敛于零，故 $\tilde{f}$ 和 $\tilde{g}$ 无法收敛于零。

## 5.3.2　仿真实例

被控对象为单级倒立摆，其动力学方程为

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

其中，$f(\boldsymbol{x})=\dfrac{g\sin x_1-mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l[4/3-m\cos^2 x_1/(m_c+m)]}$；$g(\boldsymbol{x})=\dfrac{\cos x_1/(m_c+m)}{l[4/3-m\cos^2 x_1/(m_c+m)]}$，其中 $x_1$ 和 $x_2$ 分别为摆角和摆速；$g=9.8\mathrm{m/s}^2$ 为重力加速度；$m_c=1\mathrm{kg}$ 为小车质量；$m=0.1\mathrm{kg}$为摆的质量；$l=0.5\mathrm{m}$ 为摆长的一半；$u$ 为控制输入。

取 $x_1=\theta$，期望轨迹为 $\theta_{\mathrm{d}}(t)=0.1\sin(t)$，系统的初始状态为 $[\pi/60,0]$。网络输入取 $x_1$ 和 $x_2$，考虑到 $x_1$ 和 $x_2$ 的取值范围，高斯函数的参数取 $\boldsymbol{c}_i=[-1.0\quad -0.5\quad 0\quad 0.5\quad 1.0]$和 $b_i=5.0$。网络的初始权值取 0.10。控制律采用式(5.17)，自适应律采用式(5.20)和式(5.21)，参数选为 $\gamma_1=10$，$\gamma_2=1.0$，$c=5.0$ 和 $\eta=0.01$。

仿真结果如图 5.5～图 5.7 所示。

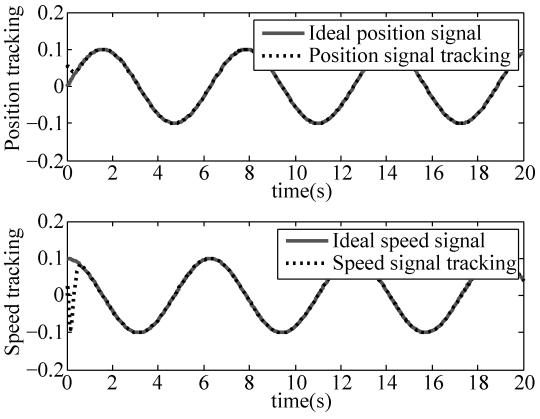



图 5.5　摆角和摆速跟踪

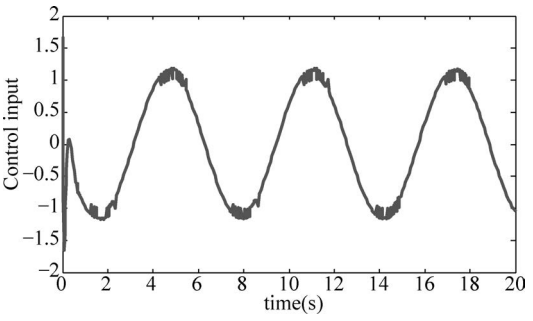



图 5.6　控制输入

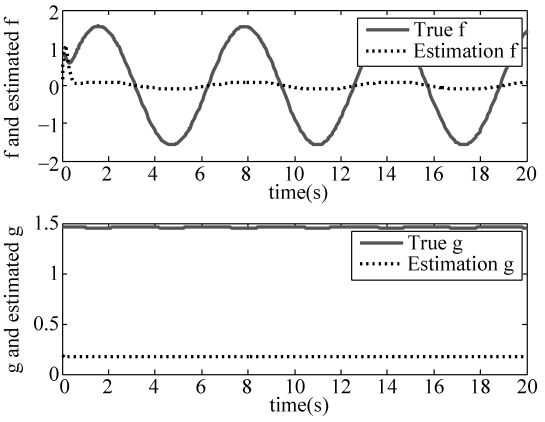



图 5.7　$f(\cdot)$和 $g(\cdot)$及逼近

仿真主程序为 chap5_2sim.mdl，详见附录。

## 5.4　基于神经网络最小参数学习法的自适应滑模控制

采用神经网络最小参数学习法[4,5]，取神经网络权值的上界估计值作为神经网络权值的估计值，通过设计参数估计自适应律代替神经网络权值的调整，自适应算法简单，便于实际工程应用。

