

如果被控对象的数学模型已知,滑模控制器可以使系统输出直接跟踪期望指令,但较大的建模不确定性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。

将滑模控制结合神经网络逼近用于非线性系统的控制中,采用神经网络实现模型未知部分的自适应逼近,可有效地降低模糊增益。神经网络自适应律通过 Lyapunov 方法导出,通过自适应权重的调节保证整个闭环系统的稳定性和收敛性。

RBF 神经网络于 1988 年提出。相比多层前馈 BP 网络,RBF 网络由于具有良好的泛化能力,网络结构简单,避免不必要和冗长的计算而备受关注。关于 RBF 网络的研究表明 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^[1]。目前,已经有许多针对非线性系统的 RBF 神经网络控制研究成果发表。

3.1 一种简单的 RBF 网络自适应滑模控制

3.1.1 问题描述

考虑一种简单的动力学系统:

$$\ddot{\theta} = f(\theta, \dot{\theta}) + u \quad (3.1)$$

其中, θ 为转动角度, u 为控制输入。

写成状态方程形式为

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x) + u \end{cases} \quad (3.2)$$

其中, $x_1 = \theta, x_2 = \dot{\theta}, f(x)$ 为未知。

角度指令为 x_d , 则误差及其导数为

$$e = x_1 - x_d, \quad \dot{e} = x_2 - \dot{x}_d$$

定义滑模函数

$$s = ce + \dot{e}, \quad c > 0 \quad (3.3)$$

则

$$\dot{s} = c \dot{e} + \ddot{e} = c \dot{e} + \dot{x}_2 - \ddot{x}_d = c \dot{e} + f(x) + u - \ddot{x}_d$$

由式(3.3)可见,如果 $s \rightarrow 0$, 则 $e \rightarrow 0$ 且 $\dot{e} \rightarrow 0$ 。

3.1.2 RBF 网络原理

由于 RBF 网络具有万能逼近特性^[1], 采用 RBF 神经网络逼近 $f(x)$, 网络算法为

$$h_j = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2b_j^2}\right) \quad (3.4)$$

$$f = \mathbf{W}^{*T} \mathbf{h}(\mathbf{x}) + \epsilon \quad (3.5)$$

其中, $\mathbf{x}$ 为网络的输入, j 为网络隐含层第 j 个节点, $\mathbf{h} = [h_j]^T$ 为网络的高斯基函数输出, $\mathbf{W}^*$ 为网络的理想权值, ϵ 为网络的逼近误差, $\epsilon \leq \epsilon_N$ 。

网络输入取 $\mathbf{x} = [x_1 \quad x_2]^T$, 则网络输出为

$$\hat{f}(x) = \hat{\mathbf{W}}^T \mathbf{h}(x) \quad (3.6)$$

3.1.3 控制算法设计与分析

由于 $f(x) - \hat{f}(x) = \mathbf{W}^{*T} \mathbf{h}(x) + \epsilon - \hat{\mathbf{W}}^T \mathbf{h}(x) = -\tilde{\mathbf{W}}^T \mathbf{h}(x) + \epsilon$ 。

定义 Lyapunov 函数为

$$V = \frac{1}{2}s^2 + \frac{1}{2\gamma} \tilde{\mathbf{W}}^T \tilde{\mathbf{W}} \quad (3.7)$$

其中, $\gamma > 0$, $\tilde{\mathbf{W}} = \hat{\mathbf{W}} - \mathbf{W}^*$ 。

则

$$\begin{aligned} \dot{V} &= s \dot{s} + \frac{1}{\gamma} \tilde{\mathbf{W}}^T \dot{\tilde{\mathbf{W}}} \\ &= s(c \dot{e} + f(x) + u - \ddot{x}_d) + \frac{1}{\gamma} \tilde{\mathbf{W}}^T \dot{\tilde{\mathbf{W}}} \end{aligned}$$

设计控制律为

$$u = -c \dot{e} - \hat{f}(x) + \ddot{x}_d - \eta \operatorname{sgn}(s) \quad (3.8)$$

则

$$\begin{aligned} \dot{V} &= s(f(x) - \hat{f}(x) - \eta \operatorname{sgn}(s)) + \frac{1}{\gamma} \tilde{\mathbf{W}}^T \dot{\tilde{\mathbf{W}}} \\ &= s(-\tilde{\mathbf{W}}^T \mathbf{h}(x) + \epsilon - \eta \operatorname{sgn}(s)) + \frac{1}{\gamma} \tilde{\mathbf{W}}^T \dot{\tilde{\mathbf{W}}} \\ &= \epsilon s - \eta |s| + \tilde{\mathbf{W}}^T \left(\frac{1}{\gamma} \dot{\tilde{\mathbf{W}}} - s \mathbf{h}(x) \right) \end{aligned}$$

取 $\eta > |\epsilon|_{\max}$, 自适应律为

$$\dot{\hat{W}} = \gamma sh(x) \quad (3.9)$$

则 $\dot{V} = \epsilon s - \eta |s| \leq 0$ 。

可见, 控制律中的鲁棒项 $\eta \operatorname{sgn}(s)$ 的作用是克服神经网络的逼近误差, 以保证系统稳定。

由于当且仅当 $s=0$ 时, $\dot{V}=0$ 。即当 $\dot{V}=0$ 时, $s=0$ 。根据 LaSalle 不变性原理, 闭环系统为渐进稳定, 即当 $t \rightarrow \infty$ 时, $s \rightarrow 0$ 。系统的收敛速度取决于 η 。

由于 $V \geq 0, \dot{V} \leq 0$, 则当 $t \rightarrow \infty$ 时, V 有界, 因此, 可以证明 $\hat{W}$ 有界, 但无法保证 $\hat{W}$ 收敛于 W^* 。

3.1.4 仿真实例

考虑如下被控对象

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

其中, $f(x) = 10x_1x_2$ 。

被控对象的初始状态取 $[0.15 \ 0]$, 位置指令为 $x_d = \sin t$, 控制律采用式 (3.8), 自适应律采用式 (3.9), 取 $\gamma = 1500, \eta = 1.5$ 。根据网络输入 x_1 和 x_2 的实际范围来设计高斯基函数的参数^[2], 参数 c_j 和 b_j 取值分别为 $0.5 \times [-2 \ -1 \ 0 \ 1 \ 2]$ 和 3.0。仿真程序中为了避免混淆, 将 $s = ce + \dot{e}$ 中的 c 写为 λ , 取 $\lambda = 10$ 。网络权值中各个元素的初始值取 0.10。仿真结果如图 3-1 和图 3-2 所示。

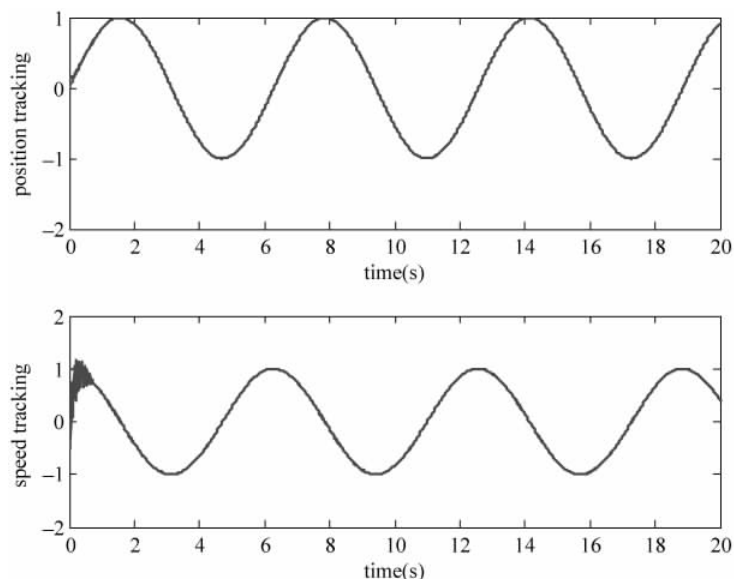


图 3-1 角度和角速度跟踪

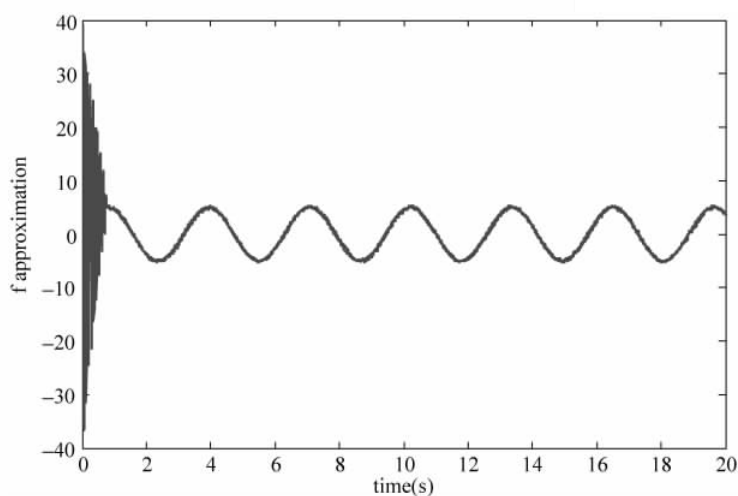
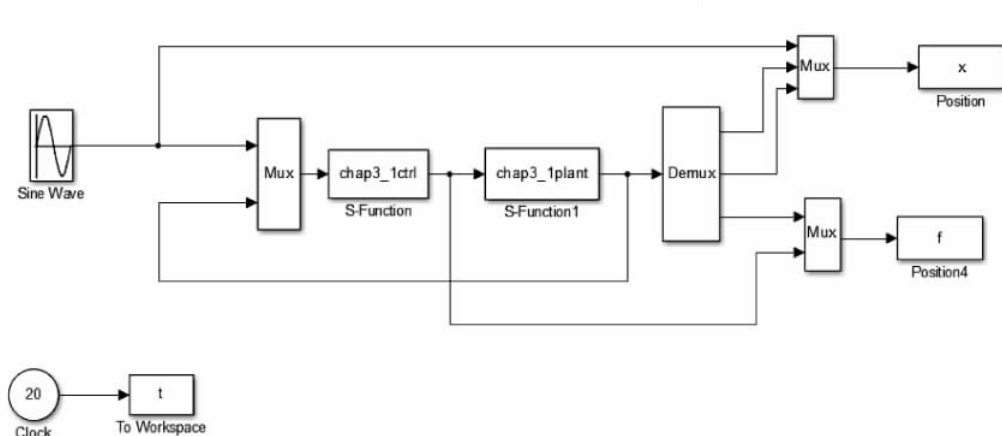


图 3-2 $f(x)$ 及逼近

仿真程序如下：

(1) Simulink 主程序：chap3_1sim.mdl



(2) 控制律及自适应律 S 函数：chap3_1ctrl.m

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

```

function [sys,x0,str,ts]=mdlInitializeSizes
global b c lama
sizes = simsizes;
sizes.NumContStates = 5;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = 0.1 * ones(1,5);
str = [];
ts = [0 0];
c = 0.5 * [-2 -1 0 1 2;
          -2 -1 0 1 2];
b = 3.0;
lama = 10;
function sys = mdlDerivatives(t,x,u)
global b c lama
xd = sin(t);
dxd = cos(t);

x1 = u(2);
x2 = u(3);
e = x1 - xd;
de = x2 - dxd;
s = lama * e + de;

W = [x(1) x(2) x(3) x(4) x(5)]';
xi = [x1;x2];

h = zeros(5,1);
for j = 1:1:5
    h(j) = exp(- norm(xi - c(:,j))^2/(2 * b ^ 2));
end

gama = 1500;
for i = 1:1:5
    sys(i) = gama * s * h(i);
end

function sys = mdlOutputs(t,x,u)
global b c lama
xd = sin(t);
dxd = cos(t);
ddxd = - sin(t);

x1 = u(2);
x2 = u(3);
e = x1 - xd;
de = x2 - dxd;

```

```

s = lama * e + de;

W = [x(1) x(2) x(3) x(4) x(5)];
xi = [x1;x2];

h = zeros(5,1);
for j = 1:1:5
    h(j) = exp(- norm(xi - c(:,j))^2/(2 * b^2));
end
fn = W * h;
xite = 1.50;

% fn = 10 * x1 + x2;    % Precise f
ut = - lama * de + ddx - fn - xite * sign(s);

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

```

(3) 被控对象 S 函数: 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 = 3;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.15;0];
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
ut = u(1);

f = 10 * x(1) * x(2);
sys(1) = x(2);
sys(2) = f + ut;

```

```
function sys = mdlOutputs(t,x,u)
f = 10 * x(1) * x(2);
```

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

(4) 作图程序: chap3_1plot.m

```
close all;

figure(1);
subplot(211);
plot(t,x(:,1),'r',t,x(:,2),'b','linewidth',2);
xlabel('time(s)');ylabel('position tracking');
subplot(212);
plot(t,cos(t),'r',t,x(:,3),'b','linewidth',2);
xlabel('time(s)');ylabel('speed tracking');