### 5.4.1　问题描述

考虑如下二阶非线性系统：

$$\ddot{\theta}=f(\theta,\dot{\theta})+g(\theta,\dot{\theta})u+d(t) \tag{5.22}$$

其中，$f$ 为未知非线性函数；$g$ 为已知非线性函数；$u\in\mathbf{R}$ 和 $y=\theta\in\mathbf{R}$ 分别为系统的输入和输出；$d(t)$为外加干扰，$|d(t)|\leqslant D$。

设位置指令为 $\theta_d$，令 $e=\theta-\theta_d$，设计切换函数为

$$s=\dot{e}+ce \tag{5.23}$$

其中，$c>0$，则

$$\dot{s}=\ddot{e}+c\dot{e}=\ddot{\theta}+c\dot{e}-\ddot{\theta}_{\mathrm{d}}=f+gu+d-\ddot{\theta}_{\mathrm{d}}+c\dot{e} \tag{5.24}$$

在实际工程中，模型不确定项 $f$ 为未知，为此，控制律无法设计，需要对 $f$ 进行逼近。

### 5.4.2　基于 RBF 神经网络逼近的自适应控制

采用 RBF 神经网络对不确定项 $f$ 进行自适应逼近。RBF 神经网络算法为

$$h_j=\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_j\|^2}{2b_j^2}\right),\quad j=1,2,\cdots,m$$

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

其中，$\boldsymbol{x}$ 为网络的输入信号；$j$ 表示网络隐含层节点的个数；$\boldsymbol{h}=[h_1,h_2,\cdots,h_m]^{\mathrm{T}}$ 为高斯函数的输出；$\boldsymbol{W}$ 为理想神经网络权值；$\varepsilon$ 为神经网络逼近误差，$|\varepsilon|\leqslant\varepsilon_{\mathrm{N}}$。

采用 RBF 神经网络逼近 $f$，根据 $f$ 的表达式，网络输入取 $\boldsymbol{x}=\begin{bmatrix}\theta & \dot{\theta}\end{bmatrix}^{\mathrm{T}}$，RBF 神经网络的输出为

$$\hat{\boldsymbol{f}}(\boldsymbol{x})=\hat{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \tag{5.25}$$

采用神经网络最小参数学习法[5]，令 $\phi=\|\boldsymbol{W}\|^2$，$\phi$ 为正常数，$\hat{\phi}$ 为 $\phi$ 的估计，$\tilde{\phi}=\hat{\phi}-\phi$。

设计控制律为

$$u=\frac{1}{g}\left[-\frac{1}{2}s\hat{\phi}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+\ddot{\theta}_{\mathrm{d}}-c\dot{e}-\eta\mathrm{sgn}(s)-\mu s\right] \tag{5.26}$$

其中，$\hat{f}$ 为 RBF 神经网络的估计值；$\eta\geqslant\varepsilon_{\mathrm{N}}+D$；$\mu>0$。

将控制律式(5.26)代入式(5.24)，得

$$\dot{s}=\boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}+\varepsilon-\frac{1}{2}s\hat{\phi}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}-\eta\mathrm{sgn}(s)+d-\mu s \tag{5.27}$$

定义 Lyapunov 函数：

$$L=\frac{1}{2}s^2+\frac{1}{2\gamma}\tilde{\phi}^2$$

其中，$\gamma>0$。

对 $L$ 求导，并将式(5.26)和式(5.27)代入，得

$$\begin{aligned}\dot{L}=s\dot{s}+\frac{1}{\gamma}\tilde{\phi}\dot{\hat{\phi}}&=s\left[\boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}+\varepsilon-\frac{1}{2}s\hat{\phi}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}-\eta\mathrm{sgn}(s)+d-\mu s\right]+\frac{1}{\gamma}\tilde{\phi}\dot{\hat{\phi}}\\&\leqslant\frac{1}{2}s^2\phi\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+\frac{1}{2}-\frac{1}{2}s^2\hat{\phi}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+(\varepsilon+d)s-\eta|s|+\frac{1}{\gamma}\tilde{\phi}\dot{\hat{\phi}}-\mu s^2\\&=-\frac{1}{2}s^2\tilde{\phi}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+\frac{1}{2}+(\varepsilon+d)s-\eta|s|+\frac{1}{\gamma}\tilde{\phi}\dot{\hat{\phi}}-\mu s^2\\&=\tilde{\phi}\left(-\frac{1}{2}s^2\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+\frac{1}{\gamma}\dot{\hat{\phi}}\right)+\frac{1}{2}+(\varepsilon+d)s-\eta|s|-\mu s^2\\&\leqslant\tilde{\phi}\left(-\frac{1}{2}s^2\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+\frac{1}{\gamma}\dot{\hat{\phi}}\right)+\frac{1}{2}-\mu s^2\end{aligned}$$