figure(2);
plot(t,f(:,1),'r',t,f(:,3),'b','linewidth',2);
xlabel('time(s)');ylabel('f approximation');
```

3.2 基于 RBF 网络逼近的机械手自适应控制

通过对文献[3]的控制方法进行详细推导及仿真分析,研究一类机械臂神经网络自适应控制的设计方法。

3.2.1 问题的提出

设 n 关节机械手方程为

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \quad (3.10)$$

其中, $M(q)$ 为 $n \times n$ 阶正定惯性矩阵, $C(q,\dot{q})$ 为 $n \times n$ 阶惯性矩阵, 其中, $G(q)$ 为 $n \times 1$ 阶惯性向量, $F(\dot{q})$ 为摩擦力, τ_d 为未知外加干扰, τ 为控制输入。

跟踪误差为

$$e(t) = q_d(t) - q(t)$$

定义误差函数为

$$r = \dot{e} + \Delta e \quad (3.11)$$

其中, $\Delta = \Delta^T > 0$, 则

$$\begin{aligned} \dot{q} &= -r + \dot{q}_d + \Delta e \\ M\dot{r} &= M(\ddot{q}_d - \ddot{q} + \Delta\dot{e}) = M(\ddot{q}_d + \Delta\dot{e}) - M\ddot{q} \\ &= M(\ddot{q}_d + \Delta\dot{e}) + C\dot{q} + G + F + \tau_d - \tau \\ &= M(\ddot{q}_d + \Delta\dot{e}) - Cr + C(\dot{q}_d + \Delta e) + G + F + \tau_d - \tau \\ &= -Cr - \tau + f + \tau_d \end{aligned} \quad (3.12)$$

其中, $f(x) = M(\ddot{q}_d + \mathbf{A}\dot{e}) + C(q_d + \mathbf{A}e) + G + F$ 。

在实际工程中,模型不确定项 f 为未知,为此,需要对不确定项 f 进行逼近。

采用 RBF 网络逼近 f , 根据 $f(x)$ 的表达式,网络输入取

$$x = [e^T \quad \dot{e}^T \quad q_d^T \quad \dot{q}_d^T \quad \ddot{q}_d^T]$$

设计控制律为

$$\tau = \hat{f} + K_v r \quad (3.13)$$

其中, $\hat{f}(x)$ 为 RBF 网络的估计值。

将控制律式(3.13)代入式(3.12),得

$$\begin{aligned} M\dot{r} &= -Cr - \hat{f} - K_v r + f + \tau_d \\ &= -(K_v + C)r + \tilde{f} + \tau_d = -(K_v + C)r + \zeta_0 \end{aligned} \quad (3.14)$$

其中, $\tilde{f} = f - \hat{f}$, $\zeta_0 = \tilde{f} + \tau_d$ 。

如果定义 Lyapunov 函数

$$L = \frac{1}{2} r^T M r$$

则

$$\begin{aligned} \dot{L} &= r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2C) r + r^T \zeta_0 \\ \dot{L} &= r^T \zeta_0 - r^T K_v r \end{aligned}$$

这说明在 K_v 固定条件下,控制系统的稳定依赖于 ζ_0 , 即 $\hat{f}$ 对 f 的逼近精度及干扰 τ_d 的大小。

采用 RBF 网络对不确定项 f 进行逼近。理想的 RBF 网络算法为

$$\phi_j = g(\|x - c_i\|^2 / b_j^2)$$

$$y = W\phi(x)$$

其中, x 为网络的输入信号, $\phi = [\phi_1 \quad \phi_2 \quad \cdots \quad \phi_n]$, j 为隐层节点的个数, i 为网络输入的个数。

3.2.2 基于 RBF 神经网络逼近的控制器

1. 控制器的设计

采用 RBF 网络逼近 f , 则 RBF 神经网络的输出为

$$\hat{f}(x) = \hat{W}^T \phi(x) \quad (3.15)$$

取

$$\tilde{W} = W - \hat{W}, \quad \|W\|_F \leq W_{\max}$$

设计控制律为

$$\tau = \hat{W}^T \phi(x) + K_v r - v \quad (3.16)$$

其中, v 为用于克服神经网络逼近误差 ϵ 的鲁棒项。

将控制律式(3.16)代入式(3.12),得

$$M\dot{r} = -(K_v + C)r + \tilde{W}^T \phi(x) + (\epsilon + \tau_d) + v = -(K_v + C)r + \zeta_1 \quad (3.17)$$

其中, $\zeta_1 = \tilde{W}^T \phi(x) + (\epsilon + \tau_d) + v$ 。

2. 稳定性及收敛性分析

根据控制律式(3.16)中是否有 $v(t)$ 项, ϵ 和 τ_d 是否存在以及神经网络自适应律设计的不同,系统的收敛性不同。

1) 取 $v(t)=0$, 存在 ϵ 和 τ_d 的情况

定义 Lyapunov 函数

$$L = \frac{1}{2} r^T M r + \frac{1}{2} \text{tr}(\tilde{W}^T F^{-1} \tilde{W}) \quad (3.18)$$

则

$$\dot{L} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + \text{tr}(\tilde{W}^T F^{-1} \dot{\tilde{W}})$$

将式(3.17)代入上式,得

$$\dot{L} = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2C) r + \text{tr} \tilde{W}^T (F^{-1} \dot{\tilde{W}} + \phi r^T) + r^T (\epsilon + \tau_d) \quad (3.19)$$

考虑机械手特性,并取

$$\dot{\tilde{W}} = -F \phi r^T$$

即神经网络自适应律为

$$\dot{\hat{W}} = F \phi r^T \quad (3.20)$$

则

$$\dot{L} = -r^T K_v r + r^T (\epsilon + \tau_d) \leq -K_{vmin} \|r\|^2 + (\epsilon_N + b_d) \|r\|$$

其中, $\|\epsilon\| \leq \epsilon_N$, $\|\tau_d\| \leq b_d$ 。

当满足下列收敛条件时, $\dot{L} \leq 0$:

$$\|r\| \geq (\epsilon_N + b_d) / K_{vmin} \quad (3.21)$$

2) 取 $v(t)=0, \epsilon=0, \tau_d=0$ 的情况

Lyapunov 函数为

$$L = \frac{1}{2} r^T M r + \frac{1}{2} \text{tr}(\tilde{W}^T F^{-1} \tilde{W}) \quad (3.22)$$

此时控制律和自适应律为

$$\tau = \hat{W}^T \phi(x) + K_v r \quad (3.23)$$

$$\dot{\hat{W}} = F \phi r^T \quad (3.24)$$

由式(3.17)知

$$M\dot{r} = -(K_v + C)r + \tilde{W}^T \phi(x)$$

则

$$\dot{L} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r = -r^T K_v r \leq 0$$