设计自适应律为

$$\dot{\hat{\phi}}=\frac{\gamma}{2}s^2\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}-k\gamma\hat{\phi} \tag{5.28}$$

其中，$k>0$。

则

$$\dot{L}\leqslant-k\tilde{\phi}\hat{\phi}+\frac{1}{2}-\mu s^2\leqslant-\frac{k}{2}(\tilde{\phi}^2-\phi^2)+\frac{1}{2}-\mu s^2=-\frac{k}{2}\tilde{\phi}^2-\mu s^2+\left(\frac{k}{2}\phi^2+\frac{1}{2}\right)$$

则

$$\begin{aligned}\dot{L}&\leqslant-k\tilde{\phi}\hat{\phi}+\frac{1}{2}-\mu s^2\\&\leqslant-\frac{k}{2}(\tilde{\phi}^2-\phi^2)+\frac{1}{2}-\mu s^2\\&=-\frac{k}{2}\tilde{\phi}^2-\mu s^2+\left(\frac{k}{2}\phi^2+\frac{1}{2}\right)\\&\leqslant-\mu s^2+\left(\frac{k}{2}\phi^2+\frac{1}{2}\right)\end{aligned}$$

可见，当 $s^2\geqslant\frac{1}{\mu}\left(\frac{k}{2}\phi^2+\frac{1}{2}\right)$ 时，$\dot{L}\leqslant0$，闭环系统稳定，则闭环系统的收敛结果为

$$t\to\infty\text{ 时，}\quad|s|\leqslant\sqrt{\frac{1}{\mu}\left(\frac{k}{2}\phi^2+\frac{1}{2}\right)}$$

根据引理 4.1，可得

$$\lim_{t\to\infty}|e|\leqslant\frac{1}{c}\sqrt{\frac{1}{\mu}\left(\frac{k}{2}\phi^2+\frac{1}{2}\right)},\quad\lim_{t\to\infty}|\dot{e}|\leqslant2\sqrt{\frac{1}{\mu}\left(\frac{k}{2}\phi^2+\frac{1}{2}\right)}$$

可见，当取 $\mu$ 足够大，$t\to\infty$ 时，$s\to0$，从而 $e\to0$，$\dot{e}\to0$。

**注**：推导中采用了以下两个结论。

(1) $s^2\phi\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+1=s^2\|\boldsymbol{W}\|^2\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+1=s^2\|\boldsymbol{W}\|^2\|\boldsymbol{h}\|^2+1\geqslant s^2\|\boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}\|^2+1\geqslant2s\boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}$，即

$$s\boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}\leqslant\frac{1}{2}s^2\phi\boldsymbol{h}^{\mathrm{T}}\boldsymbol{h}+\frac{1}{2}$$

(2) 由于 $(\tilde{\phi}+\phi)^2\geqslant0$，则 $\tilde{\phi}^2+2\tilde{\phi}\phi+\phi^2\geqslant0$，$\tilde{\phi}^2+2\tilde{\phi}(\hat{\phi}-\tilde{\phi})+\phi^2\geqslant0$，即 $2\tilde{\phi}\hat{\phi}\geqslant\tilde{\phi}^2-\phi^2$。

采用神经网络最小参数学习法的不足之处为：由于采用了神经网络权值的上界作为神经网络权值的估计值，而且又利用了不等式进行了放大，所设计的控制算法过于保守。

### 5.4.3 仿真实例

被控对象取单级倒立摆，其动态方程如下：

$$\dot{x}_1=x_2$$

$$\dot{x}_2=\frac{g\sin x_1-mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l[4/3-m\cos^2x_1/(m_c+m)]}+\frac{\cos x_1/(m_c+m)}{l[4/3-m\cos^2x_1/(m_c+m)]}u$$

其中，$f(\cdot)=\dfrac{g\sin x_1-mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l[4/3-m\cos^2x_1/(m_c+m)]}$；$g(\cdot)=\dfrac{\cos x_1/(m_c+m)}{l[4/3-m\cos^2x_1/(m_c+m)]}$；

$x_1$ 和 $x_2$ 分别为摆角和摆速；$g=9.8\text{m/s}^2$；$m_c=1\text{kg}$ 为小车质量；$m_c=1\text{kg}$；$m$ 为摆杆质量；$m=0.1\text{kg}$；$l$ 为摆长的一半；$l=0.5\text{m}$；$u$ 为控制输入。

取 $x_1=\theta$，摆的角度指令为 $\theta_d=0.1\sin(t)$。倒立摆初始状态为 $[\pi/60,0]$，控制律取式(5.26)，取 $\eta=0.5$，$\mu=40$，自适应律取式(5.28)，自适应参数取 $\gamma=150$，$k=1.0$。在滑模函数中，取 $c=15$，采用 RBF 神经网络逼近函数 $f(x_1,x_2)$，网络结构为 2-5-1，根据网络输入的实际范围来设计高斯函数的参数，参数 $\boldsymbol{c}_j$ 和 $b_j$ 的取值分别为 $\begin{bmatrix}-1 & -0.5 & 0 & 0.5 & 1\\ -1 & -0.5 & 0 & 0.5 & 1\end{bmatrix}$ 和 5.0。仿真结果如图 5.8 和图 5.9 所示。仿真主程序为 chap5_3sim.mdl。

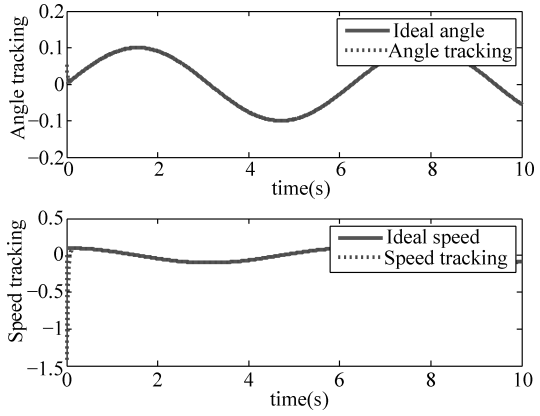



图 5.8　角度和角速度跟踪

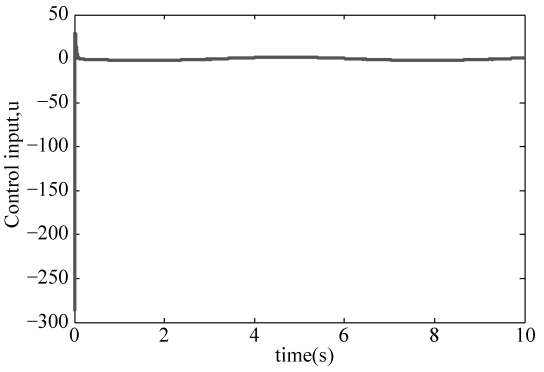
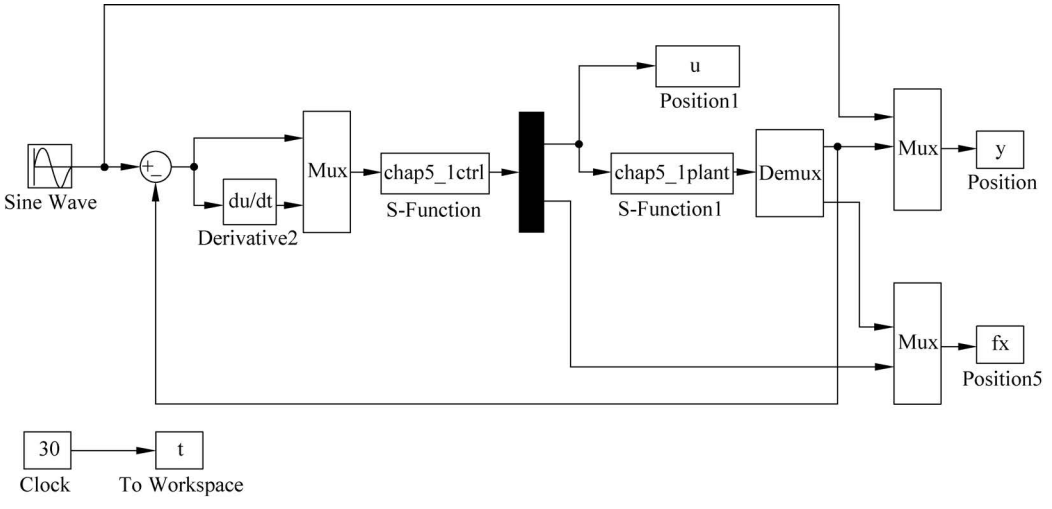