3) 取 $v(t)=0$, 存在 $\mathbf{e}$ 和 $\mathbf{e}_d$, 自适应律采取 UUB 的形式
Lyapunov 函数和控制律取式(3.22)和式(3.23)。

自适应律为

$$\dot{\hat{\mathbf{W}}} = \mathbf{F}\hat{\mathbf{p}}\mathbf{r}^T - k\mathbf{F} \|\mathbf{r}\| \hat{\mathbf{W}} \quad (3.25)$$

则根据式(3.19), 有

$$\dot{L} = -\mathbf{r}^T \mathbf{K}_v \mathbf{r} + \frac{1}{2} \mathbf{r}^T (\dot{\mathbf{M}} - 2\mathbf{C}) \mathbf{r} + \text{tr} \tilde{\mathbf{W}}^T (\mathbf{F}^{-1} \dot{\tilde{\mathbf{W}}} + \hat{\mathbf{p}} \mathbf{r}^T) + \mathbf{r}^T (\mathbf{e} + \mathbf{e}_d)$$

将式(3.25)代入上式, 得

$$\begin{aligned} \dot{L} &= -\mathbf{r}^T \mathbf{K}_v \mathbf{r} + \text{tr} \tilde{\mathbf{W}}^T (-\hat{\mathbf{p}} \mathbf{r}^T + k \|\mathbf{r}\| \hat{\mathbf{W}} + \hat{\mathbf{p}} \mathbf{r}^T) + \mathbf{r}^T (\mathbf{e} + \mathbf{e}_d) \\ &= -\mathbf{r}^T \mathbf{K}_v \mathbf{r} + k \|\mathbf{r}\| \text{tr} \tilde{\mathbf{W}}^T (\mathbf{W} - \tilde{\mathbf{W}}) + \mathbf{r}^T (\mathbf{e} + \mathbf{e}_d) \end{aligned}$$

由于

$$\text{tr} \tilde{\mathbf{W}}^T (\mathbf{W} - \tilde{\mathbf{W}}) = (\tilde{\mathbf{W}}, \mathbf{W})_F - \|\mathbf{W}\|_F^2 \leq \|\tilde{\mathbf{W}}\|_F \|\mathbf{W}\|_F - \|\tilde{\mathbf{W}}\|_F^2$$

则

$$\begin{aligned} \dot{L} &\leq -K_{vmin} \|\mathbf{r}\|^2 + k \|\mathbf{r}\| \|\tilde{\mathbf{W}}\|_F (W_{max} - \|\tilde{\mathbf{W}}\|_F) + (\epsilon_N + b_d) \|\mathbf{r}\| \\ &= -\|\mathbf{r}\| (K_{vmin} \|\mathbf{r}\| + k \|\tilde{\mathbf{W}}\|_F (\|\tilde{\mathbf{W}}\|_F - W_{max}) - (\epsilon_N + b_d)) \end{aligned}$$

由于

$$\begin{aligned} &K_{vmin} \|\mathbf{r}\| + k \|\tilde{\mathbf{W}}\|_F (\|\tilde{\mathbf{W}}\|_F - W_{max}) - (\epsilon_N + b_d) \\ &= k (\|\tilde{\mathbf{W}}\|_F - W_{max}/2)^2 - kW_{max}^2/4 + K_{vmin} \|\mathbf{r}\| - (\epsilon_N + b_d) \end{aligned}$$

则要使 $\dot{L} \leq 0$, 需要

$$\|\mathbf{r}\| \geq \frac{kW_{max}^2/4 + (\epsilon_N + b_d)}{K_{vmin}} \quad (3.26)$$

或

$$\|\tilde{\mathbf{W}}\|_F \geq W_{max}/2 + \sqrt{W_{max}^2/4 + (\epsilon_N + b_d)/k} \quad (3.27)$$

4) 存在 ϵ 和 $\mathbf{e}_d$, 考虑鲁棒项 $v(t)$ 设计的情况

将鲁棒项 v 设计为

$$\mathbf{v} = -(\epsilon_N + b_d) \text{sgn}(\mathbf{r}) \quad (3.28)$$

控制律取式(3.16), 神经网络自适应律取式(3.20)。

定义 Lyapunov 函数为

$$L = \frac{1}{2} \mathbf{r}^T \mathbf{M} \mathbf{r} + \frac{1}{2} \text{tr}(\tilde{\mathbf{W}}^T \mathbf{F}^{-1} \tilde{\mathbf{W}})$$

则

$$\dot{L} = \mathbf{r}^T \mathbf{M} \dot{\mathbf{r}} + \frac{1}{2} \mathbf{r}^T \dot{\mathbf{M}} \mathbf{r} + \text{tr}(\tilde{\mathbf{W}}^T \mathbf{F}^{-1} \dot{\tilde{\mathbf{W}}})$$

将(3.17)式代入上式, 得

$$\dot{L} = -\mathbf{r}^T \mathbf{K}_v \mathbf{r} + \frac{1}{2} \mathbf{r}^T (\dot{\mathbf{M}} - 2\mathbf{C}) \mathbf{r} + \text{tr} \tilde{\mathbf{W}}^T (\mathbf{F}^{-1} \dot{\tilde{\mathbf{W}}} + \hat{\mathbf{p}} \mathbf{r}^T) + \mathbf{r}^T (\mathbf{e} + \mathbf{e}_d + \mathbf{v})$$

则

$$\dot{L} = -r^T K_v r + r^T (\mathbf{e} + \mathbf{r}_d + \mathbf{v})$$

由于

$$r^T (\mathbf{e} + \mathbf{r}_d + \mathbf{v}) = r^T (\mathbf{e} + \mathbf{r}_d) + r^T \mathbf{v} = r^T (\mathbf{e} + \mathbf{r}_d) - \|r\| (\mathbf{e}_N + b_d) \leq 0$$

则

$$\dot{L} \leq 0$$

针对以上4种情况,由于当 $\dot{L} \equiv 0$ 时, $r \equiv 0$,根据LaSalle不变性原理,闭环系统渐近稳定, $t \rightarrow \infty$ 时, $r \rightarrow 0$ 。由于 $L \geq 0, \dot{L} \leq 0$,则 L 有界,从而 $\tilde{W}$ 有界,但无法保证 $\tilde{W}$ 收敛于0。

3.2.3 针对 $f(x)$ 中各项分别进行神经网络逼近

控制律为

$$\mathbf{r} = \hat{W}^T \boldsymbol{\phi}(x) + K_v r - \mathbf{v} \quad (3.29)$$

鲁棒项 $\mathbf{v}$ 取式(3.28)。由式(3.12)知,被控对象中的 $f(x)$ 项可写为

$$f(x) = M(q)\ddot{\mathbf{q}}_d + C(q, \dot{\mathbf{q}})\dot{\mathbf{q}}_d + G(q) + F(\dot{\mathbf{q}})$$

其中, $\boldsymbol{\xi}_1(t) = \ddot{\mathbf{q}}_d + \mathbf{A}\dot{\mathbf{e}}, \boldsymbol{\xi}_2(t) = \dot{\mathbf{q}}_d + \mathbf{A}\mathbf{e}$ 。