图 5.9　控制输入信号

# 附录　仿真程序

## 5.2 节的程序

### 1. 仿真主程序：chap5_1sim.mdl



**2. 控制律设计程序：chap5_1ctrl. m**

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
global cij bj c
sizes = simsizes;
sizes.NumContStates   = 5;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 2;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 0;
sys = simsizes(sizes);
x0 = 0*ones(1,5);
str = [];
ts = [];
cij = 0.10*[-1 -0.5 0 0.5 1;
            -1 -0.5 0 0.5 1];
bj = 5.0;
c = 15;
function sys = mdlDerivatives(t,x,u)
global cij bj c
e = u(1);
de = u(2);
s = c*e+de;

xi = [e;de];
h = zeros(5,1);
for j = 1:1:5
    h(j) = exp(-norm(xi-cij(:,j))^2/(2*bj^2));
end
gama = 0.015;
W = [x(1) x(2) x(3) x(4) x(5)]';
for i = 1:1:5
    sys(i) = -1/gama*s*h(i);
end
function sys = mdlOutputs(t,x,u)
global cij bj c
e = u(1);
de = u(2);
thd = 0.1*sin(t);
dthd = 0.1*cos(t);
ddthd = -0.1*sin(t);
x1 = thd-e;

s = c*e+de;
W = [x(1) x(2) x(3) x(4) x(5)]';
```

```
xi = [e;de];
h = zeros(5,1);
for j = 1:1:5
    h(j) = exp( - norm(xi - cij(:,j))^2/(2 * bj^2));
end
fn = W' * h;

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 < = 1.5
    xite = 1.0;
else
    xite = 0.10;
end
ut = 1/gx * ( - fn + ddthd + c * de + xite * sign(s));
sys(1) = ut;
sys(2) = fn;
```

### 3. 控制对象程序：chap5_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;
% % % % % % % % %
dt = 0 * 10 * sin(t);
% % % % % % % % %
```

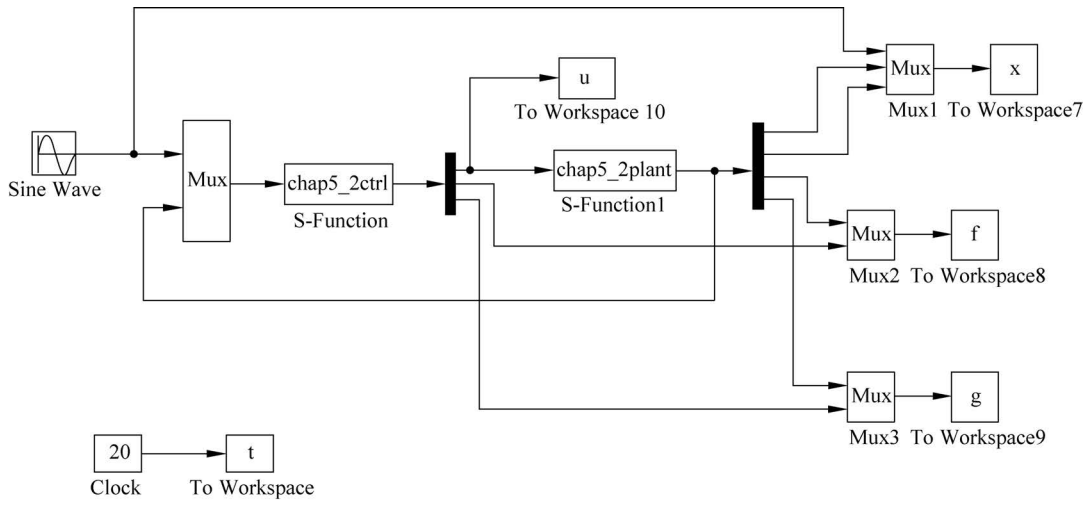
```
sys(1) = x(2);
sys(2) = fx + gx * u + dt;
function sys = mdlOutputs(t,x,u)
g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

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;
```