采用RBF神经网络,可以对 $f(x)$ 中的各项分别进行逼近:

$$\hat{M}(q) = W_M^T \boldsymbol{\phi}_M(q)$$

$$\hat{C}(q, \dot{q}) = W_V^T \boldsymbol{\phi}_V(q, \dot{q})$$

$$\hat{G}(q) = W_G^T \boldsymbol{\phi}_G(q)$$

$$\hat{F}(\dot{q}) = W_F^T \boldsymbol{\phi}_F(\dot{q})$$

则

$$\hat{f}(x) = [W_M^T \boldsymbol{\xi}_1(t) \quad W_V^T \boldsymbol{\xi}_2(t) \quad W_G^T \quad W_F^T] \begin{bmatrix} \boldsymbol{\phi}_M \\ \boldsymbol{\phi}_V \\ \boldsymbol{\phi}_G \\ \boldsymbol{\phi}_F \end{bmatrix} \quad (3.30)$$

$$\text{其中, } \boldsymbol{\phi}(x) = \begin{bmatrix} \boldsymbol{\phi}_M \\ \boldsymbol{\phi}_V \\ \boldsymbol{\phi}_G \\ \boldsymbol{\phi}_F \end{bmatrix}, W^T = [W_M^T \quad W_V^T \quad W_G^T \quad W_F^T].$$

自适应律为

$$\dot{\hat{W}}_M = F_M \boldsymbol{\phi}_M r^T - k_M F_M \|r\| \hat{W}_M \quad (3.31)$$

$$\dot{\hat{\mathbf{W}}}_V = \mathbf{F}_V \boldsymbol{\phi}_V \mathbf{r}^T - k_V \mathbf{F}_V \|\mathbf{r}\| \hat{\mathbf{W}}_V \quad (3.32)$$

$$\dot{\hat{\mathbf{W}}}_G = \mathbf{F}_G \boldsymbol{\phi}_G \mathbf{r}^T - k_G \mathbf{F}_G \|\mathbf{r}\| \hat{\mathbf{W}}_G \quad (3.33)$$

$$\dot{\hat{\mathbf{W}}}_F = \mathbf{F}_F \boldsymbol{\phi}_F \mathbf{r}^T - k_F \mathbf{F}_F \|\mathbf{r}\| \hat{\mathbf{W}}_F \quad (3.34)$$

其中, $k_M > 0, k_V > 0, k_G > 0, k_F > 0$ 。

稳定性分析如下:

定义 Lyapunov 函数为

$$\begin{aligned} L = & \frac{1}{2} \mathbf{r}^T \mathbf{M} \mathbf{r} + \frac{1}{2} \text{tr}(\tilde{\mathbf{W}}_M^T \mathbf{F}_M^{-1} \tilde{\mathbf{W}}_M) + \frac{1}{2} \text{tr}(\tilde{\mathbf{W}}_V^T \mathbf{F}_V^{-1} \tilde{\mathbf{W}}_V) \\ & + \frac{1}{2} \text{tr}(\tilde{\mathbf{W}}_G^T \mathbf{F}_G^{-1} \tilde{\mathbf{W}}_G) + \frac{1}{2} \text{tr}(\tilde{\mathbf{W}}_F^T \mathbf{F}_F^{-1} \tilde{\mathbf{W}}_F) \end{aligned}$$

则

$$\begin{aligned} \dot{L} = & \mathbf{r}^T \mathbf{M} \dot{\mathbf{r}} + \frac{1}{2} \mathbf{r}^T \dot{\mathbf{M}} \mathbf{r} + \text{tr}(\tilde{\mathbf{W}}_M^T \mathbf{F}_M^{-1} \dot{\tilde{\mathbf{W}}}_M) + \text{tr}(\tilde{\mathbf{W}}_V^T \mathbf{F}_V^{-1} \dot{\tilde{\mathbf{W}}}_V) \\ & + \text{tr}(\tilde{\mathbf{W}}_G^T \mathbf{F}_G^{-1} \dot{\tilde{\mathbf{W}}}_G) + \text{tr}(\tilde{\mathbf{W}}_F^T \mathbf{F}_F^{-1} \dot{\tilde{\mathbf{W}}}_F) \end{aligned}$$

将式(3.17)代入上式,得

$$\begin{aligned} \dot{L} = & -\mathbf{r}^T \mathbf{K}_v \mathbf{r} + \frac{1}{2} \mathbf{r}^T (\dot{\mathbf{M}} - 2\mathbf{V}_m) \mathbf{r} + \mathbf{r}^T (\boldsymbol{\epsilon} + \boldsymbol{\epsilon}_d) + \mathbf{r}^T \mathbf{v} + \text{tr} \tilde{\mathbf{W}}_M^T (\mathbf{F}_M^{-1} \dot{\tilde{\mathbf{W}}}_M + \boldsymbol{\phi}_M \mathbf{r}^T) \\ & + \text{tr} \tilde{\mathbf{W}}_V^T (\mathbf{F}_V^{-1} \dot{\tilde{\mathbf{W}}}_V + \boldsymbol{\phi}_V \mathbf{r}^T) + \text{tr} \tilde{\mathbf{W}}_G^T (\mathbf{F}_G^{-1} \dot{\tilde{\mathbf{W}}}_G + \boldsymbol{\phi}_G \mathbf{r}^T) + \text{tr} \tilde{\mathbf{W}}_F^T (\mathbf{F}_F^{-1} \dot{\tilde{\mathbf{W}}}_F + \boldsymbol{\phi}_F \mathbf{r}^T) \end{aligned} \quad (3.35)$$

考虑机械手特性,并将神经网络自适应律式(3.31)~式(3.34)代入上式,得

$$\begin{aligned} \dot{L} = & -\mathbf{r}^T \mathbf{K}_v \mathbf{r} + k_M \|\mathbf{r}\| \text{tr} \tilde{\mathbf{W}}_M^T (\mathbf{W}_M - \tilde{\mathbf{W}}_M) + k_V \|\mathbf{r}\| \text{tr} \tilde{\mathbf{W}}_V^T (\mathbf{W}_V - \tilde{\mathbf{W}}_V) \\ & + k_G \|\mathbf{r}\| \text{tr} \tilde{\mathbf{W}}_G^T (\mathbf{W}_G - \tilde{\mathbf{W}}_G) + k_F \|\mathbf{r}\| \text{tr} \tilde{\mathbf{W}}_F^T (\mathbf{W}_F - \tilde{\mathbf{W}}_F) + \mathbf{r}^T (\boldsymbol{\epsilon} + \boldsymbol{\epsilon}_d) + \mathbf{r}^T \mathbf{v} \end{aligned}$$

由于

$$\text{tr} \tilde{\mathbf{W}}^T (\mathbf{W} - \tilde{\mathbf{W}}) = (\tilde{\mathbf{W}}, \mathbf{W})_F - \|\mathbf{W}\|_F^2 \leq \|\tilde{\mathbf{W}}\|_F \|\mathbf{W}\|_F - \|\tilde{\mathbf{W}}\|_F^2$$

考虑鲁棒项(3.28),则

$$\begin{aligned} \dot{L} \leq & -K_{vmin} \|\mathbf{r}\|^2 + k_M \|\mathbf{r}\| \|\tilde{\mathbf{W}}_M\|_F (W_{Mmax} - \|\tilde{\mathbf{W}}_M\|_F) \\ & + k_V \|\mathbf{r}\| \|\tilde{\mathbf{W}}_V\|_F (W_{Vmax} - \|\tilde{\mathbf{W}}_V\|_F) \\ & + k_G \|\mathbf{r}\| \|\tilde{\mathbf{W}}_G\|_F (W_{Gmax} - \|\tilde{\mathbf{W}}_G\|_F) + k_M \|\mathbf{r}\| \|\tilde{\mathbf{W}}_F\|_F (W_{Fmax} - \|\tilde{\mathbf{W}}_F\|_F) \\ = & -\|\mathbf{r}\| [K_{vmin} \|\mathbf{r}\| + k_M \|\tilde{\mathbf{W}}_M\|_F (\|\tilde{\mathbf{W}}_M\|_F - W_{Mmax}) \\ & + k_V \|\tilde{\mathbf{W}}_V\|_F (\|\tilde{\mathbf{W}}_V\|_F - W_{Vmax}) + k_G \|\tilde{\mathbf{W}}_G\|_F (\|\tilde{\mathbf{W}}_G\|_F - W_{Gmax}) \\ & + k_F \|\tilde{\mathbf{W}}_F\|_F (\|\tilde{\mathbf{W}}_F\|_F - W_{Fmax})] \end{aligned}$$

由于

$$k \|\tilde{\mathbf{W}}\|_F (\|\tilde{\mathbf{W}}\|_F - W_{\max}) = k (\|\tilde{\mathbf{W}}\|_F - W_{\max}/2)^2 - kW_{\max}^2/4$$

要使 $\dot{L} \leq 0$, 需要

$$\|\mathbf{r}\| \geq \frac{k_M W_{M\max}^2/4 + k_V W_{V\max}^2/4 + k_G W_{G\max}^2/4 + k_F W_{F\max}^2/4}{K_{\min}} \quad (3.36)$$

或

$$\|\tilde{\mathbf{W}}_M\|_F \geq W_{M\max} \text{ 且 } \|\tilde{\mathbf{W}}_V\|_F \geq W_{V\max} \text{ 且 } \|\tilde{\mathbf{W}}_G\|_F \geq W_{G\max} \text{ 且 } \|\tilde{\mathbf{W}}_F\|_F \geq W_{F\max} \quad (3.37)$$

由式(3.35)可见, 由于 $\dot{L} \leq 0$ 当 $\dot{L} \equiv 0$ 时, $\mathbf{r} \equiv 0$, 根据 LaSalle 不变集原理, 当 $t \rightarrow \infty$ 时, $\mathbf{r} \rightarrow 0$ 。由于 $L \geq 0, \dot{L} \leq 0$, 则 L 有界, 从而 $\mathbf{r}$ 和 $\tilde{\mathbf{W}}_i$ 有界, 但无法保证 $\tilde{\mathbf{W}}_i$ 收敛于零。

3.2.4 仿真实例

选择二关节机械臂系统, 其动力学模型为

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\dot{\mathbf{q}}) + \boldsymbol{\tau}_d = \boldsymbol{\tau}$$

其中,

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} p_1 + p_2 + 2p_3 \cos q_2 & p_2 + p_3 \cos q_2 \\ p_2 + p_3 \cos q_2 & p_2 \end{bmatrix}$$

$$\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -p_3 \dot{q}_2 \sin q_2 & -p_3 (\dot{q}_1 + \dot{q}_2) \sin q_2 \\ p_3 \dot{q}_1 \sin q_2 & 0 \end{bmatrix}$$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} p_4 g \cos q_1 + p_5 g \cos(q_1 + q_2) \\ p_5 g \cos(q_1 + q_2) \end{bmatrix}$$

$$\mathbf{F}(\dot{\mathbf{q}}) = 0.2 \operatorname{sgn}(\dot{\mathbf{q}}), \quad \boldsymbol{\tau}_d = [0.1 \sin(t) \quad 0.1 \sin(t)]^T$$

取 $\mathbf{p} = [p_1, p_2, p_3, p_4, p_5] = [2.9, 0.76, 0.87, 3.04, 0.87]$ 。RBF 网络高斯基函数参数的取值对神经网络控制的作用很重要, 如果参数取值不合适, 将使高斯基函数无法得到有效的映射, 从而导致 RBF 网络无效。故 c 按网络输入值的范围取值, 取 $b_j = 0.20$,

$$c = 0.1 \begin{bmatrix} -1.5 & -1 & -0.5 & 0 & 0.5 & 1 & 1.5 \\ -1.5 & -1 & -0.5 & 0 & 0.5 & 1 & 1.5 \\ -1.5 & -1 & -0.5 & 0 & 0.5 & 1 & 1.5 \\ -1.5 & -1 & -0.5 & 0 & 0.5 & 1 & 1.5 \\ -1.5 & -1 & -0.5 & 0 & 0.5 & 1 & 1.5 \end{bmatrix}, \text{ 网络的初始权值取零, 网络输入取 } \mathbf{z} =$$

$$[e \quad \dot{e} \quad q_d \quad \dot{q}_d \quad \ddot{q}_d]^T。$$

系统的初始状态为 $[0.09 \ 0 \ -0.09 \ 0]$, 两个关节的角度指令分别为 $q_{1d} = 0.1\sin t, q_{2d} = 0.1\sin t$, 控制参数取 $K_v = \text{diag}\{20, 20\}, F = \text{diag}\{1.5, 1.5\}, \Delta = \text{diag}\{5, 5\}$, 在鲁棒项中, 取 $\epsilon_N = 0.20, b_d = 0.10$ 。

采用 Simulink 和 S 函数进行控制系统的设计, 神经网络权值矩阵中任意元素初值取 0.10。总体逼近控制器子程序 chap3_9ctrl.m, 按 3.2.2 节第 4 种情况设计控制律, 控制律取式(3.16), v 取式(3.28), 自适应律取式(3.20)。采用总体逼近控制器, 仿真结果如图 3-3~图 3-6 所示。

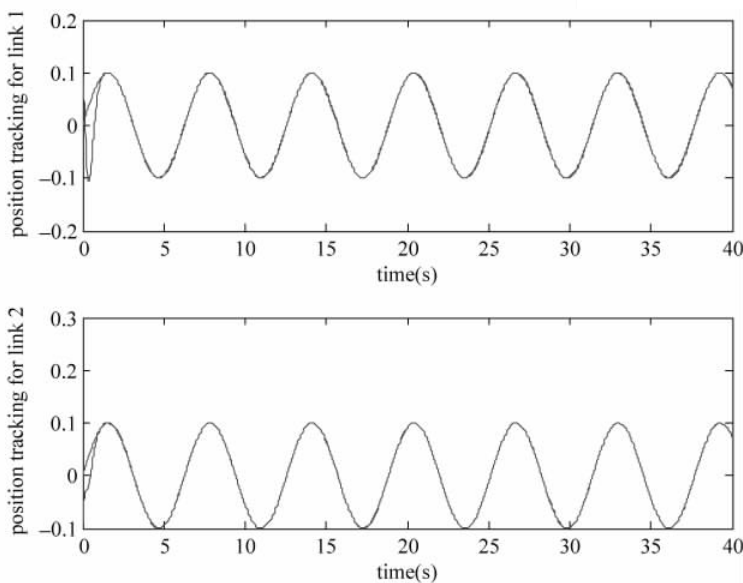


图 3-3 关节 1 及关节 2 的角度跟踪

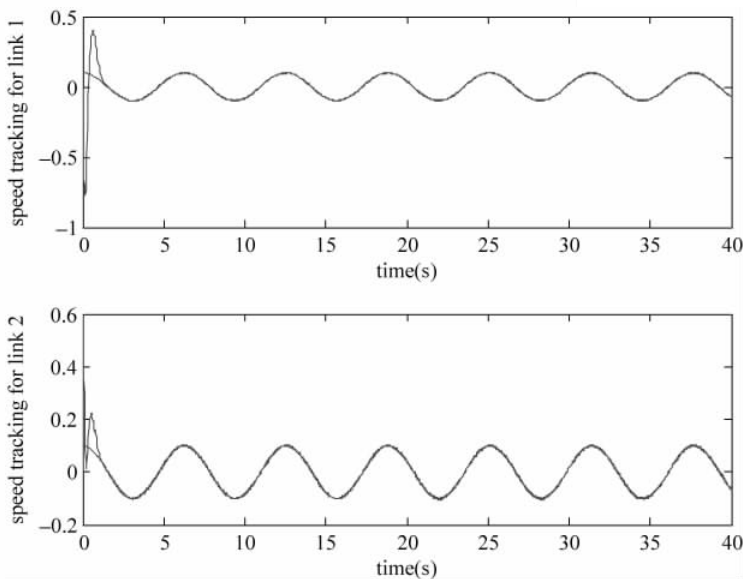


图 3-4 关节 1 及关节 2 的角速度跟踪

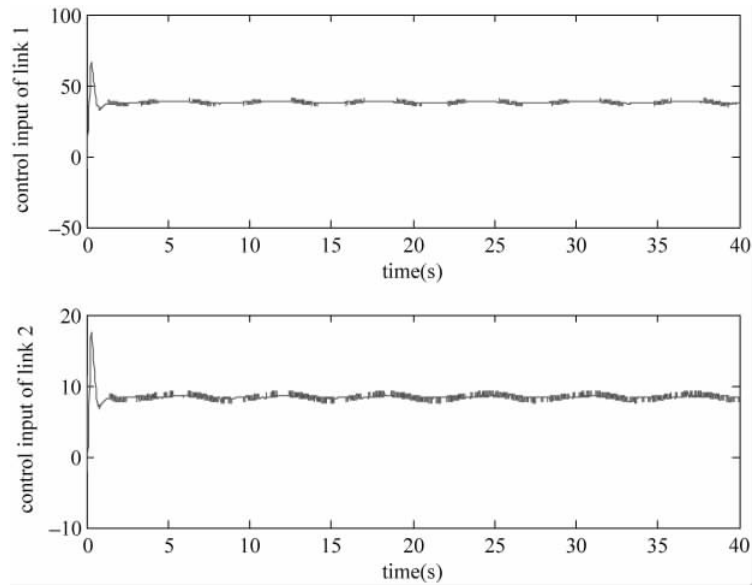


图 3-5 关节 1 及关节 2 的控制输入

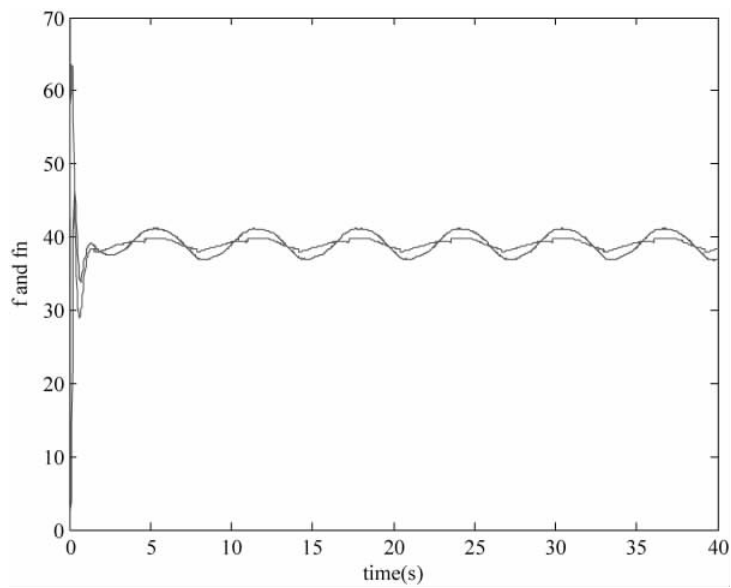
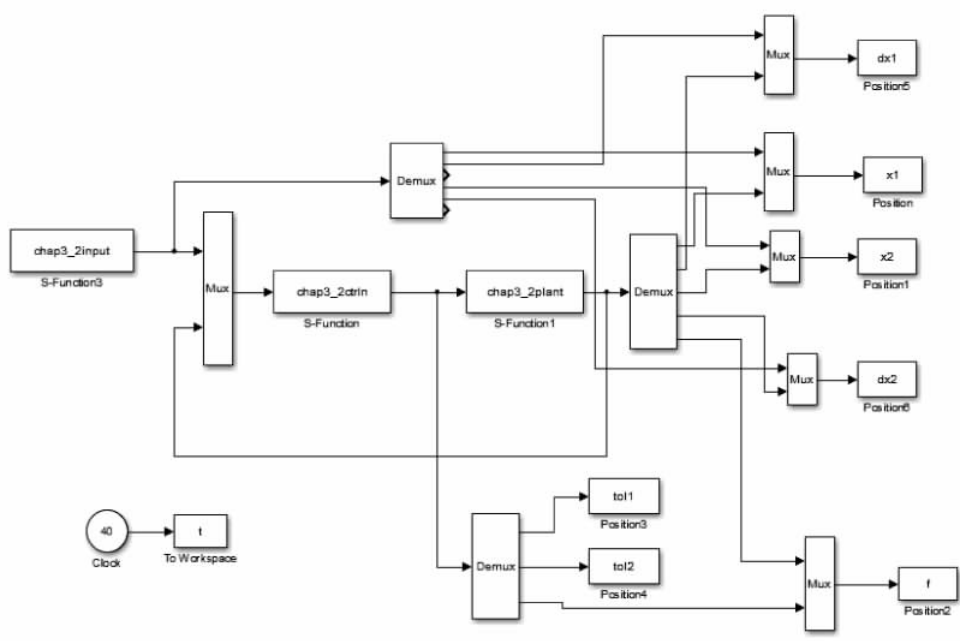


图 3-6 关节 1 及关节 2 的 $\|f(x)\|$ 及其逼近 $\|\hat{f}(x)\|$

仿真程序如下：

- (1) Simulink 主程序：chap3_2sim.mdl
- (2) 位置指令子程序：chap3_2input.m