**4. 作图程序：chap5_1plot. m**

```
close all;

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

## 5.3 节的程序

**1. 仿真主程序：chap5_2sim. mdl**




**2. 控制律设计程序：chap5_2ctrl. m**

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
```

```
switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
global xite cij bj h c
sizes = simsizes;
sizes.NumContStates   = 10;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 3;
sizes.NumInputs       = 5;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 0;
sys = simsizes(sizes);
x0 = 0.1*ones(10,1);
str = [];
ts = [];
cij = [-1 -0.5 0 0.5 1;
       -1 -0.5 0 0.5 1];
bj = 5;
h = [0,0,0,0,0];
c = 5;
xite = 0.01;
function sys = mdlDerivatives(t,x,u)
global xite cij bj h c
thd = u(1);
dthd = 0.1*cos(t);
ddthd = -0.1*sin(t);

x1 = u(2);
x2 = u(3);
e = thd-x1;
de = dthd-x2;

s = c*e+de;

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

for i = 1:1:5
    wf(i,1) = x(i);
end
for i = 1:1:5
    wg(i,1) = x(i+5);
end
fxn = wf'*h';
gxn = wg'*h'+0.01;

ut = 1/gxn*(-fxn+ddthd+xite*sign(s)+c*de);
```

```
gama1 = 10;gama2 = 1.0;
S1 = -gama1 * s * h;
S2 = -gama2 * s * h * ut;
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 xite cij bj h c
thd = u(1);
dthd = 0.1 * cos(t);
ddthd = -0.1 * sin(t);

x1 = u(2);
x2 = u(3);
e = thd - x1;
de = dthd - x2;

s = c * e + de;

for i = 1:1:5
    wf(i,1) = x(i);
end
for i = 1:1:5
    wg(i,1) = x(i+5);
end

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

fxn = wf' * h';
gxn = wg' * h' + 0.01;

ut = 1/gxn * (-fxn + ddthd + xite * sign(s) + c * de);

sys(1) = ut;
sys(2) = fxn;
sys(3) = gxn;
```

### 3. 控制对象程序：chap5_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      = 4;
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;

sys(1) = x(2);
sys(2) = fx + gx * u;
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;
gx = cos(x(1))/(mc + m);
gx = gx/S;

sys(1) = x(1);
sys(2) = x(2);
sys(3) = fx;
sys(4) = gx;
```

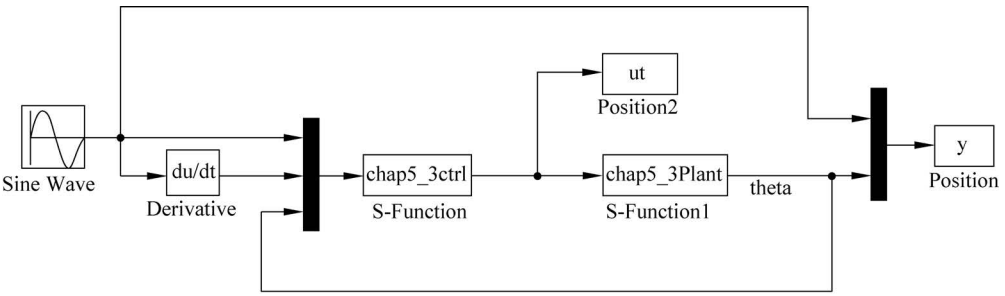
**4. 作图程序：chap5_2plot. m**

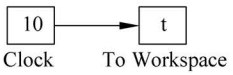
```
close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r',t,x(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('Position tracking');
legend('Ideal position signal','Position signal tracking');
subplot(212);
plot(t,0.1 * cos(t),'r',t,x(:,3),'k:','linewidth',2);
xlabel('time(s)');ylabel('Speed tracking');
legend('Ideal speed signal','Speed signal tracking');

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

figure(3);
subplot(211);
plot(t,f(:,1),'r',t,f(:,2),'k:','linewidth',2);
```

```
xlabel('time(s)');ylabel('f and estiamted f');
legend('True f','Estimation f');
subplot(212);
plot(t,g(:,1),'r',t,g(:,2),'k:','linewidth',2);
xlabel('time(s)');ylabel('g and estimated g');
legend('True g','Estimation g');
```

## 5.4 节的程序

### 1. Simulink 主程序：chap5_3sim.mdl




### 2. 控制器 S 函数：chap5_3ctrl.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
global cc bb c miu
sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = 0;
str = [];
ts = [];
cc = [-2 -1 0 1 2;
      -2 -1 0 1 2];
bb = 1;
c = 200;
```

```
miu = 30;
function sys = mdlDerivatives(t,x,u)
global cc bb c miu
x1d = u(1);
dx1d = u(2);
x1 = u(3);
x2 = u(4);

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

xi = [x1;x2];
h = zeros(5,1);
for j = 1:1:5
    h(j) = exp( - norm(xi - cc(:,j))^2/(2 * bb * bb));
end
gama = 150;
k = 1.0;
sys(1) = gama/2 * s^2 * h' * h - k * gama * x;

function sys = mdlOutputs(t,x,u)
global cc bb c miu
x1d = u(1);
dx1d = u(2);
x1 = u(3);
x2 = u(4);

e = x1 - x1d;
de = x2 - dx1d;
thd = 0.1 * sin(t);
dthd = 0.1 * cos(t);
ddthd = - 0.1 * sin(t);
s = c * e + de;

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

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

xite = 0.5;
miu = 40;

ut = 1/gx * ( - 0.5 * s * fi * h' * h + ddthd - c * de - xite * sign(s) - miu * s);
sys(1) = ut;
```

### 3. 被控对象 S 函数：chap5_3plant. m

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

```
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2, 4, 9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [pi/60 0];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
g = 9.8;mc = 1.0;m = 0.1;l = 0.5;
S = l*(4/3-m*(cos(x(1)))^2/(mc+m));
fx = g*sin(x(1))-m*l*x(2)^2*cos(x(1))*sin(x(1))/(mc+m);
fx = fx/S;
gx = cos(x(1))/(mc+m);
gx = gx/S;
%%%%%%%%%
dt = 0.1*10*sin(t);
%%%%%%%%%

sys(1) = x(2);
sys(2) = fx+gx*u+dt;
function sys = mdlOutputs(t,x,u)
g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

S = l*(4/3-m*(cos(x(1)))^2/(mc+m));
fx = g*sin(x(1))-m*l*x(2)^2*cos(x(1))*sin(x(1))/(mc+m);
fx = fx/S;

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

**4. 作图程序：chap5_3plot. m**

```
close all;

figure(1);
subplot(211);
plot(t,y(:,1),'r',t,y(:,2),'b:','linewidth',2);
xlabel('time(s)');ylabel('angle tracking');
legend('ideal angle','angle tracking');
subplot(212);
plot(t,0.1*cos(t),'r',t,y(:,3),'b:','linewidth',2);
xlabel('time(s)');ylabel('speed tracking');
```

```
legend('ideal speed','speed tracking');

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

## 参考文献


[1] HUANG S,TAN K K,TONG H L,et al. Adaptive Control of Mechanical Systems Using Neural Networks[J]. IEEE Transactions on Systems,Man and Cybernetics Part C,2014,37(5): 897-903.

[2] TSAI C H,CHUNG H Y,YU F M. Neuro-sliding mode control with its applications to seesaw systems [J]. IEEE Transactions Neural Networks,2004,15(1): 124-134.

[3] EDWARDS C,SPURGEON S. Sliding mode control: theory and applications[M]. London: Taylor & Francis,1998.

[4] YANG Y S,REN J S. Adaptive fuzzy robust tracking controller design via small gain approach and its application[J]. IEEE Transactions on Fuzzy Systems,2003,11(6): 783-795.

[5] CHEN B,LIU X P,LIU K F,et al. Direct adaptive fuzzy control of nonlinear strict-feedback systems [J]. Automatica,2009,45: 1530-1535.

[6] 刘金琨. 滑模变结构控制 MATLAB 仿真[M]. 2 版. 北京：清华大学出版社,2012.

[7] LIU J K,LU Y. Adaptive RBF neural network control of robot with actuator nonlinearities[J]. Journal of Control Theory and Applications,2010,8(2): 150-156.

[8] LASALLE J, LEFSCHETZ S. Stability by Lyapunov's direct method[M]. New York: Academic Press,1961.

[9] HASSAN K H. Nonlinear systems[M]. 3rd ed. New Jersey: Prentice Hall,2002.