```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 6;
sizes.NumInputs = 0;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys = mdlOutputs(t,x,u)
qd1 = 0.1 * sin(t);
d_qd1 = 0.1 * cos(t);
dd_qd1 = -0.1 * sin(t);
```



```

qd2 = 0.1 * sin(t);
d_qd2 = 0.1 * cos(t);
dd_qd2 = -0.1 * sin(t);

```

```

sys(1) = qd1;
sys(2) = d_qd1;
sys(3) = dd_qd1;
sys(4) = qd2;
sys(5) = d_qd2;
sys(6) = dd_qd2;

```

(3) 总体逼近控制器子程序: chap3_2ctrl.m

```

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

```

```

function [sys,x0,str,ts] = mdlInitializeSizes
global node c b Fai
node = 7;
c = 0.1 * [-1.5 -1 -0.5 0 0.5 1 1.5;
           -1.5 -1 -0.5 0 0.5 1 1.5;
           -1.5 -1 -0.5 0 0.5 1 1.5;
           -1.5 -1 -0.5 0 0.5 1 1.5;
           -1.5 -1 -0.5 0 0.5 1 1.5];
b = 10;
Fai = 5 * eye(2);

```

```

sizes = simsizes;
sizes.NumContStates = 2 * node;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 11;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = 0.1 * ones(1,2 * node);
str = [];
ts = [];
function sys = mdlDerivatives(t,x,u)
global node c b Fai

```

```

qd1 = u(1);
d_qd1 = u(2);
dd_qd1 = u(3);
qd2 = u(4);
d_qd2 = u(5);
dd_qd2 = u(6);

q1 = u(7);
d_q1 = u(8);
q2 = u(9);
d_q2 = u(10);

q = [q1;q2];

e1 = qd1 - q1;
e2 = qd2 - q2;
de1 = d_qd1 - d_q1;
de2 = d_qd2 - d_q2;
e = [e1;e2];
de = [de1;de2];
r = de + Fai * e;

qd = [qd1;qd2];
dq = [d_qd1;d_qd2];
dqr = dq + Fai * e;
ddqd = [dd_qd1;dd_qd2];
ddqr = ddqd + Fai * de;

z1 = [e(1);de(1);qd(1);dq(1);ddqd(1)];
z2 = [e(2);de(2);qd(2);dq(2);ddqd(2)];
for j = 1:1:node
    h1(j) = exp(-norm(z1 - c(:,j))^2/(b * b));
    h2(j) = exp(-norm(z2 - c(:,j))^2/(b * b));
end

F = 1.5 * eye(node);

for i = 1:1:node
    sys(i) = 1.5 * h1(i) * r(1);
    sys(i + node) = 1.5 * h2(i) * r(2);
end

function sys = mdlOutputs(t,x,u)
global node c b Fai

qd1 = u(1);
d_qd1 = u(2);
dd_qd1 = u(3);
qd2 = u(4);
d_qd2 = u(5);
dd_qd2 = u(6);
    
```

```

q1 = u(7);
d_q1 = u(8);
q2 = u(9);
d_q2 = u(10);

q = [q1;q2];

e1 = qd1 - q1;
e2 = qd2 - q2;
de1 = d_qd1 - d_q1;
de2 = d_qd2 - d_q2;
e = [e1;e2];
de = [de1;de2];
r = de + Fai * e;

qd = [qd1;qd2];
dq = [d_qd1;d_qd2];
dqr = dq + Fai * e;
ddq = [dd_qd1;dd_qd2];
ddqr = ddq + Fai * de;

z = [e;de;qd;dq;ddq];
W_f1 = [x(1:node)]';
W_f2 = [x(node + 1:node * 2)]';

z1 = [e(1);de(1);qd(1);dq(1);ddq(1)];
z2 = [e(2);de(2);qd(2);dq(2);ddq(2)];
for j = 1:1:node
    h1(j) = exp(-norm(z1 - c(:,j))^2/(b * b));
    h2(j) = exp(-norm(z2 - c(:,j))^2/(b * b));
end

fn = [W_f1 * h1';
      W_f2 * h2'];
Kv = 20 * eye(2);

epN = 0.20;
bd = 0.1;
v = -(epN + bd) * sign(r);
tol = fn + Kv * r - v;
fn_norm = norm(fn);

sys(1) = tol(1);
sys(2) = tol(2);
sys(3) = fn_norm;

(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
global p g
sizes = simsizes;
sizes.NumContStates = 4;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 5;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.09 0 -0.09 0];
str = [];
ts = [];

p = [2.9 0.76 0.87 3.04 0.87];
g = 9.8;
function sys = mdlDerivatives(t,x,u) % From Ge book program
global p g

M = [p(1) + p(2) + 2 * p(3) * cos(x(3)) p(2) + p(3) * cos(x(3));
     p(2) + p(3) * cos(x(3)) p(2)];
V = [-p(3) * x(4) * sin(x(3)) -p(3) * (x(2) + x(4)) * sin(x(3));
     p(3) * x(2) * sin(x(3)) 0];
G = [p(4) * g * cos(x(1)) + p(5) * g * cos(x(1) + x(3));
     p(5) * g * cos(x(1) + x(3))];
dq = [x(2);x(4)];
F = 0.2 * sign(dq);
told = [0.1 * sin(t);0.1 * sin(t)];

tol = u(1:2);

S = inv(M) * (tol - V * dq - G - F - told);

sys(1) = x(2);
sys(2) = S(1);
sys(3) = x(4);
sys(4) = S(2);
function sys = mdlOutputs(t,x,u)
global p g

```

```

M = [p(1) + p(2) + 2 * p(3) * cos(x(3)) p(2) + p(3) * cos(x(3));
      p(2) + p(3) * cos(x(3)) p(2)];
V = [-p(3) * x(4) * sin(x(3)) -p(3) * (x(2) + x(4)) * sin(x(3));
      p(3) * x(2) * sin(x(3)) 0];
G = [p(4) * g * cos(x(1)) + p(5) * g * cos(x(1) + x(3));
      p(5) * g * cos(x(1) + x(3))];
dq = [x(2); x(4)];
F = 0.2 * sign(dq);
told = [0.1 * sin(t); 0.1 * sin(t)];

qd1 = sin(t);
d_qd1 = cos(t);
dd_qd1 = -sin(t);
qd2 = sin(t);
d_qd2 = cos(t);
dd_qd2 = -sin(t);
qd1 = 0.1 * sin(t);
d_qd1 = 0.1 * cos(t);
dd_qd1 = -0.1 * sin(t);
qd2 = 0.1 * sin(t);
d_qd2 = 0.1 * cos(t);
dd_qd2 = -0.1 * sin(t);

q1 = x(1);
d_q1 = dq(1);
q2 = x(3);
d_q2 = dq(2);
q = [q1; q2];
e1 = qd1 - q1;
e2 = qd2 - q2;
de1 = d_qd1 - d_q1;
de2 = d_qd2 - d_q2;
e = [e1; e2];
de = [de1; de2];
Fai = 5 * eye(2);
dqdd = [d_qd1; d_qd2];
dqdr = dqdd + Fai * e;
ddqdd = [dd_qd1; dd_qd2];
ddqdr = ddqdd + Fai * de;
f = M * ddqdr + V * dqdr + G + F;
f_norm = norm(f);

sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
sys(4) = x(4);
sys(5) = f_norm;

```

(5) 绘图子程序: chap3_2plot.m

```
close all;
```

```
figure(1);
subplot(211);
plot(t,x1(:,1),'r',t,x1(:,2),'b');
xlabel('time(s)');ylabel('position tracking for link 1');
subplot(212);
plot(t,x2(:,1),'r',t,x2(:,2),'b');
xlabel('time(s)');ylabel('position tracking for link 2');

figure(2);
subplot(211);
plot(t,dx1(:,1),'r',t,dx1(:,2),'b');
xlabel('time(s)');ylabel('speed tracking for link 1');
subplot(212);
plot(t,dx2(:,1),'r',t,dx2(:,2),'b');
xlabel('time(s)');ylabel('speed tracking for link 2');

figure(3);
subplot(211);
plot(t,tol1(:,1),'r');
xlabel('time(s)');ylabel('control input of link 1');
subplot(212);
plot(t,tol2(:,1),'r');
xlabel('time(s)');ylabel('control input of link 2');

figure(4);
plot(t,f(:,1),'r',t,f(:,2),'b');
xlabel('time(s)');ylabel('f and fn');
```

参 考 文 献

- [1] J. Park, I. W. Sandberg. Universal Approximation Using Radial Basis Function Networks[J]. Neural Computation, 1991, 3(2): 246-257.
- [2] 刘金琨. RBF 神经网络自适应控制 MATLAB 仿真. 北京: 清华大学出版社, 2014.
- [3] Lewis F L, Liu K, Yesildirek A. Neural Net Robot Controller with Guaranteed Tracking. Performance. IEEE Transactions on Neural Networks, 1995, 6(3): 703-715.